Adaptive Infrastructure meets Adaptive Applications

Nigel Edwards, Guillaume Belrose, Klaus Brand, Sven Graupner,
Jerry Rolia, Lawrence Wilcock
Enterprise Systems and Software Laboratory
HP Laboratories Bristol
HPL-2007-138
August 21, 2007*

enterprise applications, business process modelling, SAP, model driven IT management, adaptive infrastructure, virtualization, UML, CIM

Enterprise class applications such as SAP consist of complex configurations of many interacting components. It requires considerable skill to match the design of an application's component configuration to an appropriate infrastructure configuration design. So even if we have an Adaptive Infrastructure it is hard to adapt such systems because of the difficulty in matching the application and infrastructure configurations. This paper proposes a model that formally links the application and infrastructure configurations. The model takes into account business process, applications, and infrastructure. It allows us to link changes in the business process to necessary and MATCHED changes to applications and infrastructure.

* Internal Accession Date Only

Adaptive Infrastructure meets Adaptive Applications

Nigel Edwards, Guillaume Belrose, Klaus Brand, Sven Graupner, Jerry Rolia, Lawrence Wilcock

HP TSG & HP Labs
{nigel.edwards, guillaume.belrose, klaus.brand, sven.graupner, jerry.rolia, lawrence.wilcock}@hp.com

Abstract. Enterprise class applications such as SAP consist of complex configurations of many interacting components. It requires considerable skill to match the design of an application's component configuration to an appropriate infrastructure configuration design. So even if we have an Adaptive Infrastructure it is hard to adapt such systems because of the difficulty in matching the application and infrastructure configurations. This paper proposes a model that formally links the application and infrastructure configurations. The model takes into account business process, applications, and infrastructure. It allows us to link changes in the business process to necessary and MATCHED changes to applications and infrastructure.

Problem Statement

The purpose of the applications and supporting infrastructure in an enterprise system is to deliver business processes to the enterprise: supply chain management, customer relationship management, etc. For many enterprises the business process and innovations therein are their competitive advantage. Gaining and maintaining this competitive advantage requires the ability to be able to make changes to the business process. Changing the business process requires making changes in the supporting applications and infrastructure to implement this change.

An adaptive infrastructure on its own is insufficient to build truly adaptive enterprise systems. The problem is how to manage consistent changes across all three viewpoints: business processes, applications and infrastructure. Enterprise applications consist of complex configurations of many interacting components. These include application servers, virtual machines, appliances (i.e. firewalls), networks, and storage. It is a complex, difficult task to design the configuration and deploy all these components to deliver the functionality of the desired business process.

Delivering appropriate non-functional requirements such as performance, reliability and security increases the complexity and difficulty even further. The resource demands of the components are dependent on the workload mix: functions invoked, size and type of data. Scaling the application to meet a given performance requirement is not simply a question of arbitrary replication (scale-out) or using larger machines (scale-up). Rather it requires great skill to achieve specific non-functional requirements, by matching the application configuration and infrastructure configurations. This means that design and performance sizing errors are common.

Once a system has been implemented, it can be extremely difficult to adapt it. Skilled and expensive consultants are required for the redesign: change is slow and expensive. This problem will become worse as companies such as Oracle [8] and SAP [11] adopt fine-grained Service Oriented Architectures (SOA): the number of interacting components will increase significantly. The inability to adapt means that systems have to be sized for their maximum use, resulting in wasted resources for much of the time. It also means that systems cannot readily take advantage of additional resources even if they are available.

Our Solution

Working with colleagues at SAP-Research we have developed an integrated model to link the design, configuration and management of: business processes, applications and infrastructure. The flow of information and components in the model are show in Fig. 1. For simplicity, we do not show the cycles and loops that would occur in this diagram as a system progresses through its lifecycle and evolves through change and maintenance activities.

The model is stored in a model repository so that different tools can access and manipulate it, managing the system through its lifecycle. Using this tool-chain, we begin with a definition of the business process in terms of functional steps (General Process Model). We specify the time-varying non-functional requirements for these steps: the number of concurrent users, response time, availability and security (Custom Process Model). Next we identify and add to the model the software components needed to deliver the functional steps (Unbound Model). We then proceed with the software configuration and infrastructure design – this is captured in the Grounded Model. Finally, the model is used to drive infrastructure configuration services, application installation services, application configuration services, and the services that manage the startup and ongoing running of the system (Bound and Deployed Models).

A crucial benefit of the model is that it provides a framework within which we can reason about and manage change. So changes in the requirements of the system, resulting from changes in business process (driven by business-need) lead to matched changes in the application and infrastructure configurations. The model is used by tools managing a shared-infrastructure, in which services for many different customers are run on a shared physical infrastructure. It is used by tools for managing a changing set of business processes and changing physical resource allocations.

The remainder of this section describes the information flow and components in the model shown in Fig. 1 in more detail. We have used UML/CIM (Unified Modeling Language/ Common Information Model) [12], [4] as our conceptual modeling language; the current implementation uses EMF (Eclipse Modeling Framework) [5] because of its integration with Java and its support for persistence in XML.

General Process Model

The General Process Model is a specification of the functional steps that make up the business process. The General Process Model is chosen from a catalog of existing, supported business processes. SAP and partners have a huge and growing set of such business processes often referred to as scenarios. The business processes in the catalog would typically be described in BPEL [3] or Event Process Chains [1]. The General Process Model also identifies the (SAP) application components that typically implement each step.

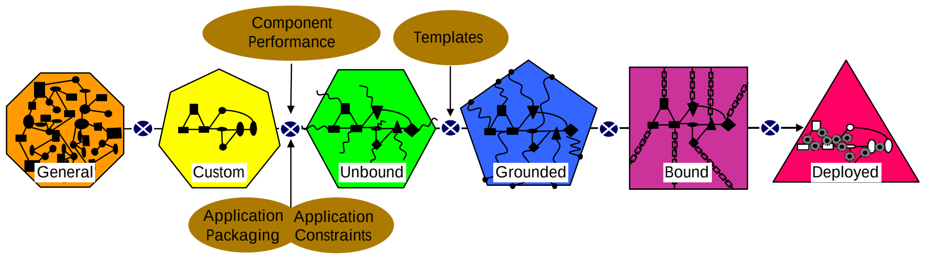


Fig. 1. The Model Information Flow and components

Customized Process Model

The Customized Process Model is a customization of the General Process Model. The designer may have added or removed steps or introduced new loops and sequences. Most importantly time-varying non-functional requirements are introduced. Fig. 2 shows an example of a customized process model for the Sales and Distribution (SD) benchmark. SD is a widely used SAP product. The SD benchmark is a subset of SD functionalities used to assess the performance of IT systems when running SAP; it consists of a well known business process. Only time-varying performance non-functional requirements are shown for brevity (Fig. 1: left side).

If we look at the Create Sales Order BPStep (Fig. 2: top, middle) we see that it is implemented by the SAP application component VA01 (Fig. 2: top right). This step consists of navigating through several screens and entering the order. The initial screen for VA01 is shown in Fig. 3. The expected detail of this workload mix, including the amount of data being entered, is specified in the BPStepToApplicationComponentMapping relation. Notice that the Business Process Steps can be composed. In Fig. 2 a single SD_Benchmark Business Process (top left hand side) comprises of five children: Create Sales Order … Create Delivery Document. Concurrency is implicit: many users can execute many steps concurrently.

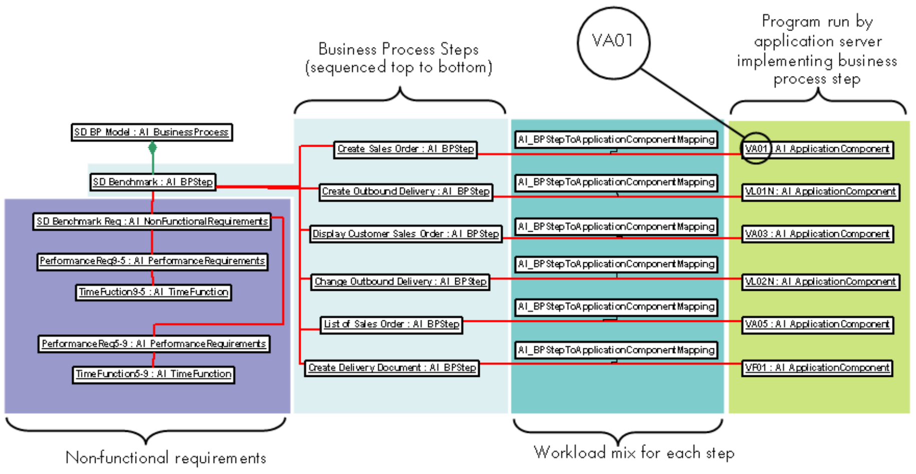
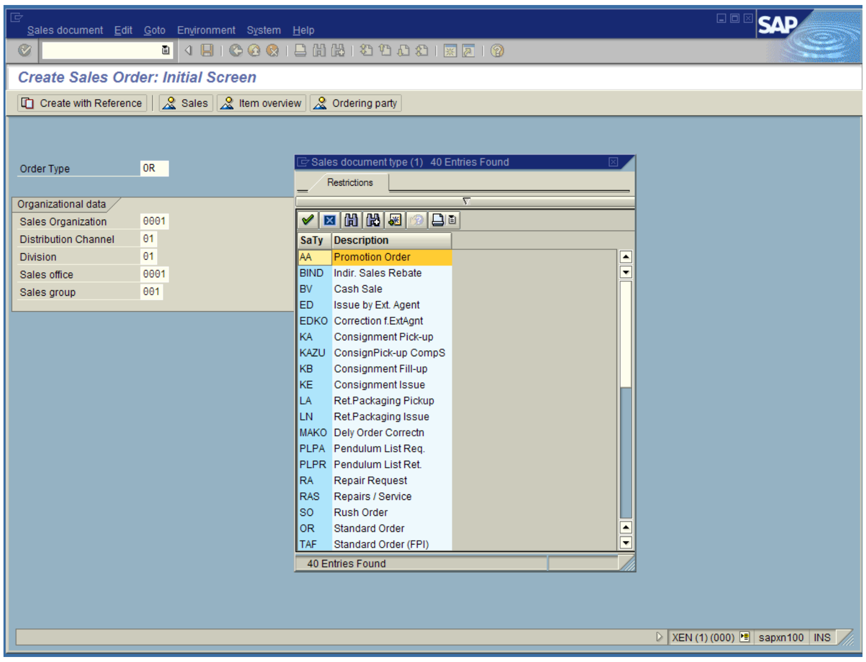

Business Process Steps
(sequenced top to bottom)
VA01
Program run by application server implementing business process step
SD BP Model : AI BusinessProcess
SD Benchmark : AI BPStep
SD Benchmark Req : AI NonFunctionalRequirements
PerformanceReq9-5 : AI PerformanceRequirements
TimeFuction9-5 : AI TimeFunction
PerformanceReq5-9 : AI PerformanceRequirements
TimeFunction5-9 : AI TimeFunction
Create Sales Order : AI BPStep
Create Outbound Delivery : AI BPStep
Display Customer Sales Order : AI BPStep
Change Outbound Delivery : AI BPStep
List of Sales Order : AI BPStep
Create Delivery Document : AI BPStep
AI_BPStepToApplicationComponentMapping
VA01 : AI ApplicationComponent
VL01N : AI ApplicationComponent
VA03 : AI ApplicationComponent
VL02N : AI ApplicationComponent
VA05 : AI ApplicationComponent
VF01 : AI ApplicationComponent
Non-functional requirements
Workload mix for each step

Fig. 2. Custom Model for Sales and Distribution Benchmark



Create Sales Order: Initial Screen

Order Type	OR

Organizational data

Sales Organization	0001
Distribution Channel	01
Division	01
Sales office	0001
Sales group	001

Sales document type (1) 40 Entries Found

SaTy	Description
AA	Promotion Order
BIND	Indir. Sales Rebate
BV	Cash Sale
ED	Issue by Ext. Agent
EDKO	Correction f.ExtAgnt
KA	Consignment Pick-up
KAZU	ConsignPick-up CompS
KB	Consignment Fill-up
KE	Consignment Issue
LA	Ret.Packaging Pickup
LN	Ret.Packaging Issue
MAKO	Dely Order Correctn
PLPA	Pendulum List Req.
PLPR	Pendulum List Ret.
RA	Repair Request
RAS	Repairs / Service
SO	Rush Order
OR	Standard Order
TAF	Standard Order (FPI)

40 Entries Found

XEN (1) (000) sapxn100 INS

Fig. 3. Initial screen for Create Sales Order Business Process Step

An example of a tool that can handle General and Customized Process Models is Aris from IDS-Scheer [7]. Integrating Aris with our Model Repository would require an adaptor to transform between the representations supported by Aris and that used by the Model Repository.

Unbound Model

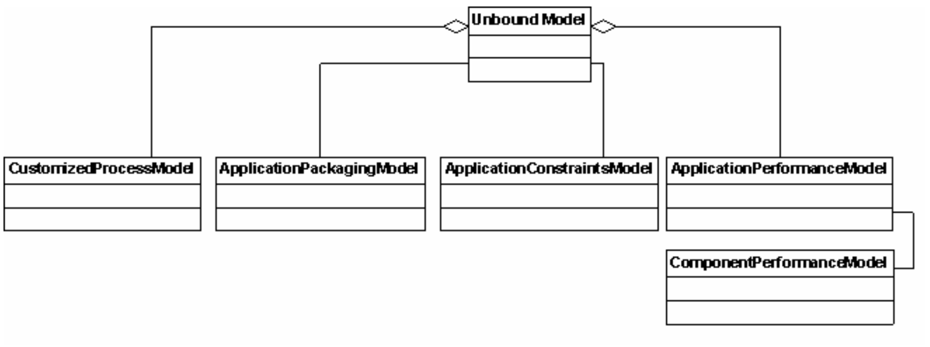

Unbound Model
CustomizedProcessModel
ApplicationPackagingModel
ApplicationConstraintsModel
ApplicationPerformanceModel
ComponentPerformanceModel

Fig. 4. Unbound Model structure

The Unbound Model is a complete functional specification of the system, but is not bound to any particular infrastructure design. As shown in Fig. 4, the Unbound Model is composed of several sub models. It includes the Custom Process Model, Application Packaging Model, Application Constraints Model and the Application Performance Model (calculated from the Component Performance Model). The Custom Process Model and Application Performance Models are specific for each Unbound Model. The Application Packaging Model, Application Constraints Model and Component Performance Models are libraries that are reused in many different Unbound Models.

In the Unbound Model all the application components are mapped to products and the necessary Execution Services that execute them (databases and application servers in the case of SAP). The Application Packaging and Application Constraints models tell us how to do this. The Application Packaging Model tells us what products and what components within those products need to be installed and configured to implement the required business process. The Application Constraints Model gives information on how different components can be packed together or consolidated. For example in the case of SAP, there are rules and best-practices that should be followed concerning how to configure multiple application server components in a single server.

The Application Packaging and Application Constraints model libraries need to be built either by the application vendor, in our case SAP, or a domain expert such as a consultant specializing in the application.

The Unbound Model also specifies the performance requirements of each service that needs to execute. The information to derive these performance requirements comes from the Component Performance Model and workload mix for each component specified in the BPStepToApplicationComponentMapping relations shown in Fig. 3. The Component Performance Model is combined with the BPStepToApplicationComponentMapping relations to calculate an Application Performance Model which is SPECIFIC to the Customized Process Model.

At the time of writing we are investigating using HP's LoadRunner [13] product to generate Component Performance Model libraries automatically. LoadRunner enables us to drive an individual component or set of components with a specified workload mix and accurately record the infrastructure demands of the component or components under that workload mix. Infrastructure demands include CPU, memory, network and storage across application server tiers.

Grounded Model

The Grounded Model is "grounded" in a particular infrastructure design. Each service that is required to deliver the business process is assigned a particular type and configuration of computer. A number of possible designs are proposed and we build a Layered Queuing Model (LQM) [9], [6] of each of these designs. An LQM consists of a queuing network for each element in the design: clients, business process steps, application components, execution services and hardware. Synchronization points are specified between each layer (e.g. a client invoking VA01). We then solve the LQM with the demands in the Unbound Model against the performance capabilities of the infrastructure to determine if the proposed design can meet the performance requirements. Each design that is evaluated is chosen from the Templates Model shown in Fig. 1.

The Templates Model consists of a catalog of templates. Each template specifies an infrastructure that can be instantiated and also how to install, configure and start applications on that infrastructure. The templates provide a number of degrees of freedom. In the case of SAP:

- The number of application server machines
- The size of server machines
- The number of components (e.g. worker processes) – typically this is a function of the size or capacity of the server

Choosing a design requires selecting a template from the template model and filling in parameters such as the number and type of physical server. From this we build a Layered Queuing Model and evaluate the performance of the design.

One crucial simplification that we made in our modeling was not to instantiate every possible configuration; rather the Templates Model catalog presents a fixed selection of physical and virtual machine configurations, for example, small, medium and large virtual machines. An example of a computer system configuration is Xen/SuSe Linux Enterprise VM, 3GB RAM, 40GB OS disk, 2 additional 40GB disks, performance 500 SAPs (this is a performance measure used by SAP). This substantially simplifies the model, as we do not have to explicitly model virtualization avoiding recursion problems such as a virtual machine on a virtual partition on a hard partition on a physical machine. Instead we provide a computer (physical or virtual) with a known set of characteristics; the virtualization-host relationship is not modeled. This also substantially simplifies the management code needed to instantiate the infrastructure.

Templates also simplify the implementation of the tools that process the models. Transforming from the Unbound to the Grounded Model requires selecting a template, filling in parameters and evaluating the predicted performance of the template against the required performance specified in the Unbound Model. The main limitation of this approach is that it restricts the application configurations that we can support. As we gain more experience we plan to introduce more and more degrees of freedom and build more sophisticated tools for transforming from the Unbound to Grounded Models.

There may be several Grounded Models for one Unbound Model, because the Unbound Model contains time-varying performance requirements, as shown in Fig. 2. This is used to manage the change in system configuration, i.e., MATCHED changes to application and infrastructure configurations, according to predicted and specified time-varying business demands. Each Grounded Model has a fixed set of physical resource associated with it. Any load can be accommodated up to the maximum specified in the Grounded Model. If the load (e.g. number of users) exceeds that specified in the Grounded Model, the performance requirements may not be meet.

Bound Model

The Bound Model binds the Grounded Model to physical resources in the data centre. These resources are no longer available for use by other services. Finding the most appropriate resources in a shared data centre fabric is non-trivial. We use a trace-based capacity planning service [10] to determine where best to place the new load. A Resource Directory and Resource Acquisition service are also needed. Sometimes we need to move existing loads, using live virtual machine migration, to make room for new loads. The model presented here provides us with a framework for managing this.

Deployed Model

The Deployed Model models the state of the running system. The Deployed Model is obtained by passing the Bound Model to our deployment services: infrastructure, software installation etc. The information carried forward from the Application Packaging, Application Constraints and Template Models enables us to install, configure and manage the execution of each software component. We instantiate management services to monitor adherence to performance requirements and provide traces of monitored workload data to the capacity planning service.

Using the model to manage change

The purpose of the model is to provide the information needed to manage a system through its lifecycle. Some changes to a system can be fully automated; others cannot and should not be fully automated. Examples of changes that can be fully automated include migration of virtual machines, changing the amount of physical resource available to a virtual machine (increasing or decreasing the amount of available RAM), adding an additional (virtual) machine to run an additional application server to increase the throughput of the system. For these classes of changes we are building services that read the model from the model repository, compute a delta operation and add it to the model repository. Other services then process the delta in the model to apply the change to the running system. These classes of well known changes and the context within which they can be applied are encoded in the Templates.

Examples of changes that would not normally be fully automated are changes to the business process, patches and system upgrades. For such changes it is appropriate to follow ITIL/ITSM processes [2]. This implies human activity such as a Change Review Board and programming. For a change in the business process, there would generally be a period of implementation and testing before the change is transported into the live system. The transport would very likely use application specific tools such as SAP Solution Manager. The model presented here formalizes the information needed to create the appropriate test and development system to implement the change. We start by extracting the model of the system from the model repository and creating a replica of the production system for test and development.

What degree of automation is possible?

The current state of the art for automation is not uniform as we progress from left to right through the model information flow. Over the last few years there have been several demonstrations of automatic infrastructure deployment, including the HP Utility Data Center. We are not aware of any attempts to automate the ongoing dynamic change of infrastructure and application configuration once the service has been deployed. Nor are we aware of any attempt to automate from such a comprehensive set of models the selection of the most appropriate application configuration and infrastructure topology before making a change or doing the initial deployment. The current focus our work is the middle to right hand side of Fig. 1.

One dimension of automation enabled by business process design (left hand side of Fig. 1) is the automated creation and generation of test cases as well as the test environments for functional and performance testing. Testing accounts for a substantial part of the application roll-out. HP LoadRunner, for instance, can generate specific load patterns and exercise specific transactions of a business process through scripts. If test configurations are part of the application design, the deployment of the test environment can be automated, which include workload and test generators. Both help to accelerate the solution deployment time.

Currently, automation of business process design (left hand side of Fig. 1) is a relatively new field in which SAP and others are investing heavily. The objective is to generate code from user designed business process models. Typically the user constructs a graph to represent the design of their business process. The tool generates code from these graphs. Our objective is to work with SAP and others so that these tools are able to construct models that contain the necessary information to install, configure, test and manage the applications on a shared infrastructure. The models need to support a range of non-functional requirements: performance, security and availability.

In the absence of full automation of business process design, we intend to construct a catalog of business processes along with their associated models including test cases. A business process is selected from the catalog and appropriate non functional requirements are specified. This is the unbound model. We then go through the automated process of generating the other models as described above.

Evidence that the Solution Works and current status

We have applied the model described above to the SAP SD application. We have built a model for SD, and are able to automatically create virtualized infrastructures (using Xen) from a Grounded Model and automatically install, configure and start SD within that virtualized infrastructure. We are able to manage the virtualized infrastructure according to performance constraints, using live migration of virtual machines to adjust to time-varying demands. We have built a Layered Queuing Model of SD to allow us to evaluate design choices in our templates. We have built an implementation of the Model Repository and a number of tools that use it including: Resource Acquisition, Infrastructure Deployment and Software Deployment Services.

Competitive Approaches

The idea of using models to link tool sets together is not new. It is being used extensively in some HP's initiatives. It is also implicit in the work on CIM within the DMTF which tries to provide a common model so that different tools can use the same models. However, we believe what is new is an integrated model comprising the set of models proposed here that go from an abstract specification of the business process; onto a specification of the software components to deliver that business process; onto a specification of the infrastructure to meet the requirements; onto a fully usable and deployed service; onto managing changes to the system.

Next Steps

We need to find or develop additional tools that automate some of the transitions described here. For example, we are collaborating with a group in HP Labs who have previously developed technology that automates the configuration of high-end Superdome servers using design constraints and a constraints satisfaction solver. We are trying to determine if we can use similar ideas for automating the design of Grounded Model from the Unbound Model using the information in the Application Packaging and Application Constraints Model. This will allow us to introduce more degrees of freedom into the Templates Model described above.

HP Labs, SAP Research and others are working on technology to automatically derive some of the necessary models, in particular the Component Performance Models. We are using LoadRunner for this; another technique involves source code analysis. Working with our colleagues at SAP Research we are designing layered queuing models for SAP systems, starting with SD.

Much of HP's work to date has been focused on transforming from the Unbound Model through the Grounded, Bound and Deployed Models. We are now increasing our efforts on business process modeling and associated tools. Finally although this work is a collaboration with SAP Research, the HP team's intention is to build a generic model that can be applied to other enterprise software stacks.

Acknowledgements

We are grateful to Daniel Gmach, Ilango Sriram, Johannes Kirschnick, Maher Rahmouni and Paul Vickers with whom it has been our pleasure to work on this project. We are grateful to John Manley for his constant support and encouragement.

References

1. Formalization and Verification of Event-driven Process Chains, W.M.P. van der Aalst ,Information and Software Technology, 41(10):639-650, 1999.
2. “Foundations of IT Service Management based on ITIL”, Jan van Bon, Van Haren Publishing, September 2005
3. Business Process Execution Language for Web Services, Oasis, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel
4. Common Information Model, Distributed Management Task Force, http://www.dmtf.org/standards/cim/
5. The Eclipse Modeling Framework, http://www.eclipse.org/emf/
6. “Performance validation tools for software/hardware systems”, U. Herzog, J. Rolia, Perform. Eval. 45(2-3): 125-146 (2001).
7. The Aris Platform, http://www.ids-scheer.com/
8. "Oracle Fusion Middleware", http:// www.Oracle.com/Middleware
9. “The Method of Layers”, J.A. Rolia and K.C. Sevcik, IEEE Transactions on Software Engineering, Vol. 21, No. 8, pp. 689-700, August 1995.
10. “A Capacity Management Service for Resource Pools”, Jerry Rolia, Ludmila Cherkasova, Martin Arlitt, Artur Andrzejak, Internet Systems and Storage Laboratory HP Laboratories Palo Alto HPL-2005-1.
11. “Enterprise Service-Oriented Architecture (Enterprise SOA)”, http://www.sap.com/solutions/esa
12. The Unified Modeling Language, http://www.uml.org/
13. HP LoadRunner, http://www.mercury.com/us/products/performance-center/loadrunner/