# System Virtualization Tools to the Rescue of Software Developers

Juan C. Dueñas, José L. Ruiz, Félix Cuadrado, Boni García, and Hugo A. Parada G.
ETSI Telecomunicación, Universidad Politécnica de Madrid
Madrid, España
+34 91 336 73 66 ext 3034
{jcduenas, jlruiz, fcuadrado, bgarcia, hparada}@dit.upm.es

## ABSTRACT
An always-on runtime environment for launching and testing applications is a very convenient asset for modern enterprise software development. Restricted access and configuration complexity of pre-production sites are show-stoppers for the software development life-cycle. This article presents a solution for offering a personal runtime to developers by means of virtualization tools. Virtualization is a very cost-effective way to provide a test environment similar to the production site. We have followed an MDA approach in order to integrate best-of-breed virtualization technologies such as Xen and VDE into Eclipse. IDE integration was a natural approach to ease adoption of these new technologies.

## Keywords
Virtualization, Development, Distributed Systems, MDA.

## 1. INTRODUCTION
The increasing complexity of IT ecosystems has exacerbated the need for specialized roles in modern organizations [1]. Specialists are required in order to efficiently and cost-effectively cope with the multiple aspects involved in the production life-cycle. In most organizations, deployment is controlled by system administrators, while previous tasks are carried out by the software development team. Please note that we use the term development team loosely, there is a great many roles involved in the development process such as requirements engineers, software architects, quality assurance staff, and testing engineers.

The organizational gap between the development team and the administrators is a show-stopper for the software development process. Deploying code over the production site is out of question, because it can have a negative impact on the normal operation of business services. Furthermore, information related to customers is usually protected by law. Thus, data access must be especially handled and is not available for development purposes.

Operation and maintenance of the production sites are the key responsibilities of system administrators. High availability and performance of services are their main concerns. In consequence, they do not have either resources or the technical skills to aid the development team at testing and deploying immature software components.

Developers' productivity and, as a result, time to market, are negatively affected by this role mismatch. The situation is even worse when agile software development models are applied. Agile processes make an extensive use of rapid iterations over the basic waterfall life-cycle. Frequent test execution is the cornerstone of these methods. Therefore, the need for a suitable deployment environment is even more pressing.

In many organizations, a pre-production site is set up to cope with this situation. Unfortunately, this is usually a problem for both teams; administrators and developers. On the one hand administrators have to install, configure and maintain a new set of servers. And on the other hand, developers without production servers expertise have difficulties to manipulate and configure them for their tests. In addition, concurrent manipulation of pre-production servers is prone to errors. This dilemma is too often solved by establishing access control mechanisms to the pre-production site, under the supervision of the administrators. Software deployment to a pre-production site comprises activities such as managing databases and application servers, loading test data, reconfiguring application components, running scripts and issuing formal requests to coordinate the work between teams.

All in all, the problem is by no means solved but duplicated. From the software development point of view the ideal situation would be that each developer enjoyed a fully independent, always on, runtime environment to launch and test her applications. With real servers and networks this quickly leads to machine sprawl. However, effective testing requires that the system under test runs on an environment as close as possible to the production site. Fortunately, virtualization tools come to

the rescue. Virtualization is nowadays a key technology enabling the utility computing paradigm [2]. In this way, physical server resources can be partitioned to match the specific needs of each consumer. It is an excellent option of embedding a pre-production site into each developer's machine [3] and reducing IT costs.

Virtualization tools have the potential to put at the developers' hand a runtime environment for testing distributed applications, in similar conditions to what they will find at the production site. However, installation and configuration of a complete virtualized and distributed runtime involves a big effort and high-skilled staff, because it requires understanding and handling a wide range of technologies: operating systems, databases, application servers, network services and so on. Tool support is necessary for integrating virtualization technologies into productive software development processes, since developers should be as oblivious of the underlying complexity as possible. Developers feel at home with their IDEs so we decided that extending the IDE was the natural approach in order to ease adoption, flatten the learning curve and thus improve productivity. Development orientation is the key difference with other virtualization solutions widely used in IT operational settings such as VMWare Virtual Center. Our goal is to provide a developer-friendly environment. In short, it is as simple as opening your favorite IDE, pushing a button to launch the virtual environment and then it is again a matter of deploying and testing software. We have built our first proof of concept on Eclipse, following an MDA (Model Driven Architecture) approach. First, we created a Platform Independent Model (PIM) for virtualized distributed systems. PIM instances are created using a visual editor and are eventually translated at launch time to specific models and technologies. This way, details of the underlying virtualization technologies are hidden from the users. The tool supports a wide range of virtualization technologies that play the role of Platform Specific Models (PSM); an overview is included in Section 2. Section 3 presents the tool design, which is based on a core component that includes the PIM and provides extension points for adding new technologies and editors. The article ends with some concluding remarks and future work.

## 2. ON VIRTUALIZATION TECHNOLOGIES

The purest approach to virtualization is full system virtualization, also known as emulation. In an emulator, the complete functionality of a hardware processor is replicated by software. It allows running a program onto different platforms, regardless the processor architecture or Operating System (OS). This technique imposes a high performance penalty and thus was discarded for our solution.

Native virtualization is the alternative to full-blown emulation. Rather than replicating a hardware platform, native virtualization provides an adaptation layer to guest machines [4]. In this scenario the three main elements are: the host OS, the guest OS and the VMM (Virtual Machine Monitor) or hypervisor. The VMM handles privileged instructions on behalf of the virtualized OS. This technique has been recently refined, in favor of lighter-weight approaches that improve performance by working with higher level abstractions. Paravirtualization and OS-level virtualization [5] are outstanding representatives of these technologies. In paravirtualization the guest OS is changed to cooperate with the VMM and the host, issuing system calls either to real devices or to the VMM. OS-level virtualization consists of partitioning the existing OS layer. Its main limitation is that host and guests must have the same OS.

A common factor among virtualization technologies is a performance penalty over x86 architectures when handling privileged processor instructions issued by the guest OS, because they have to be dynamically translated by the VMM or controlled by the host OS. This problem has been addressed by processor manufacturers [6] and now there is hardware support that accelerates VMM call management.

**Table 1 Virtualization Technologies**

| Product | Type | License | Highlights | Guest Performance |
|---|---|---|---|---|
| Bochs | Emulator | Open Source | Allows debugging the guest OS | Very slow |
| QEMU | Emulator/ Native virtualization | Open Source | Supports a wide range of hardware architectures | Slow (10% of host) / Close to native |
| VMWare | Native Virtualization | Commercial | Provides a mature product family to manage virtual infrastructures | Close to native |
| VirtualBox | Native Virtualization | Dual license | Remote desktop protocol support in commercial version | Close to native |
| UML (User | Paravirtualization | Open Source | Stable support for Linux systems | Close to native |

| | | | | |
|---|---|---|---|---|
| Mode Linux) | | | | |
| Xen | Paravirtualization | Open Source | Supports vm migration on the fly | Native |
| OpenVZ | OS-level virtualization | Open Source | Efficient resource partitioning | Native |

Table 1 shows a comparison of some of the most relevant commercial and OSS (Open Source Software) technologies for server virtualization. There is a trade-off between the performance of the product and its flexibility. Our aim is supporting a wide range of production scenarios and thus a unique virtualization technology cannot be selected beforehand. Since many of the analyzed technologies share a conceptual basis, e.g. guest systems are distributed as image files, a generic approach to support them is feasible.

We have so far focused on the creation of virtual nodes, but in order to emulate a distributed production scenario we also need to create a virtual network. Communication among the nodes is a must, as well as connectivity with external systems. Node virtualization technologies usually provide their own networking options. As an alternative to interconnect different solutions we opted for VDE (Virtual Distributed Ethernet) [7] that provides a virtual switch capable of creating a virtual network compatible with multiple node virtualization technologies (Xen, QEMU and UML).

# 3. SYSTEM DESIGN

## 3.1 Use Case Analysis

Our solution consists of a tool framework aimed at easing distributed applications execution and testing. It provides developers a personal system to deploy, start and test their applications. We carried out a use case analysis in order to understand what functionalities the tool system should provide.

The target scenario is a complex distributed system with connection to Internet. Potentially, the nodes included in the system are powerful servers, with a plethora of different software components running on them. As a reference, in the context of the ITECBAN project [8] we are using a distributed runtime environment composed of four Linux-boxes provisioned with certain middleware components that provide an advanced SOA/BPM (Business Process Management) execution environment (see the right hand side of Figure 2).

In our study we identified two main actors: the developer and the system administrator. The developer codes, runs and tests applications in the distributed system. The administrator's responsibility is managing the basic IT infrastructure (OS, installed components, databases and so on) for the virtualized elements included in the system.

Under these assumptions we identified the following use cases:

- Administrators create virtual node configurations. The tool system must provide editors and storage facilities for these configurations, including the virtual images. Node configurations can be eventually used to build complete virtual distributed systems; the data included in these node configurations should be as close as possible to the production environment.
- Developers (testers) or administrators define the deployment scenario, modeled in resemblance to the physical distributed system. Tools must enable a rapid definition of virtualized scenarios (including network, nodes, and software layers), with minimum effort. Scenarios can be reused and shared.
- Developers launch any virtual model. Applications can be deployed to the virtual runtime environment. Tests over the production site can therefore be executed in the developers' machines.
- Monitoring must be supported. The state of the virtual system can be continuously monitored by the developer, which can also carry out some operations controlling the execution of the host as well as the guest.

## 3.2 Distributed System Metamodel

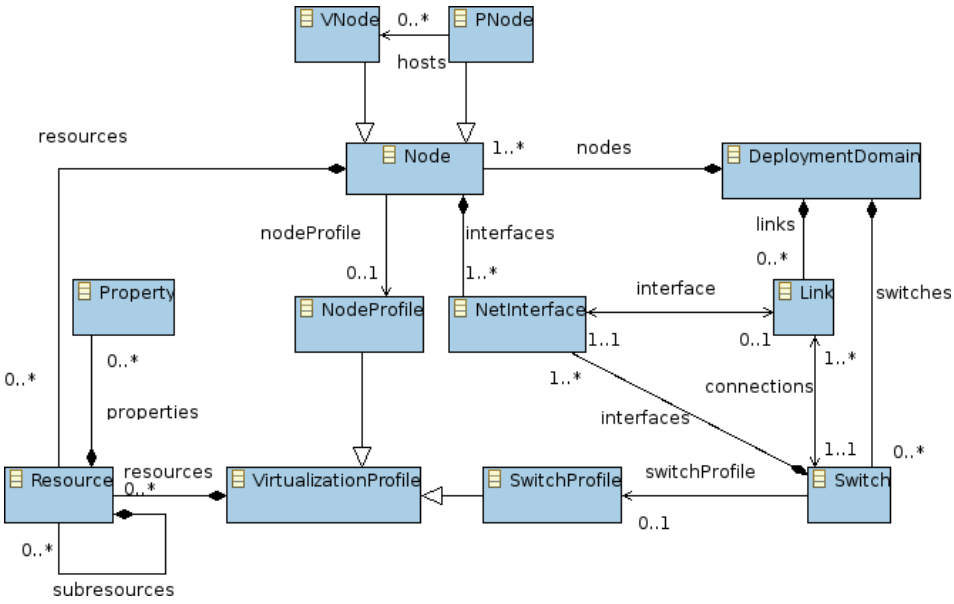


**Figure 1 Distributed System PIM**

A primary goal of our tool system is supporting a wide range of virtualization technologies such as Xen, QEMU, VDE, and UML, while retaining vendor independence. We have followed an MDA (Model Driven Architecture) approach to design our tool framework. The PIM (Platform Independent Model) is shown in Figure 1. PIM instances are transformed and enriched to PSM's (Platform Specific Models) by framework extensions, which contribute the specifics of particular technologies.

Of course, we did not design our PIM from scratch. Distributed system modeling is not a novel field. We found valuable inputs for our model in the OMG D&C (Deployment and Configuration for Distributed Systems) specification [9] and network simulation technologies such as VNUML[10], a DSL for easily defining complex network configurations. As we also intended to deploy software over runtime virtualized environments, we thought the OMG D&C would be a good basis for our model. However, the specification assumes the deployment target is already in place, whereas we need to create the virtual environment on the fly. Network simulators have already addressed this situation, but they are strongly tied to the underlying technology, e.g. UML in VNUML. We have designed the PIM in a technology agnostic way in order to handle as many technologies as possible. This problem vanishes once the virtual environment is launched, because the means to describe it are perfectly compatible with the D&C model.

With these factors in consideration, we have defined the PIM as shown in Figure 1. As in the OMG D&C model, the central entity is the deployment domain. A deployment domain aggregates switches and nodes, which can be qualified with resources modeling software, hardware and communication capabilities. The model includes virtual nodes (*VNodes*) and physical nodes (*PNodes*). A *PNode* can be included in a PIM instance either because it hosts *VNodes* or because it is a target for software deployment. Our PIM considers multi-hosted simulations; the association between *PNode* and *VNode* allows expressing host relationships.

Virtual networks are represented by the *Switch* element. Nodes connect to a network through the *NetInterface* element and the resulting connection is modeled as a *Link*. Configurable network parameters are included in the *NetInterface* and *Switch* entities.

In order to promote model reuse the PIM includes the notion of profiles. A profile is a collection of parameters that completely define a virtual node or switch. Profiles are linked to a specific virtualization technology, e.g. VDE. In addition, profiles are stored and can be shared among developers using the profile repository (see left-hand side of Figure 2). Development teams will share complete model instances or reference profiles, e.g. an application server profile (including the OS, network services and server applications). Change control in profile definitions is supported by means of version attributes in the PIM. Following the aforementioned example, a development team could test applications against different versions of the application server profile.

## 3.3 Tool Framework

The architecture of our tooling environment is shown in Figure 2. Basically, each development station is leveraged with our virtualization tools. Distributed development is supported by networked repositories. Typically, an organization controls its

software development with an SCM repository. Our tools use two additional repositories: the SIR (System Images Repository) and the MPR (Model & Profiles Repository).

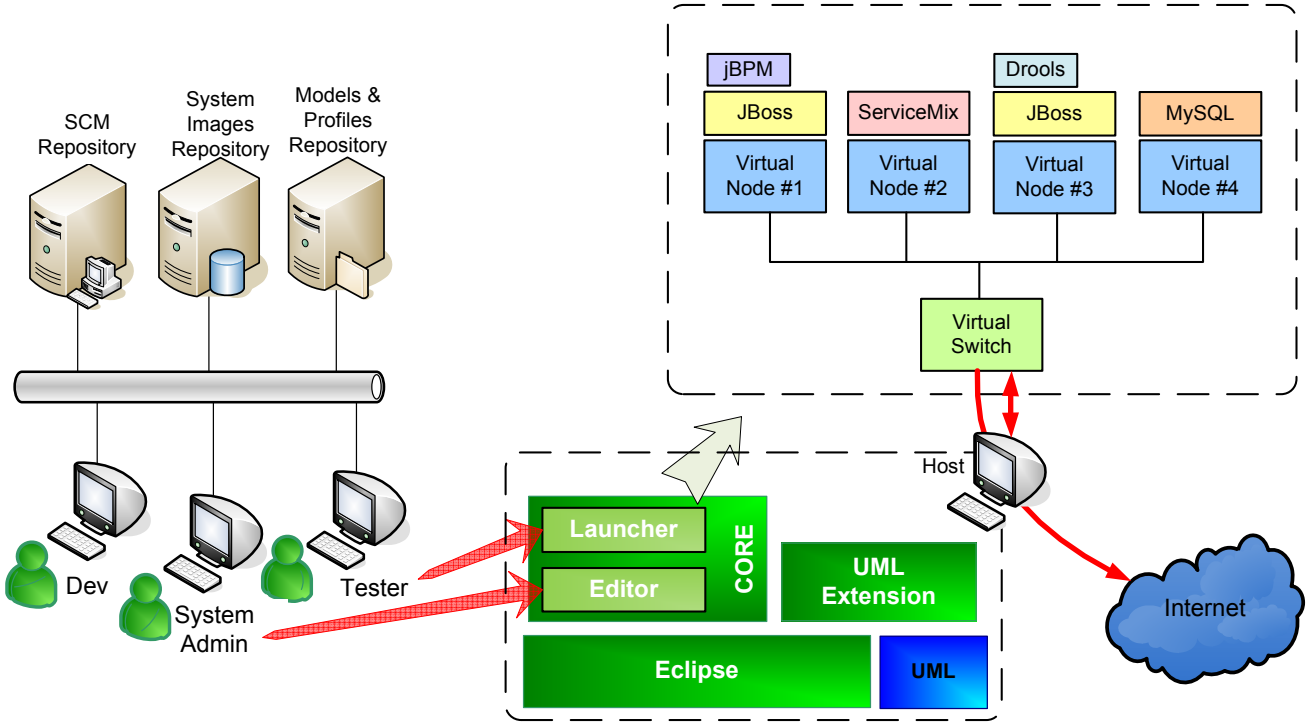


**Figure 2 Tool Architecture**

System administrators create virtual system images and store them at the SIR. Both the OS and middleware (without applications) are stored as a virtual image. Administrators create, configure and add virtual images to the SIR. The extended IDE retrieves from the SIR the necessary elements to launch a virtualization.

Eclipse was selected as the base platform to integrate our tool, because of its wide adoption and its extensible architecture [11]. Eclipse's architecture is leveraged by an OSGi kernel [12], which provides a service oriented programming model to the Eclipse platform. Through this model we have divided the system's architecture into two main blocks: a core framework and technology extensions. This architecture facilitates the integration between virtualization technologies (strongly tied to the OS level) and the IDE. Our framework currently supports the definition and execution of distributed system virtualizations with VNUML, UML, VDE, QEMU and XEN.

The core framework builds on the distributed systems metamodel, adding three fundamental aspects: profile definition, model creation and virtualization launch:

Firstly, the core provides the profile manager. This component allows administrators creating, editing, saving and sharing profiles through the MPR. Profiles are defined in the PIM as the necessary configuration parameters to fully characterize a virtual node or switch. Profiles allow uniformly handling element instances. However, profile parameters depend on the specific technology, and thus have to be provided by technology extensions. Additional configuration parameters such as native technologies installation location and correctness checking can also be configured through preference pages.

Secondly, it includes a distributed systems editor. This component provides a rich, graphical editor for creating model instances. An administrator can drag and drop nodes, networks and connections and quickly define distributed system models. Manual configuration is reduced to a minimum thanks to the use of default values and the abstraction provided by profiles. This abstraction allows the editor to be technology agnostic. Thanks to that, the editor supports any technology contributed through the extension points without modifications.

And finally, the core is capable of launching and controlling virtualization scenarios. Initiating a launch requires no additional arguments or configuration, and is invoked just with a mouse click. Internally, each technology extension creates a virtualized element instance from the profile parameters. For instance, when we launch a distributed virtualization scenario with a VDE switch, the VDE delegate analyses the range of IPs for both the nodes connected to its network and the switch configuration

parameters. This extension uses DNSMasq to provide DNS/DHCP services to the guest nodes. Also, the host network configuration is modified to provide external Internet access to the guests, as well as virtual consoles. Networking processes are controlled by the tool framework in order to provide a clean virtualization stop.

An example of extension to the core is the VNUML one, which contributes extensions for configuring the native technologies as well as defining and storing VNUML profiles. Besides, the plugin registers a launcher for VNUML systems, able to receive the PIM model as input, and perform a model transformation, converting the generic model into the VNUML PSM model, which can be directly processed by the VNUML Perl parser. The tool system also supports extensions for QEMU, UML, and XEN virtualization technologies. Conceptually the implementation of the abovementioned extensions is similar to the VNUML one.

## 4. CASE STUDY

On the right-hand side of Figure 2 a representation of a distributed enterprise runtime environment is depicted. The virtual runtime uses OSS servers to leverage a SOA/BPM architecture. The entire configuration has been created beforehand by administrators. It is composed of four UML/Linux virtual machines interconnected by a VDE switch. The following is a description of the services installed in the environment:

- Virtual Node #1. JEE application server (JBoss) extended with JBPM. JBPM is a runtime for long-term processes. It manages the processes life-cycle by means of interacting with the database installed and configured in the Virtual Node #4.
- Virtual Node #2. ESB (Enterprise Service Bus) platform (ServiceMix), that provides a service-oriented integration layer for the virtual environment
- Virtual Node #3. A JBoss server extended with Drools, an enterprise framework that provides a Business Rules Management System (BRMS) for business rules edition, change and management.
- Virtual Node #4. Database (MySQL) for applications persistence.

This is an excellent example of a complex enterprise runtime. However, once created it can be seamlessly integrated in the developer IDE (in this case, Eclipse), so developers don't need to know the internal details of this distributed system. This approach allows developers to focus on developing, deploying and testing their software components, regardless of the details of the host, virtual nodes and services.

Figure 3 is a snapshot of our virtualization tool in action in this case study. The left hand side panel shows a deployment scenario composed of four nodes connected to the network switch. The deployment model can be launched with no further configuration required. On the right hand side of the image we can see the terminals of the virtual nodes at runtime. Through them, ftp connections can be opened to the external SCM repository in order to get, install, configure and launch the versions of the software to be tested. As there is connection from the development host to the virtual system, the developer can interact with the system using a common web browser.

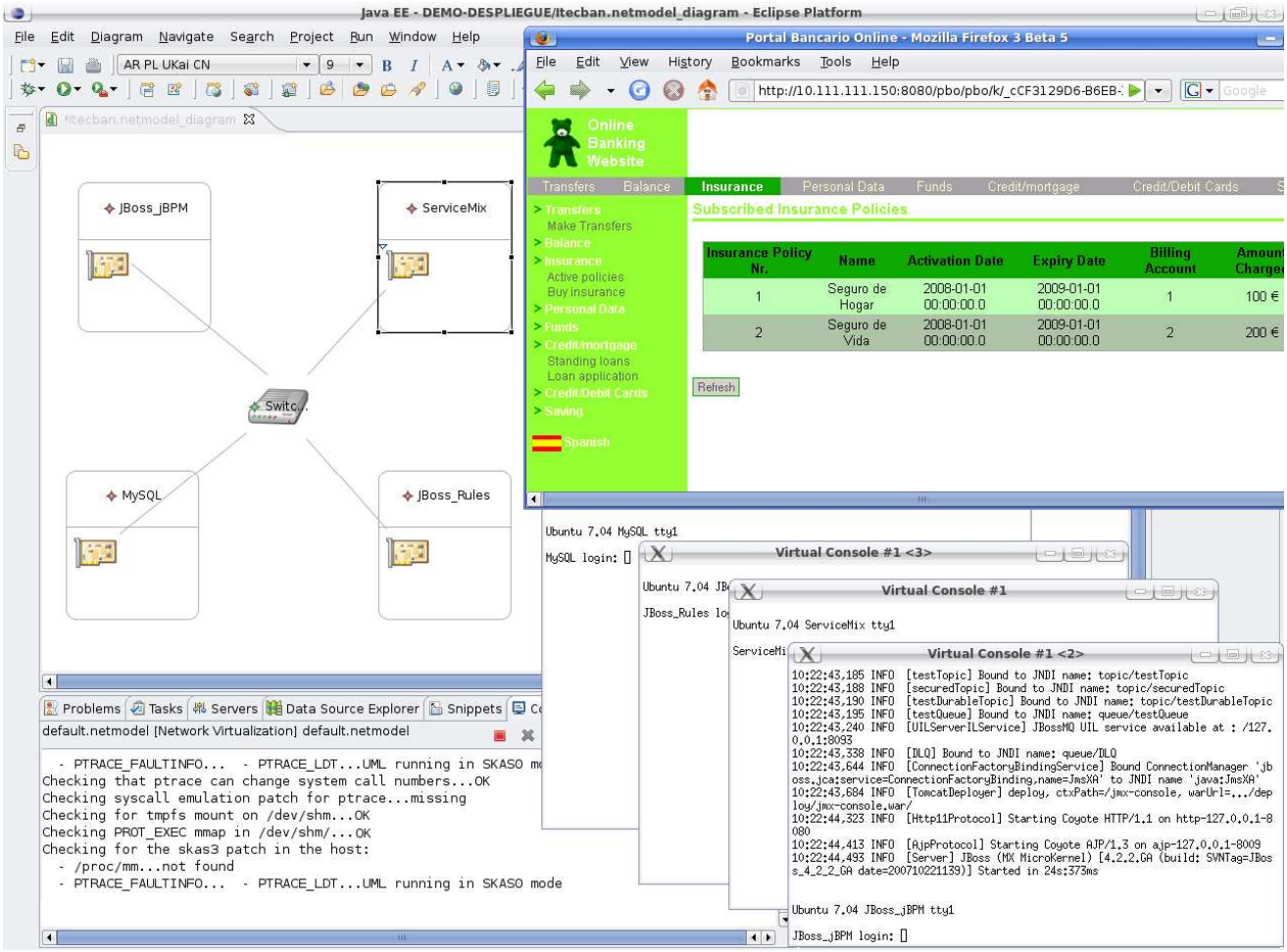


**Figure 3 Virtualization tool in action**

## 5. CONCLUSIONS AND FUTURE WORK

In this article, we report our experience in designing an Eclipse-based virtualization tool framework. Our aim is offering developers a personal runtime environment for launching and testing their applications. The availability of a runtime environment breaks barriers imposed by access control to the production and pre-production sites. In addition, it reduces the burden of the IT administrators. Virtualization technologies were selected as a suitable means for facing these problems. We designed the architecture of our tool system after analyzing software developers needs. Virtualization technologies share a conceptual basis, which enabled us to design our tool system as a framework. Following the MDA approach, we created a platform independent model, specific extensions and model transformations for certain virtualization technologies.

There is still work on the way, especially regarding the integration with a software deployment and testing infrastructure that automates component distribution, resolution of component dependencies (keeping version constraints) and adaptation of the process to the virtual network resources. Additionally, the tool system (see Figure 2) includes a System Images Repository (SIR), managed by IT administrators. It hosts a collection of virtual nodes ready to be used in virtualizations. Currently, each variant of a node image must be stored as a complete file. This approach is not very efficient, because the repository is rapidly filled with multiple versions of the same base image. This limitation can be addressed by modularizing image persistence in order to assemble images on the fly. However, this modification presents important technical challenges, as the repository logic would have to cope with a wide range of operating systems and software deployment mechanisms. Each supported deployment model needs a specific weaver for generating images. A repository based on this principle will have all the advantages of current SCM systems, combining versioning capabilities and optimizing storage resources with differential storage.

There are also limitations to the use of this tool: while it has proven successful for functional testing, it is clearly out of scope for stress, performance and reliability testing, as the behavior of guests is not the same than the actual systems. In fact, the developers' machines should be resourceful enough to hold several production machines without interferences, but this is not so common. Just to overcome this problem we are working on the capability to launch remote virtualizations on a dedicated cluster of hosts, with enough raw computing power to faithfully mirror the actual production systems.

Another ongoing line of work aims at monitoring the virtual system through the tools environment. We are developing a resource gathering infrastructure, that will enable us to detect anomalous behavior in the system, and to create a model of a mirror virtualized system from extracted information of the real one.

## 6. REFERENCES


[1] Lentz, J. and Bleizeffer, T. *IT Ecosystems: Evolved Complexity and Unintelligent Design*. Proceedings of the 2007 symposium on Computer human interaction for the management of information technology, ACM Press, ISBN:1-59593-635-6, March 30–31, 2007, Cambridge, MA, U.S.A.

[2] Mendoza, A. *Utility Computing. Technologies, Standards and Strategies.* Ed. Artech House, 2007. ISBN: 1-59693-024-1

[3] Seetharamanm, S. and Murthy, K. *Test Optimization Using Software Virtualization*. IEEE Software, Vol. 23, Issue 5, September 2006

[4] Rosenblum, M. and Garfinkel, T., *Virtual Machine Monitors: Current Technology and Future Trends.* IEEE Computer, Vol. 38, Issue 5. May 2005

[5] Vaughan-Nichols, S. *New Approach to Virtualization Is a Lightweight*. IEEE Computer, Vol. 39, Issue 11, November 2006.

[6] Uhlig, R et al, *Intel Virtualization Technology*.; IEEE Computer, Vol.38, Issue 5, May 2005

[7] Davoli, R. *VDE: Virtual Distributed Ethernet.* Proceedings of Testbeds and Research Infrastructures for the Development of Networks and Communities conference. Tridentcom 2005.

[8] Ruiz, J.L. Dueñas, J. C. and Cuadrado F. *A Service Component Deployment Architecture for e-Banking.* Third International IEEE Workshop on Service Oriented Architectures in Converging Networked Environments (SOCNE 08). March 2008, Okinawa, Japan, ISBN: 978-0-7695-3096-3. pp. 1369-1374.

[9] *Deployment and Configuration of Component-based Distributed Applications Specification*. OMG formal specification version 4.0 formal/06-04-02.

[10] Galán, F., Fernández, D., Ruiz, J., Walid, O. and de Miguel, T., *Use of Virtualization Tools in Computer Network Laboratories.* Proceedings of the Information Technology Based Higher Education and Training, ITHET 2004, Istanbul, Turkey, ISBN: 0-7803-8596-9.

[11] Gamma E., Beck K, *Contributing to Eclipse: principles, patterns and plugins*, Ed. Addison-Wesley, 2003

[12] Gruber, O., Hargrave, B. J., Rapicault, P., McAffer, J. and Watson, T., *The Eclipse 3.0 Platform. Adopting OSGi Technology*, IBM Systems Journal, 2005


## 7. ACKNOWLEDGEMENTS


ITECBAN is an IT innovation project partially funded by CENIT (a Spanish public R&D program). The authors are grateful to MITIC (Ministerio de Industria, Turismo y Comercio) and CDTI (Centro para el Desarrollo Tecnológico e Industrial) for supporting ITECBAN through CENIT.