A workflow on the dynamic composition and distribution of orchestration for testbed provisioning

Christos Tranoris, Spyros Denazis
University of Patras
Greece
tranoris@ece.upatras.gr, sdena@upatras.gr

Anastasius Gavras
Eurescom GmbH
Germany
gavras@eurescom.eu

Abstract— **Pan-European laboratory (Panlab) is based on a federation of distributed interconnected testbeds, providing access to platforms, networks and services for testing. Core components of Panlab is a tool called "Teagle" which provides the means for a customer to express the testing needs and the Panlab Testbed Manager (PTM) which implements the interactions between the set-up and configuration requests by Teagle and the components in the testbed it manages. This paper discusses a solution and proposes a workflow for distributing the federation's intelligence during provisioning not only to Teagle but also to each testbed (each PTM), where local issues like policies, dependencies, constraints and even availability must be solved. Among other application areas, the dynamic provisioning of testbed resources is important in the area of experimentation for future Internet resilience.**

Keywords- Testbed, Federation, Federated Testbeds, Panlab, rule-based composition, orchestration

I. INTRODUCTION

Richer environments for testing and implementation are the demand of future Internet activities. Panlab, the Pan-European laboratory, is based on a federation of distributed testbeds that are interconnected, providing access to required platforms, networks and services for broad interoperability testing. In this context a testbed federation is the interconnection of two or more independent testbeds for the temporary creation of a richer environment for testing and experimentation [1]. The following main roles of the Panlab concept are presented in figure 1.

Coordination of the testing activities, such as infrastructure provisioning, and the overall maintenance of the environment, is ensured by the Panlab Office.

The Panlab architecture relies on a tool called "Teagle" which provides the means for a customer to express the testing needs. Teagle utilizes the Panlab repository (database of partner testbeds) and manages the complete set-up of a desired testbed infrastructure, like resource reservations and interconnection. Functionalities of Teagle tools are described in [2] and further specific details can be found in the documents available at [3].

Another important component for the implementation of a testbed infrastructure is the Panlab Testbed Manager (PTM).

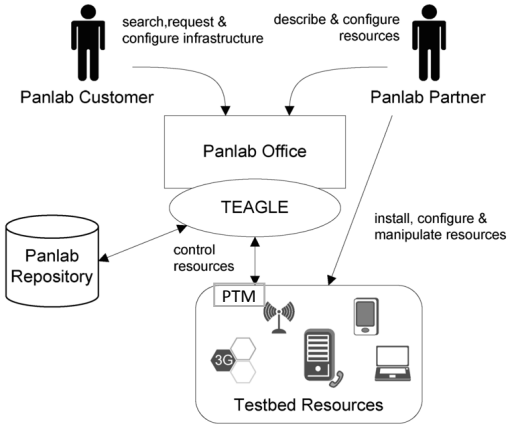

Figure 1: Panlab components and roles

It is implementing the interactions between the set-up and configuration requests by Teagle and the components in the testbed it manages. The PTM is therefore capable to adapt commands received as web services into control commands applicable to the testbed component.

This paper proposes a dynamic service orchestration by distributing the federation intelligence during provisioning not only to Teagle but also to each testbed (each PTM), where local issues like policies, dependencies, constraints and even availability must be solved. First one possible application area and its abstract requirements are presented. Then, a three layer view of a testbed's infrastructure is presented, depending on the level of abstraction. Then the testbed provisioning follows, mapping each view to a layer. The workflow for the dynamic composition of orchestration and distribution of provisioning actions is described in detail. Finally, suggested ways of implementing such an orchestration composition service are also presented.

II. EXPERIMENTATION ON FUTURE INTERNET RESILIENCE

Information and Communication Technologies (ICT) are the heart of our Information Society and have become a critical infrastructure. This means that a disruption of its smooth operation has major impact on the European and worldwide economy and society. Disruptions caused by natural disasters, terrorist attacks, malicious human action,

9781-4244-3941-6/09/$25.00 ©2009 IEEE

hardware failure or faults in software pose serious risks. Today, the main assertions about the resilience, robustness and security of the Internet are based on real world incidents, as well as from platforms for testing of functional properties. However the assessment of these properties of the Internet should be based on more solid and compelling evidence. Substantiated assertions concerning the resilience, robustness and security of the Internet is obviously necessary for technical reasons; however they become imperative for supporting policy and business decisions.

There is a strong need to use experimental platforms suitable for conducting empirical security research. Such experimental platforms will enable (i) researchers to use rigorous scientific methods for studying vulnerabilities, threats, systemic faults, or malicious actions, (ii) operators and technology providers to try new systems under different security scenarios, and (iii) authorities to better understand the security implications of the Internet infrastructure and the related applications.

Due to the increasing complexity of the Internet, attention is being devoted to large scale experimental platforms testbeds that can support resilience, robustness and security experiments spanning the network, service delivery and application layers. Such Experimental ICT Platforms for Internet Resilience and Security should be specifically tailored for this purpose.

The dynamic composition and distribution of orchestration for testbed provisioning is one possible way to enable experimentation on resilience in an environment in which failures and other incidents can programmatically be induced.

III. THREE VIEWS OF A TESTBED'S INFRASTRUCTURE

From user requirements until testbed provisioning, 3 different views of a testbed's infrastructure have been identified: the User View, the Federation View and the Testbed View. Each view drives a path from a more conceptual and abstract view of a testbed (a Virtual Testbed) to the more detailed view; in brief it starts from what the user wants, how the federation translates the requirements and finally how the Virtual Testbed is implemented.

The **User view**, where an example conceptually depicted in figure 2, has the requirements of the user; the conceptual design of the testbed that he wants: the Virtual Testbed. The design is topology agnostic about the physical location of the resources and how the federation will allocate and provision them. In this view the user will search for available computing resources and service offerings across the federation.

This view can be created in many ways: textually (by filling forms), graphically (by using a graph tool), by means of a Domain Specific Language (DSL) or a combination of all. The goal remains the same: give to the user the ability of defining in a consistent way the requirements of his virtual testbed. This requirement definition could be done also in several views given by the Teagle tool, depending on the way the federation gathers the requirements. For example: Connectivity view where the user selects computing resources connectivity and networking; Services view where the user can select and configure services that must be installed and preconfigured by the federation. The user will also require services to be installed for each resource and will define how these services will interoperate.

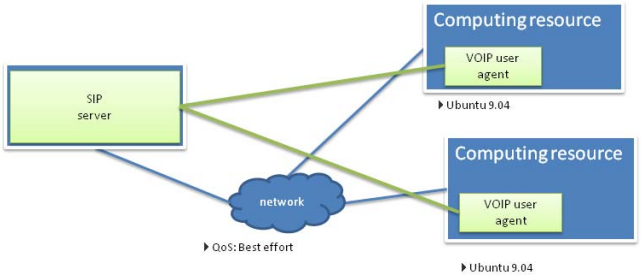

Figure 2: User view

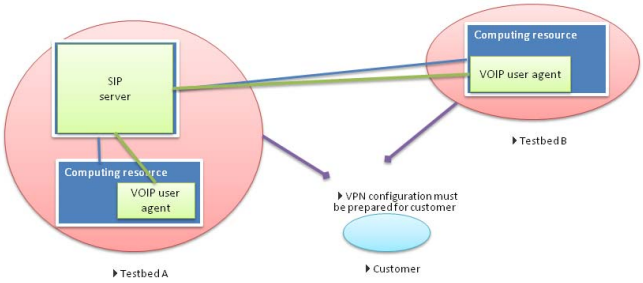

Figure 3: The Federation view

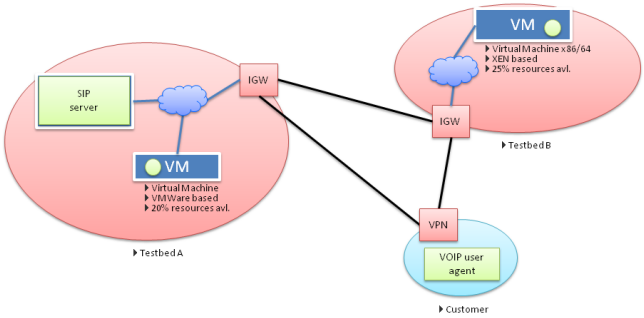

Figure 4: The Testbed view

In the **Federation view**, conceptually depicted in figure 3, a federated infrastructure is expressed: the testbeds that will participate for the creation of the Virtual Testbed, every testbed resource and what is going to be provisioned. It is the way the federation "transforms" customer requests and takes actions on proper orchestration of services. This view is agnostic about the internal infrastructure and topologies of the testbeds. Federation's responsibility concerning connectivity is on the boundaries of the testbeds. Requests should be made on each participating testbed in order e.g.

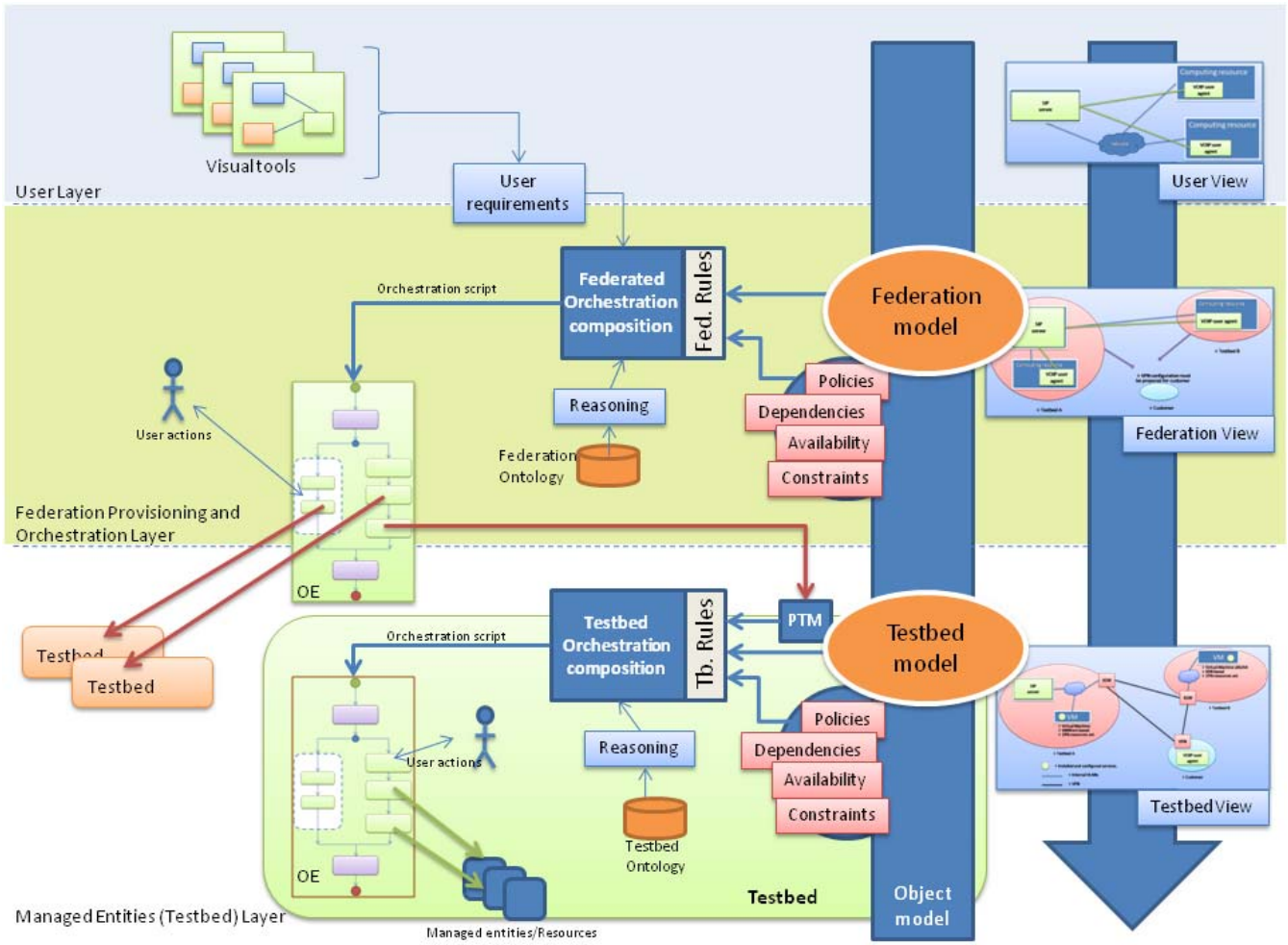

Figure 5: Views, Layers and workflows for dynamic orchestration on each layer

to create a VPN between all testbeds, to "unite" computing resources into the final federated testbed, to deploy Operating Systems, Services, Applications and finally configure them.

Finally, the **Testbed view**, as conceptually appears on figure 4, contains the realization of the Virtual Testbed. It is the actual federated Testbed as implemented by federation requests. It fulfills the requirements of the Federation view. It exhibits all internal infrastructures and consists of internal gateways and switches participating for the fulfillment of user requirements of the Virtual Testbed.

IV. TESTBED PROVISIONING

Testbed Provisioning starts when the customer and the Panlab office have both agreed on the SLA (scheduling, QoS, availability of resources, results management, legal issues, etc), and Teagle can now initiate the provisioning of the testbed in order to be delivered to the customer. Provisioning is carried out through a number of interfaces implemented by the Panlab control architectural elements. One major aspect of testbed provisioning is the wide variety of configuration operations that may need to be performed on testbed components. Therefore, configuration operations depend on control interfaces available on the testbed devices while problems of interoperability for controlling these devices exist. Resolving them requires that the testbed components implement open or standard interfaces for their configuration.

One of the most challenging issues in testbed provisioning is the proper orchestration of actions. Issues like: policies, dependencies between resources, dependencies between actions and between data, availability of resources, and communication conditions that govern the federation should be taken account. The orchestration must decide for all the above issues and must "sort" and "follow" properly every necessary action (web service calls to each individual testbed), because, as mentioned in previous paragraph, only the federated testbed view has the knowledge of what must be provisioned.

Things might be much more complex, because the above issues could exist, not only across the federation, but

also internally on each provider's testbed. The problem here is that the upper view is agnostic about internal provider issues and cannot solve them in a federated manner.

A. Distribution and rule-driven dynamic composition of orchestration on testbed provisioning

A solution to the issues raised on testbed provisioning is to distribute the decision of actions and the orchestration between the Federation View and the Testbed View. This concept is illustrated in figure 5.

Each view (user, federation, and testbed) is reflected to an equivalent layer: User layer, Federation Provisioning and Orchestration Layer and finally Managed Entites (Testbed) Layer. All layers are governed from a single object or information model, like DEN-ng for example.

At the User layer, visual tools are the mean for gathering user requirements like general services or applications (e.g. some Web servers, some VOIP servers, some MySQL databases), specific services from specific providers (e.g. University of Patras' IMS network resources), availability, quality of service, connectivity, etc.

B. Federation Layer services

Federation Layer services (Teagle) must decide for the proper and correct sequence of actions in order to fulfill user requirements but also must satisfy any other issue like policies or dependencies. For example the provisioning process must first ensure that a server is installed, configured and up and running and then configure any other service that will use this server, thus, the orchestration script must be composed dynamically. Afterwards the orchestration will be executed by invoking managed resources in a proper composed sequence.

As mentioned, Federation layer services use a common object model with all layers. Part of this model is called **Federation model** and it describes entities that are known to this layer. Additionally user's requirements are expressed with entities of this common object model. This Federation model and user's requirements are the input for the composition of the federated orchestration. The following entities (which could rely on repositories) are also expressed on this layer:

- **Federation policies** that govern the whole federation: i.e. Quality of Service between testbeds
- **Dependencies** of data and actions between testbeds and their resources: e.g. the IP of a web server that was given by DHCP must be set on a service relying on another testbed
- **Availability** of resources: the dynamic composition could solve and provide dynamic allocation of resources on this level. (i.e. find a virtual machine regardless the testbed)
- **Control flow Constraints**: the correct precedence of constraints, where a service can only be reached after another service has started or a certain state has been reached.

All the aforementioned entities compose a set (a repository) of condition-action (CA) Rules (Federation Rules); Event-Condition-Action-rules (ECA) could also be used. These rules (which are actually a knowledge base of the federation) drive the decisions in order to compose the orchestration. Moreover a shared ontology could be used across the federation resources and inference rules that could be transformed into CA-rules (by assigning some values) are also provided into the rules engine. The final orchestration is then executed by the orchestration engine, where invocations are made to each testbed.

Following, an example is given: A customer needs an NTP server and ten Linux Machines synchronized by this NTP server. The federation found that testbed A supports a NTP server and testbed B offers empty Virtual Machines. How to determine what to do first? The NTP server on testbed A must be configured first then Linux must be installed on every empty computing resource of Testbed B and then configure these (ie via a cron job) to use the NTP server.

As an example, rules (expressed formally based on the model or on a Domain Specific Language) could exist like:

- Rule:
  a Linux Machine LMX is configured to use NTP server on NTPX
  THEN *ensure that the NTP server is already configured.(NTPX.configured==true)*
- Rule:
  Linux Machine LMX is to be configured
  THEN *ensure the Linux OS is already installed on LMX*

These rules for example "drive" the proper execution of actions by the orchestration engine. Reordering and injection of service invocations takes place. So the following "textual" sequence of actions could be generated:

- MyTestbed.NTPS = TestbedA.ReserveResource("MyTestbed", NTPserver)
- TestbedA.ConfigureResource(MyTestbed.NTPS, params) //must be done before configuring ntp job on clients
- MyTestbed.VM1 = TestbedB.ReserveResource(MyTestbed, VM001)
- TestbedB.InstallOS(MyTestbed.VM1,"Linux") //must be done before configuring ntp job on clients
- TestbedB.ConfigureNTPjob(MyTestbed.VM1, MyTestbed.NTPS)
- <continues for all machines>

C. *Managed entities Layer provisioning*

Actions presented in the previous paragraph are reaching first the PTM of a testbed and then each managed entity (resource) to be configured. A question raised here is the following: what is the actions' level of granularity needed to be known by the Federation Layer services?

For instance, going back to the NTP server example, the orchestration reaches a step where VLANs are created on each testbed and resources must be configured to participate on these VLANs. At this step, is it necessary for the orchestration process (executed by the upper layer - the Federation service layer) to instruct each resource to configure the underlying OS and then make the resource to be part of the VLAN? The action ReserveResource should be just enough to imply all the required actions in order to realize this step.

It appears that the same problems faced during the Federated testbed provisioning are now encountered inside a testbed as soon as an action received from the upper layer. Thus, a similar mechanism like the one on the Federation Layer is replicated here. Just before configuring each managed entity, an orchestration must be prepared. The difference here is that a Testbed model exists (again a part of the whole Object model, sharing similar entities with all layers). Also same entities are present here: Testbed Policies, Testbed dependencies, Availability and Control flow Constraints. Again these entities compose a repository of CA-rules, the Testbed Rules. This repository of rules might be common to each testbed: for example what actions to do when a ReserveResource action is requested. Also specific rules to a testbed might exist internally: What InstallOS action means for TestbedB? Should actual VLAN ids are exchanged between the layers or just the "alias" name of the Virtual Testbed is enough?

To continue with our NTP example, rules could exist like:

- Rule:
  After installing an OS
  THEN *ensure that the machine will be part of the reserved VLAN*

Of course an orchestration should be executed by the PTM and an orchestration engine must be present inside the PTM. For example a command like the following: TestbedB.InstallOS(MyTestbed.VM1,"Linux"), implies the composition of an orchestration script where dependencies must be resolved internally and a new set of actions should be properly ordered and invoked.

V. ORCHESTRATION COMPOSITION

In the previous paragraphs entities were mentioned that compose a set (a repository) of Federation Rules. These rules drive the decisions for the proper composition of the orchestration. In order for this dynamic composition to be accomplished, similar concepts to rule-driven service composition [4] might be used, such as: ability to pursue alternative execution paths, adaptability and reusability of rules. As pointed also by [4] considerable efforts have been invested in rule-engines to support service compositions. Between RuleML engines [5] and Aspect-oriented extensions to BPEL[6], noticeable is the approach on [7] where it is suggested to incorporate business rules in BPEL specifications, while enforcing them in rule engines that work in concert with BPEL engines, and coordinate themselves through an Enterprise Service Bus (ESB). Concepts from [8] might be used, like Domain Specific Languages semi-automatically generated from information model and baseline ontology for policy transformation and conflict detection. Finally a solution is examined of just using a rules engine, like Drools[9], to perform this CA process and then a service uses the rule-engine output to create the final composed orchestration by properly sorting the decisions made by the rule engine.

VI. CONCLUSION

The work presented in this paper proposes a way of distributing the provisioning process across the federated virtual testbed and additionally copes with rule-driven composition of the orchestration process. Thus, complexity might be removed from a single point and can be distributed to all other components of the testbed's infrastructure. Local issues of a testbed like policies, dependencies, constraints and even availability can be solved with this approach. This distribution of decisions on actions makes the services to be agile to changing conditions across the federation of testbeds. The dynamic composition and distribution of orchestration for testbed provisioning is one possible way to enable experimentation on resilience in an environment in which failures and other incidents can programmatically be induced.

VII. ACKNOWLEDGMENT

Parts of the work presented are of the Project PII which receives funding from the European Commission's Seventh Framework Programme.

REFERENCES

[1] Anastasius GAVRAS et al., "Control of Resources in Pan-European Testbed Federation",Towards the Future Internet G. Tselentis et al. (Eds.) IOS Press, 2009

[2] S. Wahle, A. Gavras, F. Gouveia, H. Hrasnica, T. Magedanz, "Network Domain Federation Infrastructure for Federated Testbeds", NEM Summit 2008, Saint-Malo, France, 13-15 October 2008

[3] www.panlab.net – website of Panlab and PII European projects, supported by the European Commission in framework programmes FP6 (2001-2006) and FP7 (2007-2013).

[4] Hans Weigand, Willem-Jan van den Heuvel and Marcel Hiel, "Rule-Based Service Composition and Service-Oriented Business Rule Management",Proceedings of Regulation Modeling and Deployment Conference 2008

[5] Nagl, C., Rosenberg, F., Dustdar, S., "ViDRE - A Distributed Service-Oriented Business Rule Engine based on RuleML", Proceedings of the 10th IEEE International Enterprise Distributed Object Computing Conference (EDOC'06), 16-20. October 2006, Hong Kong, China

[6] Charfi, A. and Mezini, M., "AO4BPEL: An Aspect-oriented Extension to BPEL", World Wide Web, V.10, nr. 3, pp.: 309-344, 2007.

[7] Rosenberg, F. and S. Dustdar, "Business rules integration in BPEL: A service-oriented approach", Proceedings of the 7th International Conference on E-Commerce Technology, IEEE, 2005

[8] Barrett, K.; Davy, S.; Strassner, J.; Jennings, B.; van der Meer, S.; Donnelly, W., "A Model Based Approach for Policy Tool Generation and Policy Analysis", Global Information Infrastructure Symposium, 2007. GIIS 2007. First International, p.99-105 (2007)

[9] http://www.jboss.org/drools/, The Business Logic integration Platform, JBoss Community