(12) INNOVATION PATENT
(19) AUSTRALIAN PATENT OFFICE

(11) Application No. AU 2015101031 A4

(54) Title
SYSTEM AND A METHOD FOR MODELLING THE PERFORMANCE OF INFORMATION SYSTEMS

(51) International Patent Classification(s)
G06F 9/455 (2006.01)

(21) Application No: **2015101031** (22) Date of Filing: **2015.07.31**

(45) Publication Date: **2016.03.10**
(45) Publication Journal Date: **2016.03.10**
(45) Granted Journal Date: **2016.03.10**

(71) Applicant(s)
Performance Assurance Pty Ltd

(72) Inventor(s)
Brebner, Paul Charles;Gray, Jonathan Patrick

(74) Agent / Attorney
Performance Assurance Pty Ltd, Nicta 7A London Cct, Canberra, ACT, 2601

2015101031 31 Jul 2015

Abstract

The present invention provides a systems and methods for modelling and predicting the performance and scalability of information systems. The parts of the system are a domain specific performance modelling language, a graphical modelling interface, a module for managing models, a transformation module for converting graphical models into simulation models, a simulation engine and associated metrics calculation modules, and a simulation graphical interface. The system provides a graphical tool for creating performance models of a plurality of components of information systems, including workloads, simple and composite services, workflows and servers. Each component may be associated with a plurality of measured or experimental performance parameters. The system provides a transformation engine to automatically convert graphical performance models to run-time simulation models, and provides a discrete event simulator to process the run-time simulation models and to compute a plurality of predicted performance and scalability metrics which are graphically displayed.

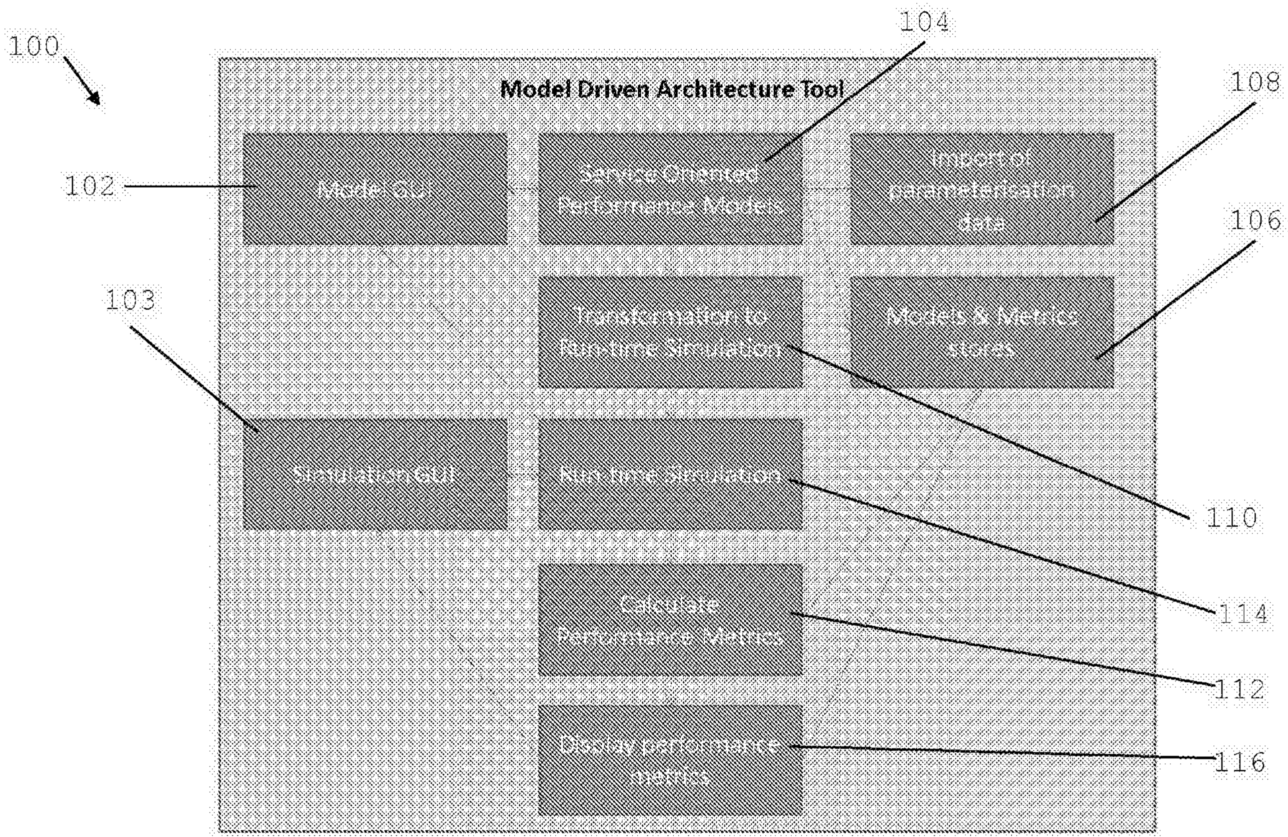

100
Model Driven Architecture Tool
102
Model GUI
104
Service Oriented Performance Models
108
Import of parameterisation data
106
Models & Metrics stores
110
Transformation to Run-time Simulation
103
Simulation GUI
Run-time Simulation
114
Calculate Performance Metrics
112
Display performance metrics
116

FIGURE 1

2015101031 31 Jul 2015

SYSTEM AND A METHOD FOR MODELLING THE PERFORMANCE OF INFORMATION SYSTEMS

Technical Field of the Invention

The present invention relates to a systems and methods for modelling the performance of information systems. In particular the invention is directed to a technology for modelling the performance and scalability of service oriented architectures.

Background of the Invention

Service oriented architectures (SOA) have been crucial for the implementation of distributed systems in the last few years. Evolving Enterprise Service Oriented Architectures (ESOAs) often have distinctive architectural characteristics which are due to their history and previous evolution, and the impact of ongoing and future evolution. The prediction of performance implications of evolution (including changes to architecture, services, infrastructure, use, etc) is crucial for designing SOAs systems. The fundamental aim of SOA is to facilitate IT agility by implementing business capabilities by using interoperable interfaces.

In recent years, governments around the world, including the Australian Government, have launched e-Government initiatives. E-Government refers to the government's use of information technologies to exchange information and services with citizens, business, and other arms of government. Although there are differences in emphasis from one country to another, these initiatives aim to:

provide more responsive, convenient, and easier access to government information and services by citizens; reduce the cost and time for business when interacting with government; and seek efficiency gains across government agencies through rationalisation of systems and increased interoperability. Many government agencies currently have a legacy of IT systems developed over several decades. However, these have typically been developed in isolation and to agency specific requirements, and are not usually designed to integrate with other agencies and external systems. When e-Government solutions are built on top of these legacy systems, and services are delivered to citizens and business through the internet, the performance demands on the legacy systems can often exceed their original design capacity, placing the e-Government solution at serious risk of failure.

Increasingly, e-systems are being designed as SOAs. They are implemented as composite service applications (services of services), consuming both internal services and external services provided by other agencies, and therefore function in the dual roles as service providers and consumers. Because of their critical role in the delivery of services to citizens and business, it is vital to understand the performance and scalability limits of these SOA systems well in advance of switch-on. Demand for the service, particularly a new service not previously offered by government, is often hard to predict, and a mismatch between demand and capacity can lead to cases of system "meltdown".

Load testing is a common strategy used to measure the capacity of traditional software systems, but it is often technically difficult to load test SOA applications end-to-end due to problems including: testing across organizational boundaries; security requirements; lack of tools and skills; high overhead of turning on low-level performance monitoring; the presence of resources that are shared with other organizations and/or production systems; the use of services provided by other organizations. Load testing may be perceived as a denial of service attack, SLAs may impose restrictions on use, or there may be a cost to use a service. By the time that integration testing is conducted on a production ready system it is inevitably too late and too expensive to radically change the software architecture to address performance and scalability deficiencies. It is therefore critical to predict the performance implications of architectural alternatives for SOAs early in the development lifecycle.

A series of Open Source and commercial tools for modelling of SOAs are available. However, all the existing modelling tools do not offer enough flexibility to model SOAs implemented over existent legacy architectures. Existing tools require extensive custom programming, or ad-hoc error-prone pre-processing of architectural and performance data prior to use. Further, they do not support composable modelling of service compositions, or shared or virtual resourcing models and are not sufficiently interactive. There is therefore a need in the art for a flexible and interactive tool which provides adaptability and easy to use graphic interfaces for SOAs performance modelling.

Summary of the Invention

2015101031 31 Jul 2015

In accordance to a first aspect, the present invention provides a system for simulating the performance of an operating environment, the system converting received parameters of the operating environment into run time structures for execution in a simulation engine and providing data representing the performance of the operating environment, the simulation engine responding in real time to changes in the received parameters. The data may be provided dynamically. The parameters may identify capabilities of the components of the operating environment, which may be a service oriented computer architecture. The components of the operating environment may be modelled in the system in order to be simulated in the simulation engine.

In accordance to a second aspect, the present invention provides a method for simulating the performance of an operating environment, comprising the steps of:

receiving parameters of an operating environment;

converting the parameters into run time structures for execution in a simulation environment;

running the run time structures in the simulation environment;

producing data representing the performance of the operating environment, wherein the simulation engine responds in real time to changes in the received parameters.

In accordance to a third aspect, the present invention provides a service oriented performance modelling tool comprising means for creating analytical models of a plurality of components of service oriented computer architectures, the models being suitable to be simulated by a simulation engine. Each component may be associated with a plurality of parameters. In some embodiments, the plurality of different components comprises: workloads, services, workflows and servers. In embodiments, each workload component is associated with at least a parameter from a list of parameters, the list of parameters comprising: name, number of users, time interval, arrival distribution and workflow. In embodiments, each workflow component comprises a series of steps, each step being associated with at least a service and may be associated with a semantics from a semantics list, the semantic lists comprising sequential semantics, parallel semantics, probabilistic semantics and split semantics. In embodiments, the services comprise simple services and composites services and each server is associated with at least a service and at least a processing unit.

In accordance to a fourth aspect, the present invention provides a run-time data structures generator which generates a run-time environment for analytical models created by a service oriented performance modelling tool in accordance to an aspect of the invention. In some embodiments, the run-time data structures generator may comprises a network of queues and servers. The run-time data structures generator may further comprise a plurality of simulation objects, each simulation object being associated with a workflow stack. In embodiments, each workflow of each workflow stack comprises a list of steps comprising a service name and a workflow type and each queue comprises a plurality of simulation objects and is associated with at least a server. In some embodiments, each server is associated with an input queue and an output queue and may be configured to perform actions. In embodiments, the run-time data structures generator further comprises a scheduling module to manage the processing of simulation objects in the input queue and an output queue of each server.

In accordance to a fifth aspect, the present invention provides a transformation engine to transform service oriented performance models to run-time data structures comprising a plurality of transformation rules to transform each component of the service oriented performance model into a run-time component. In embodiments, the transformation rules transform each component of the service oriented performance model into a run-time queue or a run-time server, each workload into a queue/server component with the queue filled with one instance of the workload, and each simple service into a queue with the name of the service and each server into a queue network server with same name and number of CPUs.

In accordance to a sixth aspect, the present invention provides a discrete event simulator to process discrete events in a sequence comprising a discrete event simulation engine which processes run-time data structures according to an aspect of the invention. In embodiments, the discrete event simulator comprises a metrics module which creates metrics for component of a run-time performance model. The metrics may comprise: arrival rate, throughput, response time, wait time, service demand, concurrency, server Utilisation and server BusyCPUs.

In accordance to a seventh aspect, the present invention provides a discrete event simulation method comprising the steps of:

creating system metrics;

creating workloads;

entering a plurality of simple services into servers;

completing processing of a plurality of services in servers;

entering events on a workflow engine;

executing the workflow engine and entering events onto a composite server;

executing composite servers.

In accordance to an eighth aspect, the present invention provides a system for modelling the performance of service oriented architectures, the system comprising:

a model graphical interface;

a simulation graphical interface;

a module for managing service oriented performance models;

a transformation module for transforming service oriented performance models into run-time data structures;

a metrics calculation modules for calculating metrics associated with run-time data structures; and

a metrics display module.

In embodiments, the model graphical interface is arranged to interact with a user. The user may perform a series of dynamic actions with the graphical interface to alter create, edit or transform a service oriented performance model. The simulation graphical interface may be arranged to interact with a user dynamically and respond in real-time. The user may perform a series of actions with the simulation interface to start or stop a simulation, modify simulation parameters, modify model parameters, or animate a model. The simulation parameters may be modified

2015101031 31 Jul 2015

dynamically during the simulation. The user may perform a series of actions with the simulation interface to select or edit metrics to be used during the simulation. The metrics may be selected or modified dynamically during the simulation.

Brief description of the drawings

Features and advantages of the present invention will become apparent from the following description of embodiments thereof, by way of example only, with reference to the accompanying drawings in which:

Figure 1 is a schematic representation of the principal modules of a Service Oriented Architectures simulation system in accordance with embodiments of the present invention;
Figure 2 is a flow-diagram of run-time data structures in accordance with embodiments of the present invention;
Figure 3 is a schematic representation of the functionalities of a transformation engine in accordance with embodiments of the present invention;
Figure 4 is a flow-diagram representing the main steps of a discrete event simulation engine in accordance with embodiments of the present invention;
Figure 5 shows two flow charts of the functionalities of servers in accordance with some embodiments of the invention.
Figure 6 is a schematic representation of a Service Oriented Performance Modelling (SOPM) Meta model describing the main components types, relationships and parameters in accordance with embodiments of the present invention;

Figure 7 is a screenshot of an example implementation of a graphical user interface in accordance with embodiments of the invention;

Figure 8 is a schematic representation of a computer system that implements a Service Oriented Architectures simulation system in accordance with embodiments of the present invention.

Detailed description of embodiments

Embodiments of the present invention relate to systems and methods for modelling service oriented architectures (SOA). The system supports the steps followed by software architects to enable them to easily produce SOA performance models, either from scratch or from existing architectural artifacts.

Models can be graphically visualized and parameterized with measured performance data, hardware capacity and optional configuration information. The system automatically provides an extensive set of performance and scalability parameters and metrics appropriate for each type and combination of model component. The model simulation can be run, paused, restarted and reset. While the model is running, selected metrics are computed continuously and graphed, and selected parameter values are graphed and can be changed giving immediate feedback. Multiple models can be run concurrently to compare architectural alternatives. Alternatives within a single model can also be compared (e.g. different workload ratios or workflow implementations). The system is intended for a range of architectural modelling tasks related to

performance and scalability, including capacity and resource planning, modelling of complex workloads, modelling complex composite applications, tuning of service deployment options and server configurations (e.g. virtual servers), comparing architectural alternatives, and developing Service Level Agreements (SLAs) for services consumed and provided.
The methods use a model-driven tool GUI for model development and visualization, and automatically transforms the model into a run-time form to be solved by a discrete event simulation engine to compute performance metrics for each component (including workloads, services, servers) and aggregations of components (including response times, throughput, concurrency, utilization, wait times, service demand, etc).

Embodiments of the present invention provide systems and methods for modelling the performance and scalability of Service Oriented Architectures (SOA) using the first order concepts of SOA systems, namely: Services (Simple and Composite) and Service Consumers (Workloads).
Further, embodiments of the invention provide a Service Oriented Performance Modelling (SOPM) tool, run-time data structures and a transformation tool for transforming SOPM structures to run-time data structures, and a simulation engine to simulate run-time data structures and to predict performance metrics.

In embodiments of the invention the components are imbedded in a custom Model Driven Architecture (MDA) tool framework which supports their definition, implementation, visualisation, modification, parameterisation, persistence, and transformation.

Referring now to figure 1, there is shown a schematic representation of the principal modules of a Service Oriented Architectures (SOA) modelling and simulation system 100 in accordance with embodiments of the present invention.

The service oriented performance models module 104 supports the definition of SOPM models, via a meta-model, and, in conjunction with the model graphic user interface (GUI) 102, creates new models, visualising models, editing models, parameterisation and checking of models (for correctness and completeness properties). The GUI 102 is provided to the system’s user to interact with service oriented models 104. In conjunction with other components, it supports users to graphically create, edit, save/load/print, parameterise and check models, and transform models to run-time version.

In conjunction with the SOPM models module 104 and the model GUI 102, the models and metrics stores module 106 enables persistence of model instances, parameters, and computed metrics to a variety of storage types and formats including local or remote binary files, XML, relational or object databases, bucket storage (E.g. Amazon S3), etc.

The import of parameterisation data module 108 acts as an interface to external sources of parameterisation data including file based data formats (e.g. log files, CSV files, Excel files, etc), and 3rd party monitoring tools (e.g. via APIs, proprietary file formats, XML files, etc). The import of parameterisation data module 108 enables relevant parameterisation data to be imported, selected, transformed and processed from the source formats to the target model formats and data types, and includes the ability to process the data statistically.

The transformation to run-time simulation module 110 performs the actual transformation of SOPM instances into run-time versions ready for run-time simulation. The transformation to run-time simulation module 110 allows keeping the operations of the underlying run-time data structures and simulation engine transparent to the users, limiting the amount of programming required.

After the model has been transformed into the run-time data structures it can be executed by the Simulation Engine component. The run-time simulation module 114 simulates the arrival and exit of users, the sequence of workflows associated with workloads, the calling and flow (via workflows) and completion of services, and the load on the servers.

The Simulation GUI works in conjunction with the Run-time Simulation Engine and enables simulations to be run (started, stopped, etc), in batch or interactive models, simulation and model parameters to be changed during simulation, models to be animated, and model metrics to be graphed.

The performance metrics calculation module 112 interacts with the run-time simulation and calculates the performance metrics from the simulation events depending on the component types. The current values of metrics are computed continuously and some metrics such as statistics are either computed continuously, periodically or once a simulation is stopped. Metrics can be stored in the models & metrics stores 106 for later retrieval and analysis.

The display performance metrics module 116 takes computed metrics from the performance metrics calculation module 112 or models & metrics stores 106 and displays them in a variety of appropriate formats. The display performance metrics module 116 operates in synergy with the simulation GUI to enable the user to select metrics to display or edit, scale or save/load/print graphs.

Referring now to figure 2 there is shown a diagram of run-time data structures 200 in accordance with embodiments of the present invention.
The run-time data structures are networks (connecting lines) of queues 201 and servers 202 (queuing networks) connected according to the architecture of figure 2, and comprise the following list of components: SimObjects; Workflows; Queues and Servers.
SimObjects are created and passed around the queue/server network. SimObjects are Stacks of Workflows. Each SimObject has a current workflow (the top workflow in the stack) and workflow step (initially the 1st step) maintained and updated for it.
Workflows are a list of steps containing service names and the workflow type, and optionally service demand times.
Queues 201 are ordered lists of SimObjects waiting to be resourced by an associated server. Queues are purely passive and are processed in First In First Out (FIFO) order.

Servers 202 are the active processing elements. They have input queues and output queues. Servers have three phases:

input actions (implemented by “inActions” functions), and output actions (implemented by the “outActions” functions). Initially all CPUs are free. During the input phase and if there is a free CPU, a server takes the next SimObject from an input queue based on a scheduling algorithm and then processes it. Once an SimObject is taken off an input queue it is delayed by the amount of time specified by the service demand time of the event during which time one CPU of the server is busy. The service demand time may possibly be scaled or modified by load dependent scaling (throughput, concurrency, CPUs, or utilisation dependent) or time quanta limits (e.g. maximum time quanta allocated for processing before being forced out and back into the input queue). Once the processing time has elapsed the SimObject is released from the server and sent to the workflow engine 203 input queue ready for the next step to be processed.
Some servers 203 (the workflow engine), are run-time artefacts and don’t correspond to model elements, while most have a direct correspondence (to composite services or H/W servers in SOPM).

The Workflow engine is implemented as a single server with one input queue, and with infinite CPUs and zero processing time. It works as follows. For each SimObject in the input queue it determines what the next current workflow and step(s) are (which depends on the workflow type), which queue(s) it/they should be sent to next, and increments the current workflow and step counter. Workflows that are completed are popped of the current SimObject stack, and workloads that are completed are then sent to the completed queue 204 (figure 2).

Referring now to figure 3, there is shown a schematic representation of the functionalities of a transformation engine 300 in accordance with embodiments of the present invention. The transformation engine operates by transforming SOPM language components (figure 6) to run-time data structures (figure 2) according to the following rules:

1. Workloads are transformed into a queue/server component 302 with the queue filled with one instance of the workload, and the server parameterised with one CPU.
2. Composite services/Workflows are transformed into (Queue network) servers 304 which have zero processing delay and infinite CPUs, so instantly create and push an instance of the workflow associated with the composite service onto the current SimObject stack and pass it on to the input queue of the workflow engine server.
3. Simple services are transformed into a queue 306 with the name of the service, which is then linked as an input queue to the (SOPM) server which the service is deployed to.
4. Servers are transformed into a (Queue network) server 308 with same name and number of CPUs. Note that a Server can have multiple input queues (simple services).

Referring now to figure 4 there is shown a schematic diagram representing the main components and steps of a discrete event simulation engine 400 in accordance with embodiments of the present invention. The discrete event simulation engine simulates the run-time data structures 200. The order of event processing is determined by a custom discrete event simulator which ensures that all events are processed in order, in other words, that all events occurring before or at the current simulation time are processed before any subsequent events, and that the system clock is advanced to the time of the next event to be processed. In embodiments of the invention, the custom discrete event simulation engine 400 processes the run-time data structures according to the following actions until the model is stopped or paused.

Input of run-time model data structures and initialisation of variables, current time, next time and model metrics 402, including list of servers (with inActions and outActions);
Execute all inActions for each server in the run-time model and update time 404;
Execute all outActions for each server in the run-time model and update time 406.
Find oldest server nextActionTime and update next action time and current time 408.

Referring now to figure 5, there are shown two flow charts 500, 550 of the functionalities of a server inAction and outAction methods in accordance with some embodiments of the invention.

Figure 5(a) is a flow chart representing the main steps of an inAction functionality of a server in accordance with embodiments of the present invention. The server checks if there are idle CPUs available 502; removes any waiting event from the respective input queue 504; computes the time delay for processing 506; sets the termination time 508 and pushes the event into a spare CPU slot 510.

Figure 5(b) is a flow chart representing the main steps of an outAction functionality of a server in accordance with embodiments of the present invention. The server checks if there are events in the CPUs slots 552; removes any event occurred at or after the current time indicator from the CPU slot 554; increments the next step 556, and sends them to the next input queue 558.

Figure 6 is a schematic representation of a SOPM Meta model 600 describing the main components types, relationships and parameters in accordance with embodiments of the present invention. A SOPM model consists of one or more of Workloads 602, Services 604 and Servers 606.

Each workload 602 has parameters of name, number of users, arrival period and arrival distribution. A workload represents a class of external or internal users of the system. A workload has a workflow represented by one or more steps 614. Each step is a call to a single named service 604. However the same service can be called by more than one step in a given workload. A step has parameters of call time (response time in ms), and call semantics which can include synchronous or asynchronous. A workload's workflow represents the business process used for the consumption of the modelled services. Workflows are not an explicit UML model component, but are modelled as an ordered list of step components and associated with either workloads or composite services.

2015101031 31 Jul 2015

Each service 604 has a name and can be simple 608 or composite 610. A simple service 608 represents an atomic service with no further dependencies on other services and no further implementation detail available or relevant. A simple service has a parameter of an optional default service demand time (response time, ms) representing the resources consumed on a server per service call. A service is deployed to a single server 606. There must be at least one simple service per model to be complete. A composite service 610 represents services than are both consumed and consume other services (including themselves recursively), and that have some internal workflow with steps calling other services associated with them. These workflows may have different semantics including sequential, parallel, or choice (probabilistic). Composite Service Workflows are modelled by one or more service steps 612. Service steps 612 are effectively one or more calls to services, called steps 614. Each step has a name of the service to be called and an optional "callTime" (which is the service demand for the called service and which is passed to the called service). Probabilistic steps have a probability which represents the probability of each step being invoked. Times passed into a composite service are passed to each step of the workflow directly. By default, steps (calls to services) are synchronous (request-response, wait for response), but may optionally be specified as asynchronous (call and forget, no response and no wait).

Each server 606 has a name, one or more simple services 608 deployed to it, and it may be associated with a number of concurrent processing units (CPUs). Servers 606 can be used to model physical server hardware and networks.

In some embodiments of the invention probabilistic workflows are used to model service time distributions. The models may be enhanced with array of time distributions in place of simple time parameters.

A SOPM model is correct and complete if it is a valid instance of a SOPM meta-model and if it has the minimum of information required in order to run on the simulation engine (at least one workload, one simple service, one server, and all required parameters set to valid values).

Figure 7 is a screenshot of an example implementation 700 of the Model GUI 102 (Figure 1), showing an example SOPM model instance and the main components and relationships, in accordance with embodiments of the present invention. The example SOPM in figure 7 is a high level view only and does not show all the details, in particular the parameter values are not shown. The example implementation 700 consists of two workload components, Workload 1 702 and Workload 2 704. Workload 1 702 represents an external application which consumes a single externally available service, Composite Service 1 706, which has a workflow with five steps which consume internal services (Simple Services1, 2 (called twice), 4, & 5).

Simple Service 1 708 represents a SOAP Web service, Simple Service 2 710 represents a Security service, Simple Service 3 712 represents pages on a web server, and Simple Services 4 and 5 714 represent application services. Simple Services 1-5 are deployed on representative servers (SOAP Server 722, Security Server 724, Web Server 726, Application Server 728).

2015101031 31 Jul 2015

Workload 2 704 represents an external user interacting with the system via web pages, and has a workflow with four steps. The first step interacts directly with Simple Service 3 712 (Web). The second step calls Composite Service 2 712. Composite Service 2 712 in turns calls Simple Service 2 710 (Security) and Simple Service 3 712 (Web). The third step calls Composite Service 3 718 which calls Simple Service 3 712 (Web) and Simple Service 4 714 (Application). Finally the fourth step calls Composite Service 4 720 which in turn calls Simple Service 2 710 (Security), Simple Service 3 712 (Web) and Simple Service 5 714 (Application).

The embodiments described in this section are exemplary embodiments of the invention and should not limit the scope of application of the systems and methods described herein. The systems and methods of the invention may be used, for example, to model client/server, n-tier, event-based, Enterprise Service Bus (ESB), architectures, business processes, and various h/w resources including servers, Virtualised servers, cloud hosting, databases, SAN, networks, etc, and systems involving humans as “resources”.

The models created by embodiments of the invention are composable and can be created from multiple other sub-models without modification. This is particularly important for service compositions where SOAs are realised by increasingly deep and complex layers of legacy services. These SOAs can be managed directly and be updated easily to reflect changes in the service compositions. In addition, models are paramaterisable from

2015101031 31 Jul 2015

real performance data obtained from monitoring infrastructure. Because models are represented using explicit SOA concepts, they can be automatically discovered and built from data sources including documentation and run-time monitoring data (e.g. transaction trace topology/response time views). Both batch mode and interactive simulations are also envisaged. Batch modes allow for changes to parameters and model structures to be specified in advance with a scripting language or directly in GUI to take effect at given times or event based which change aspects of the workload, service compositions, timing or server resourcing dynamically as the simulation runs. Interactive mode allows the model parameters to be interactively changed by a user with immediate effect on computed metrics.

Embodiments of the modelling system of the present invention are preferably carried out on a computer system 800 such as the one schematised in figure 8. The computer system 800 may be a high performance machine, such as a supercomputer, a desktop workstation or a personal computer, or may be a portable computer such as a laptop or a notebook or may be a distributed computing array or a computer cluster or a networked cluster of computers.

The computer system 800 also comprises a suitable operating system and appropriate software for implementation of embodiments of the present invention.

The computer system 800 comprises one or more data processing units (CPUs) 802; memory 804, which may include volatile or non volatile memory, such as various types of

2015101031 31 Jul 2015

RAM memories, magnetic discs, optical disks and solid state memories; a user interface 806, which may comprise a monitor, keyboard, mouse and/or touch-screen display; a network or other communication interface 808 for communicating with other computers as well as other devices; and one or more communication busses 810 for interconnecting the different parts of the system 800.

The computer system 800 may also be connected directly to remote systems and/or data analysis and visualisation equipment 812 to present modelling data. The remote systems 812 may include IT systems monitoring stations, maintenance centres or design centres.

The computer system 800 may also access data stored in a remote database 814 via network interface 808. Remote database 814 may be a distributed database.

A computer system for implementing embodiments of the invention is not limited to the computer system described in the preceding paragraphs. Any computer system architecture may be utilised, such as standalone computers, networked computers, dedicated computing devices, handheld devices or any device capable of receiving processing information in accordance with embodiments of the present invention. The architecture may comprise client/server architecture, or any other architecture. The software for implementing embodiments of the invention may be processed by "cloud" computing architecture.

It will be appreciated by persons skilled in the art that numerous variations and/or modifications may be made to the invention as shown in the specific embodiments without departing from the spirit or scope of the invention as broadly described. The present embodiments are, therefore, to be considered in all respects as illustrative and not restrictive.

i

The claims of the invention are as follows:

1. A system and methods for modelling the performance of information systems such as (but not limited to) service oriented architectures, the system comprising:

a model graphical interface;

a simulation graphical interface;

a module for managing service oriented performance models;

a transformation module for transforming service oriented performance models into run-time data structures;

a simulation module consisting of a discrete event simulation engine;

a metrics calculation module for calculating metrics associated with run-time data structures; and

a metrics display module.

2. A graphical modelling tool comprising means for creating, editing and managing performance models of a plurality of components of service oriented computer architectures, wherein the plurality of different components comprises: workloads, simple and composite services, workflows and servers, and wherein each component is associated with a plurality of measured or experimental performance parameters.

3. A transformation engine to transform service oriented performance models to run-time data structures comprising a plurality of transformation rules to transform each component of the service oriented performance model into a run-time component capable of being processed by the simulation engine of claim 4.

4. A discrete event simulator to process discrete events in a sequence comprising a discrete event simulation engine which processes run-time data structures according

to claims 2 to 3, and which responds to changes of parameter vales in real-time.

5. A metrics system comprising a module which creates and computes values for metrics for components of a run-time performance model, wherein the metrics comprise: arrival rate, throughput, response time, wait time, service demand, concurrency, server Utilisation and server BusyCPUs; and a graphics module which displays metrics in real-time.

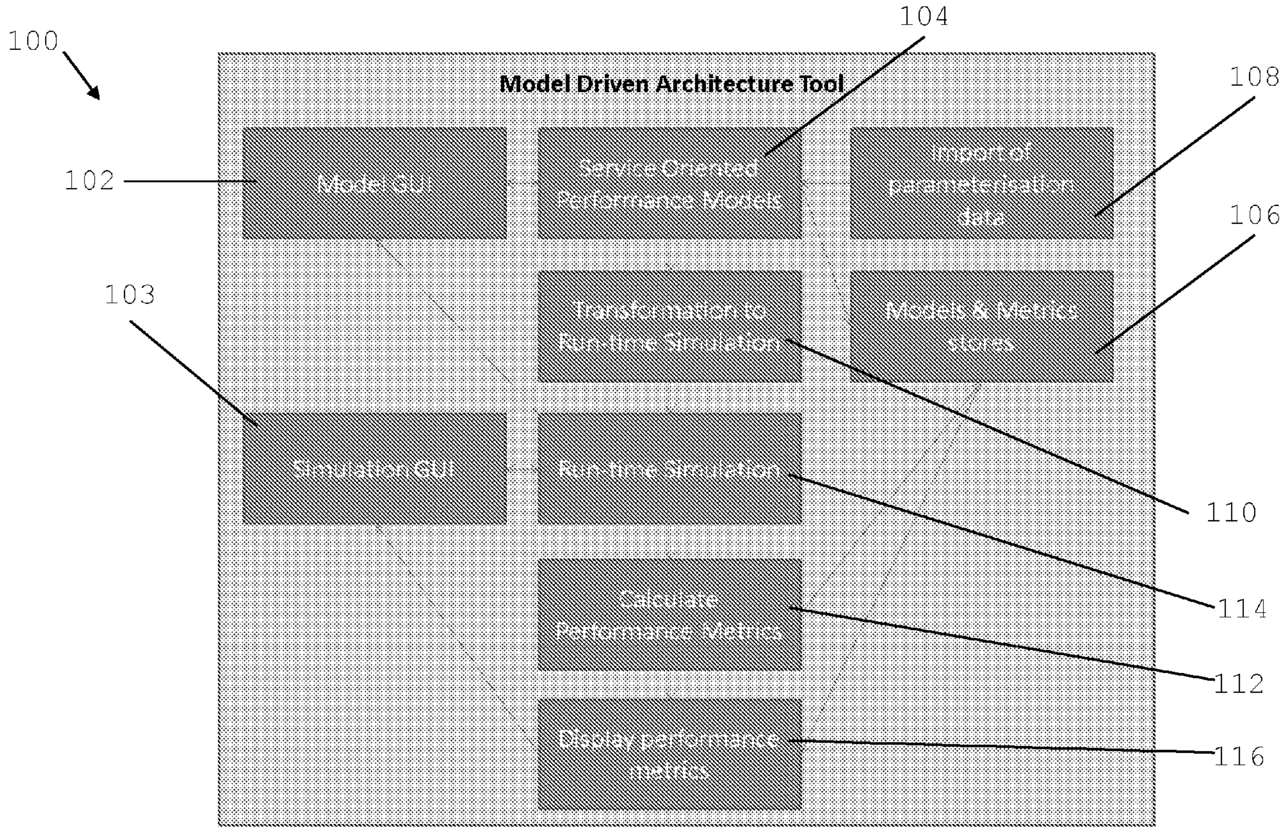

100
Model Driven Architecture Tool
102
Model GUI
104
Service Oriented Performance Models
108
Import of parameterisation data
106
Models & Metrics stores
103
Transformation to Run-time Simulation
Simulation GUI
Run-time Simulation
110
114
Calculate Performance Metrics
112
Display performance metrics
116

FIGURE 1

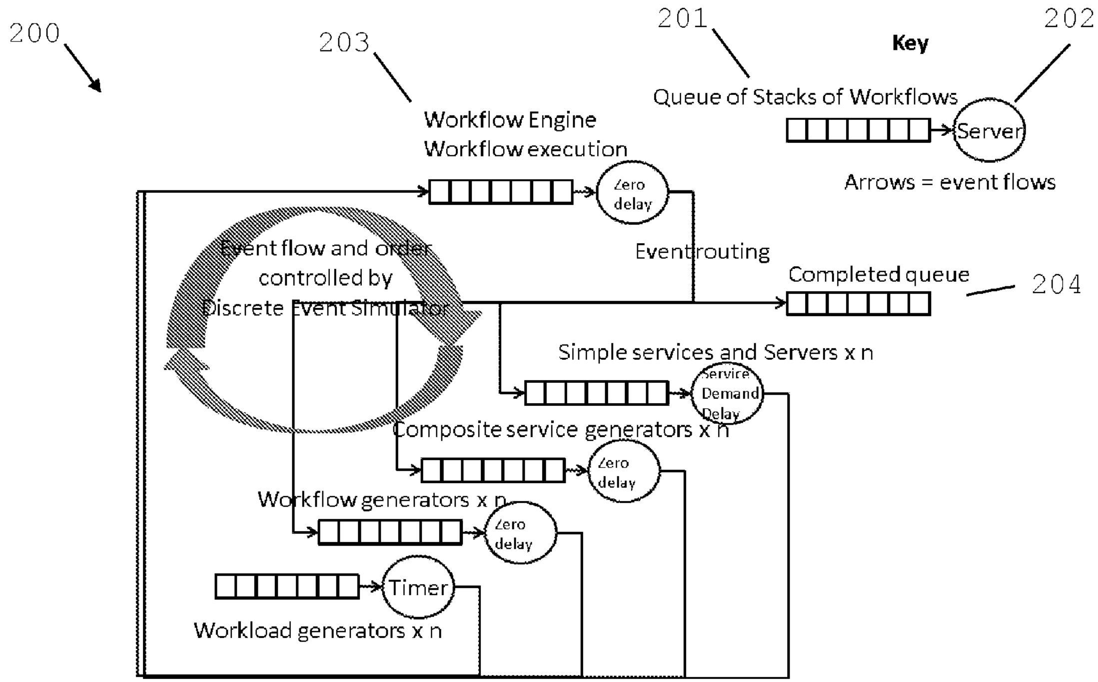

200
203
Workflow Engine
Workflow execution
Zero delay
201
Key
202
Queue of Stacks of Workflows
Server
Arrows = event flows
Event routing
Completed queue
204
Event flow and order controlled by Discrete Event Simulator
Simple services and Servers x n
Service Demand Delay
Composite service generators x n
Zero delay
Workflow generators x n
Zero delay
Timer
Workload generators x n

FIGURE 2

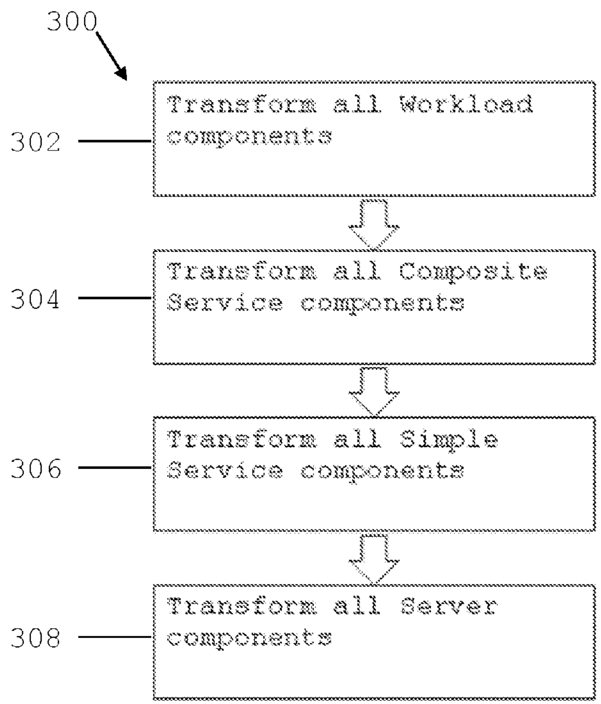

300
302
Transform all Workload components
304
Transform all Composite Service components
306
Transform all Simple Service components
308
Transform all Server components

FIGURE 3

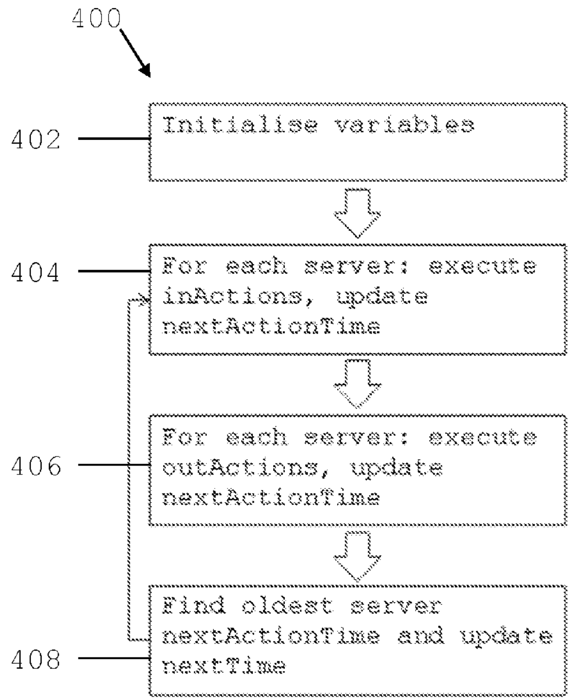

400
402
Initialise variables
404
For each server: execute inActions, update nextActionTime
406
For each server: execute outActions, update nextActionTime
408
Find oldest server nextActionTime and update nextTime

FIGURE 4

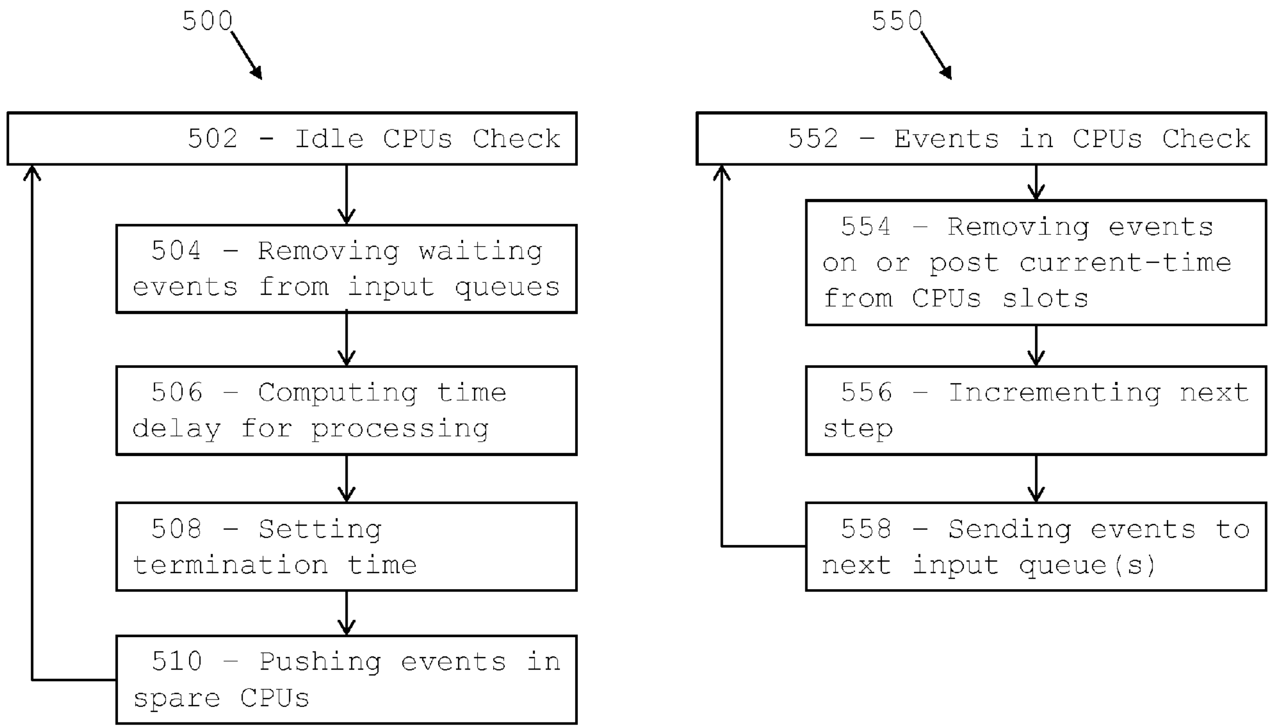
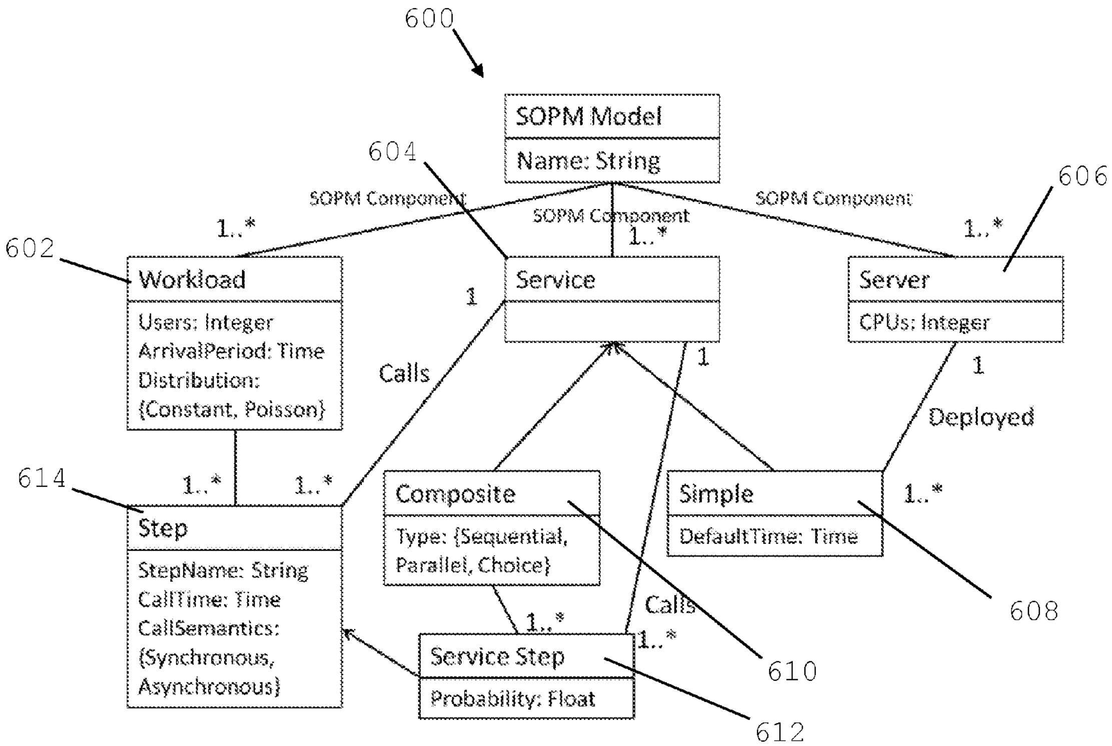

500
502 - Idle CPUs Check
504 - Removing waiting events from input queues
506 - Computing time delay for processing
508 - Setting termination time
510 - Pushing events in spare CPUs
550
552 - Events in CPUs Check
554 - Removing events on or post current-time from CPUs slots
556 - Incrementing next step
558 - Sending events to next input queue(s)

FIGURE 5



600
SOPM Model
Name: String
604
SOPM Component
SOPM Component
SOPM Component
606
1..*
1..*
1..*
602
Workload
Users: Integer
ArrivalPeriod: Time
Distribution:
{Constant, Poisson}
1
Service
Server
CPUs: Integer
Calls
1
1
Deployed
614
1..*
1..*
Step
StepName: String
CallTime: Time
CallSemantics:
{Synchronous,
Asynchronous}
Composite
Type: {Sequential,
Parallel, Choice}
Simple
DefaultTime: Time
1..*
608
1..*
Calls
1..*
Service Step
Probability: Float
610
612

FIGURE 6

GRAPHIC USER INTERFACE FIGURE

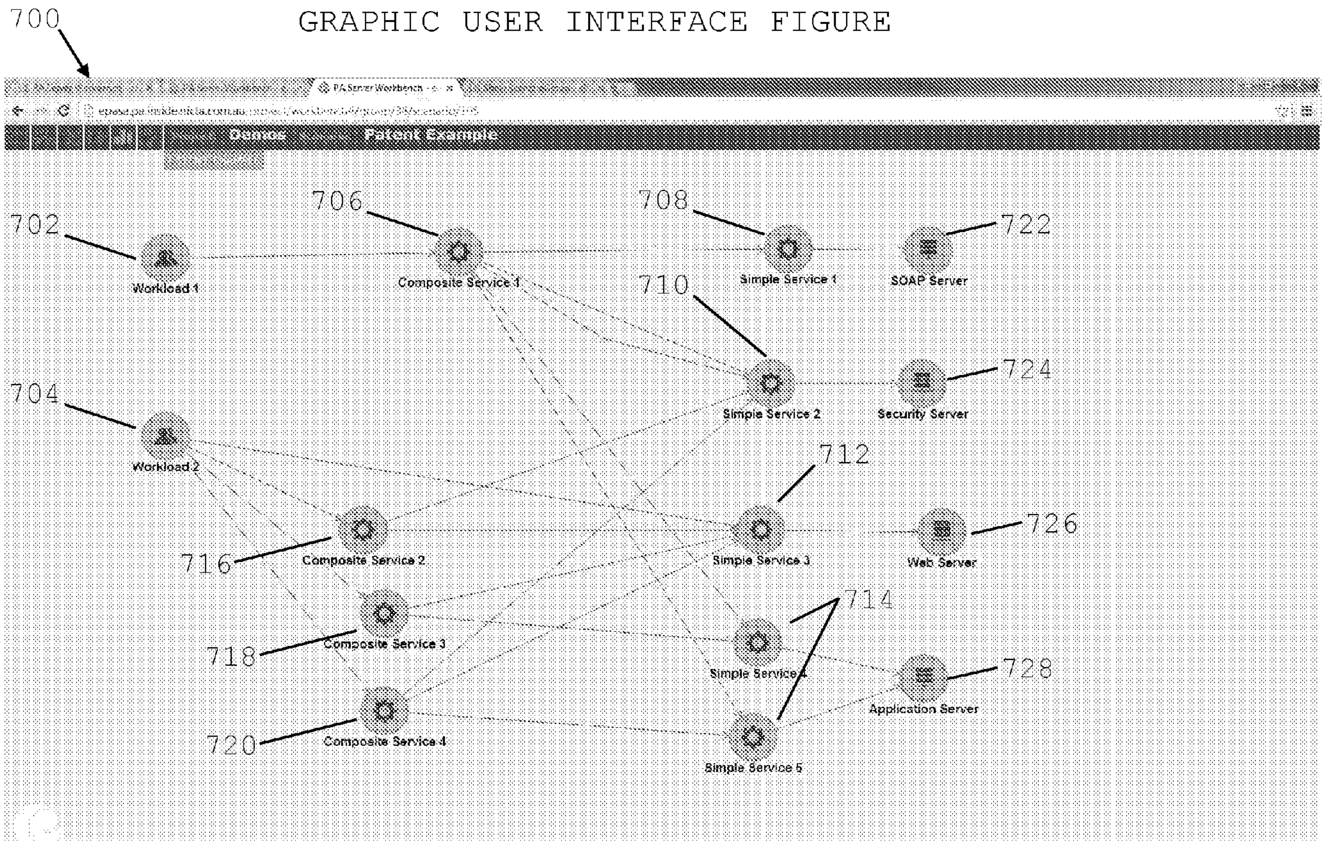


FIGURE 7

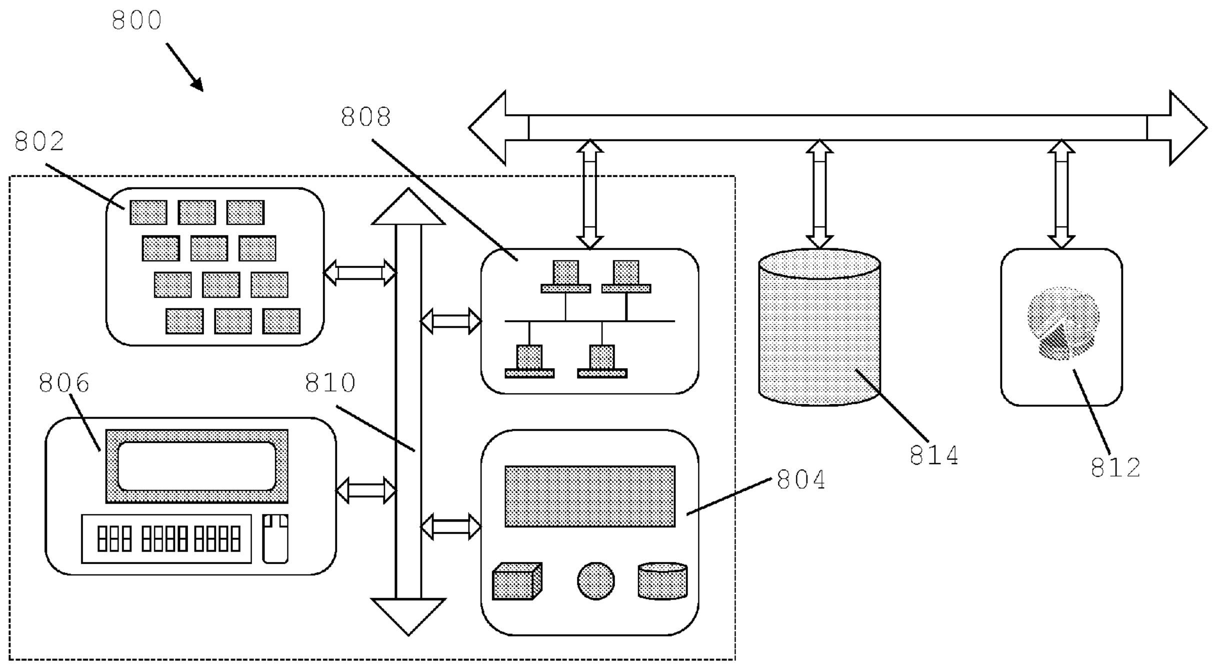


FIGURE 8