Multi-Cloud Governance Service based on Model Driven Policy Generation

Juan Li, Wendpanga Francis Ouedraogo and Frédérique Biennier
Université de Lyon, CNRS INSA-Lyon, LIRIS UMR 5205, 20 Avenue Albert Einstein, F-69621 Villeurbanne cedex, France

Keywords: Multi-Cloud Governance, Policy Generation, Model Driven Engineering, NFP Management, Security.

Abstract: Cloud computing is an innovative and promising paradigm that is leading to remarkable changes in the way we manage our business. Cloud computing can provide scalable IT infrastructure, QoS-assured services and customizable computing environment. Such scalable and agile environment increases the call for agile and dynamic deployment and governance environments over multi-cloud infrastructure. Unfortunately, by now, governance and Non Functional Properties (such as security, QoS…) are managed in a static way, limiting the global benefits of deploying service-based information system over multi-cloud environments. To overcome this limit, we propose a contextualised policy generation process to allow both an agile management NFP in a multi-cloud context and a secured deployment of the service-based information system. Thanks to the generation of these NFP policies, NFP management functions can be orchestrated at runtime so that the exact execution context can be taken into account.

1 INTRODUCTION

Cloud computing is transforming the way enterprises purchase and manage computing resources (Gartner, 2012), increasing corporate information System robustness and scalability thanks to multi-cloud implementation strategy. Moreover, this multi-cloud re-organisation also fits the collaborative business stake as collaborative business processes are distributed among different IS and clouds. According to a "cloud provider vision", this multi-cloud strategy leads to different challenges such as the ability to automatically provision services, effectively manage workload segmentation and portability (i.e., seamless movement of workloads across many platforms and clouds), and manage virtual service instances, while optimizing use of the resources and accelerating the deployment of new services (DMTF, 2009). These challenges increase the call for a common cloud service reference architecture, enabling cloud portability and cloud service governance.

As far as the "cloud consumer" vision is concerned, the (multi-)cloud strategy adoption increases the call for developping agile and secured deployment means so that the target cloud characteristics can be taken into account in a transparent way.

Several works (presented in the related works section) cope partly with these agile and secured deployment and governance challenges: some of them are related to the "technical" side of the multi-cloud system (such as QoS management, middcleware improving cloud portability,…) without taking into account the way business requirements can be integrated to adjust the deployment (this may lead to set more protection or resources than really needed) whereas others are mostly "Business" oriented i.e. defining high-level requirements to secure and govern the business processes without integrating dynamic knowledge related to the target (multi-)cloud.

To overcome these limits, we propose to federate the business and cloud vision in a single model-driven approach to generate security and governance policies (section 3). These policies are turned in a "model at runtime" organisation and and are used to orchestrate conveniently the required security or governance services at runtime. Lastly a use case is presented (section 4).

2 STATE OF THE ART

In this paper we adopt the NIST definition of Cloud computing which is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction (Mell, 2011). To tune the deployment process depending on the cloud characteristics, one can use the 2-dimension typology introduced by (Zhang, 2010):

- **The Service Dimension** (which refers to the common XaaS vision) is used to split the cloud offer into 4 layers (hardware/ data center Infrastructure as a service layer, Platform as a Service layer and Application / Software as a Service layer) that may even be "enriched" with a Business Process as a Service layer.
- **The Deployment Model Dimension** which depends on who owns and who uses the cloud resources. This dimension leads to classify cloud resources in 4 categories: Public clouds, Private clouds, Community clouds and Hybrid clouds which mix public and private resources.

Unfortunately, the layered service dimension does not really integrate a business vision. Moreover QoS and security management are often defined statically depending on the target deployment model.

2.1 Adapting QoS Management and Governance to Multi-Cloud Systems

As stated in (Rodero-Merino, 2010), users increase their call for functionalities that automate the management of services as a whole unit, from the business to the infrastructure layer to increase efficiency and global benefits. To this end (Papazoglou, 2006) provides a formal model to express visibility constraints, manage compliance and configure cloud resources on demand but this model do not allow a dynamic reconfiguration at runtime depending on the context.

To overcome this limit, and allow a fine-grain tuning of resources, specification at the IaaS layer must integrate both computational and network constraints while adjusting the delivered infrastructure components to the users' requirements and SLAs. Unfortunately, if pricing models can be used to evaluate the "price" of infrastructure services depending on the required SLA, they do not integrate QoS assurance (Freitas, 2012). Moreover, some low-level scalability rules, such as VMs adjustment (Vaquero, 2012) must be integrated in a larger QoS management vision as application consolidation processes (used to increase resource utilization) should take into account the performance interference of co-located workloads to fit the application required QoS (Zhu, 2012). Lastly, while optimizing the "real" resource utilization at runtime, a particular attention must be paid on the way "elastic QoS" is defined in order to avoid penalties due to the risk of deviation from the agreed QoS level (Jayasinghe, 2012) scalability, On the opposite, higher-level approaches fail to provide mechanisms for a fine grained control of services at runtime (Moran, 2011). This increases the call for a global governance system allowing an efficient execution and support of collaborative business processes, without wastes and cause of defect.

This requirements involves monitoring both infrastructure and services, taking into account SLAs, elasticity, QoS, etc (Clayman, 2010). As trust, managerial capability and technical capability have a significant relationship with cloud-deployment performance (Garrison, 2012); SLAs should be understood by both cloud expert and non experts so that common performance indicators can be recognised. Unfortunately, as stated in the SLA survey made by (Alhamad, 2011), SLA frameworks are focused on technical attributes and do neither take into account security nor other business related non-functional properties. Moreover, resources measuring techniques need further refinement to be applied in the cloud context in order (1) to ensure some level of trust between service providers and customers, (2) to provide a flexible and agile way to tune performance metrics parameters and (3) to support real costs evaluation means.

To this end, (Katsaros, 2012), proposes a self-adaptive hierarchical monitoring mechanism for clouds. This framework implemented at the Platform as a Service layer, allows monitoring the QoS parameters based on SLA for business benefits. Nevertheless it lacks of providing a flexible policy enforcement mechanism and it does not indicate its scalability in web service framework. While considering quality from a "customer" point of view, (Jureta, 2009) proposes a comprehensive quality model for service-oriented systems. It allows specifying the quality level, determining the dependency value and ranking the quality priority. However, the performance issues related to cloud resources are not discussed and details are missing regarding the correlation of the quality model with the service cost model.

Lastly, the monitoring mechanisms should be non-intrusive and should let a minimum runtime footprint over the execution environment (Heward, 2010). It has also (1) to fit the cloud elasticity to keep up when an application or infrastructure scales up or down dynamically (Gogouvitis, 2012) and (2) to provide a "simple" interface so that the cloud complexity is as transparent as possible for end users. This requires being able to "generate and deploy on the fly" contextual monitoring means.

2.2 Integration of Security Management

Due to the variety of cloud deployment models, different security challenges must be taken into account. While private Cloud deployment does not require to enforce the corporate security policy, the openness involved by the other kinds of deployment involves paying attention to data isolation and to data storage protection so that the corporate security policy can cope the particular vulnerabilities involved by the cloud deployment (Ouedraogo, 2012). To fit these security challenges, cloud security-oriented models have also been defined such as:

- **The Cloud Security Alliance Security Stack** (Cloud Security Alliance, 2012) uses the XaaS levels to identify threats and mitigation means.
- **The Jericho Forum Security Cube Model** (Jericho, 2009) uses a 4 criteria classification: (resources physical location, the resource provider, the cloud technology and the operating area) to evaluate the risks and propose the associated mitigation means.

Nevertheless, both models are organised to identify and deploy security countermeasures in a static vision and do not fit an "adaptive" security deployment in a multi-cloud context. This often leads either to "over-protect" information systems while deploying them on clouds (that may reduce the Quality of Service) or to "forget" protecting corporate assets while transferring them on a cloud platform.

In order to overcome this limit, one can adapt the Model Driven Engineering (MDE) approach as it allows generating code from requirements thanks to different transformation steps (Marcos, 2006), raising the abstraction level and introducing more automation in software development (Van Der Straeten, 2009). Such an engineering strategy can be worthy used in a multi-cloud context as it can improve reusing abilities of requirements, Platform Independent Models and parts of Platform specific models depending on the deployment platform (see for example (Torres, 2012) that present: how BP-driven web applications can be developed using technology independent models before setting transformation strategies to generate "technology specific" applications).

Moreover, MDE has also been adapted to define the Model Driven Security (MDS) strategy (Basin, 2003; Clavel, 2008). MDS defines a framework used to generate security policies out of annotated business process models (Souza, 2009, Wolter, 2009) (thanks to either UML based security model integration (Jürjens, 2005) or BPMN annotations (Mülle, 2011)) taking advantage of SOA agility and policy flexibility. Nevertheless, this approach does not allow generating and deploying automatically the convenient monitoring functions that are necessary to guaranty that the system is safe (Loganayagi, 2011).

2.3 Challenges

Taking advantage of the multi-cloud deployment to support collaborative business requires integrating a unified approach to deploy secured BP and govern performances and security from the business to the infrastructure in a dynamic way. This requires first to enrich the traditional XaaS layer model with a "Business as a Service" level, used to express business-dependant performance and security requirements. Then, to fit the dynamicity required by a multi-cloud deployment, transformation models should also integrate a "model at runtime" vision to support the necessary flexibility.

By now, the different works do not cover these requirements nor are end-user oriented (so that requirements can be captured more easily). To overcome these limits, we propose to take advantage of the well-know MDE strategy to generate and deploy service-related policies that will be used to take into account non-functional requirements (as security and quality of service) while deploying and monitoring service oriented systems over a multi-cloud infrastructure.

3 MODEL DRIVEN POLICY GENERATION PROCESS

To support the policy generation process to allow an agile management Non Functional Properties (NFP) in a multi-cloud context, we couple the MDE to the Pattern based engineering to generate service-related policies. Then NFP management functions that can be orchestrated at runtime so that the exact execution context can be taken into account. To achieve this goal, we propose a format model used to weave the business and deployment environment related knowledge in order to define transformation patterns and generate, compose and orchestrate NFP related policies.

3.1 Global Architecture

We use a multi-layer architecture to generate contextualised policies that will be used at runtime to select and orchestrate the security and governance services accordingly. Functionally, Business Requirements analysis allows designing the process workflow that is used to select and compose services to achieve each task. Accordingly, services are also used to select and orchestrate lower level components. This leads us to organise a 3-layer architecture on the top of the XaaS model:

- **Business Layer (BL)** includes all business context information, such as business deciders, business requirements that are used to identify the Business Process workflow.
- **Service Layer (SL)** includes all virtualized service context information, such as service provider/ consumer/ register and includes all the services that are selected and compose to achieve a business task.
- **Implementation Layer (IL)** includes all implementation components, such as hardware, equipments, human resources and XaaS resources etc. It contains all the components that must be composed and orchestrated to support a service execution. It is an abstraction level of the different XaaS components.

Our governance architecture has been designed in a "transversal" way on this architecture in a Governance as a Service strategy (Li, 2012). Governance services are composed and orchestrated while running the enterprises' business processes depending on the NFP management requirements and taking advantage of the "functional knowledge" to select, compose and orchestrate the NFP management components accordingly. This provides a rather non-intrusive system that minimizes its footprint by transforming and composing policy rules depending on the context thanks to a set of transformation patterns.

To select the convenient transformation patterns NFP are classified into different groups ('Performance', 'Usability', 'Maintainability', 'Reliability', 'Security'…). Each NFP group is divided into Critical Success Factors sets (CSFs) associated to metrics used to constrain and / or evaluate its accomplishment so that Governance Agreements can be set and associated to the different layers. Each agreement refers to the target layer objectives, CSFs and factors metrics.

To allow a dynamic deployment and orchestration of the NFP management components, we propose to couple NFP related policy rules to the different functional components. Using a pattern-based transformation process, requirements are turned into Platform Independent Policies and Platform Dependant Policies used to orchestrate the NFP management components at runtime. Moreover, we take advantage of the functional specification (i.e. the BP workflow description used to compose and orchestrate the different services and cloud components) to compose the NFP orchestration and governance policies accordingly.

The policy generation process (figure 1) takes advantage of both Model Driven Engineering and of Pattern-based Engineering approaches adapting the security patterns methods (Yoshioka, 2008) (Uzunov, 2012) and those proposed as "best practices" in security engineering methods (as OCTAVE, EBIOS.).

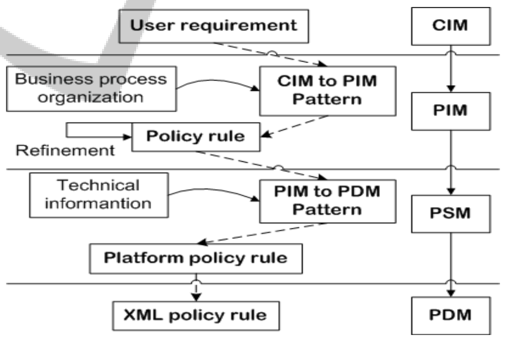


Figure 1: Policy transformation steps.

The transformation strategy relies on a global model that connects business workflow, security and governance requirements to monitoring patterns and policies (figure 2). A 'Resource' is associated either to a 'task', a 'service' or an 'implementation' component. Each resource has its own properties and interacts with others depending on the business workflow. NFP requirements (either related to security or governance) are associated to resources. Each requirement is analysed and transformed into policy rules thanks to 'transformation pattern'.

'Security rules' are used to select, orchestrate and invoke 'security deployment pattern' to implement security means accordingly. In a similar way, 'governance rules' select, orchestrate and invoke 'monitoring pattern' to constrain accomplishment of functional rules by assigning and computing 'KPIs'. 'Monitoring pattern' also can invoke 'Action engine (AE)' to tune and improve the resources performance by doing actions on related resources, then at the runtime to improve the performance of business activities.

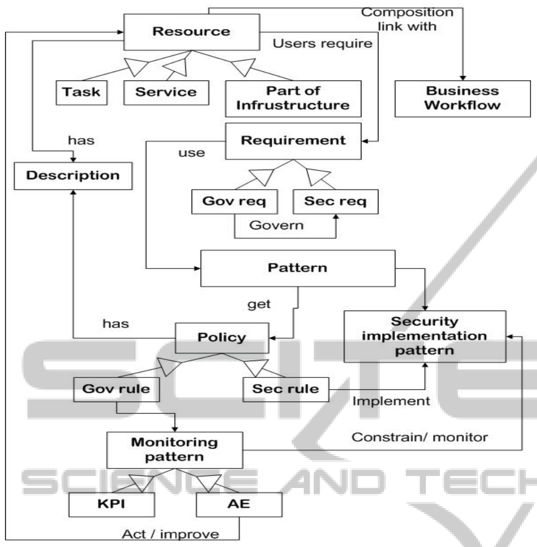


Figure 2: Global model.

3.2 Policy Formal Model

Our policy transformation process is based on a formal model integrating requirements, patterns and policy so that policy rules can be designed and transformed in a generic way.

Formally, a Requirement i is defined as a tuple:

$$\mathbf{Req_i} = (RR, (RT, RM), RG, RL, RC) \tag{1}$$

Where

- **RR (Requirement Resource)** defines the resource concerned by the requirement. It can be a task (business activity), a service or a part of the infrastructure / piece of data.
- **RT (Requirement Type)** defines the type of the requirement (governance of one of the NFP group, ensure security, support QoS control…)
- **RG (Requirement Goal)** defines precisely the goal of this requirement (governance or other functional requirement related to a NFP)
- **RM (Requirement Metrics)** is the metric (specifically defined or standardized) to measure this requirement's implementation.
- **RL (Requirement Layer)** defines the layer (BL/SL/IL) associated to the resource targeted by this requirement.
- **RC (Requirement Context)** the condition of this requirement's implementation. Such as, association of involved resource's with other resources in business workflow.

According to this definition, we can gather all the requirements for all the resources as:

$$\mathbf{Reqs} = \{Req_i\} \text{ where } 0<i<Ni; \tag{2}$$

Where "i" is the requirement number and Ni the total of the all requirements of all resources.

We can also get the requirements associated to a resource R_k ($Reqs(R_k)$) by selecting (thanks to the selection function σ) all the requirements in Reqs which associated resource (RR) matches with R_k.

$$\mathbf{Reqs(R_k)} = \sigma(Reqs.RR{=}R_k)(Reqs) \tag{3}$$

Where "i" is the requirement number and "Ni" is the total number of requirements of resource R_k.

A pattern j is defined as a tuple:

$$\mathbf{Pat_j} = (PatN, PatG, \{PatCtx\}, PatP, \{PatCol\}, \{PatR\}, \{PatCsq, PatStep\}\ ) \tag{4}$$

Where

- **PatN** is the name which identifies the pattern. This name is related to the requirement type, a NFP identification or or a CSF for a group of NFP.
- **PatG** defines the pattern goal (or the reason for using it). This goal is similar to the requirement goal and the associated value can be either governance or any other functional requirement related to a NFP or CSF.
- **PatCtx** identifies the context under which which this pattern can be used.
- **PatP** defines a list of "participants" and their roles in the pattern definition. A participant can refer to requirement which need to be transformed, transformed policy rule, relevant sub-pattern, collaborative business process organization, etc.
- **PatCol** describes the collaboration strategy used by the participants to interact with each other.
- **PatR** a set of related patterns. For example, governance requirement pattern requires a set of the CSF pattern for a parent NFP group pattern.)
- **PatCsq** describes the results or actions implemented by the pattern.
- **PatStep** defines the transformation step (CIM to PIM, PIM to PSM or PSM to PDM) for which the pattern must be used.

In a similar way, the set of patterns is defined by:

$$\mathbf{Pats} = \{Pat_j\} \text{ where } 0<j<=Nj \tag{5}$$

Where "j" is the pattern number and "Nj" the total number of patterns.

A policy rule x is also defined as a tuple:

$$\mathbf{PolR_x} = (PR, PT, PG, PL, \{PC\}, PP) \tag{6}$$

Where

- **PR** is the set of resources involved in the policy.
- **PT** is the Policy type. It refers to the requirement type and Pattern Name (RT and PatN) as well, as to the NFP group or the CSFs for a parent NFP group)
- **PG** defines the Policy goal and is related to RG and PatG such as governance or any other functional goal related to the NFP.
- **PL** is the layer of the involved resource for this policy. As we defined before it could be BL/SL/IL.
- **PC** is the set of conditions to decide if the policy can be used or not. (For governance policy rule, it is related to business process and organization of business workflow)
- **PP** identifies the pattern to use to define the policy implementation.

According to this definition, we can gather all the policy rules of all resources as:

$$\textbf{PolRs} = \{PolR_x\} \text{ where } 0<x<Nx; \quad (7)$$

Where "x" is the policy rule number and Nx the total number of policy rules (for all resources).

Lastly, policy rules attached to any resource R_k can be defined by selecting (σ) the policy rules in the PolRs while the policy resource (PR) matches with resource R_k:

$$\textbf{PolRs}(\mathbf{R_k}) = \sigma(PolRs.PR=R_k)\ (PolRs) \quad (8)$$

Where "x" is the policy rule number and "Nx" the total number of policy rules involving resource R_k.

3.3 From Requirements to Computer Independent Model (CIM)

At the beginning of the process, users define their requirements using a rather high abstraction level and do not have to provide any implementation technical details. As NFP management requirements are mostly specified at the Business layer, we use the "functional composition process" knowledge to select the resources belonging to the lower-levels and involved in the BL resource deployment and to propagate these requirements to these SL and IL resources. Our CIM elicitation process is based on this "composition-based" propagation models.

As stated in our formal model (see eq. 1), each requirement is defined by specifying the resource (RR) to which this requirement is associated to and the layer to which this resource belongs as well as the type of requirement (RT), its goal (RG) and the associated metric (RM).

As a resource R_k ('k' is numbering the resource) can be associated to many requirements, the Computer Independent Model is defined as the set of requirements associated to the different resources:

$$Reqs = \{Reqs\ (R_k)\} \text{ where} 0< i<Ni,\ 0< k<Nk \quad (9)$$

Where 'i' is the requirement number, 'Ni' is the total number of requirements associated to the resource R_k.

3.4 From Requirements to Policy Rules

After gathering and formatting the requirements in a single Computer Independent Model, the policy generation process consists in turning each CIM assertion in a Platform Independent policy rule.

Basic policy rules are generated thanks to a pattern-based transformation process. Our NFP classification is used to organize transformation patterns depending on the NFP they are related to. Pattern's name (PatN) and patterns' goal (PatG) are used to identify each pattern.

For each resource, the requirements are turned one after the other in a policy rule. To this end, for a given requirement i associated to a resource R_k: Req_i, the convenient pattern (Pat) is selected from the patterns set (Pats) thanks to the selection function (σ) that extract the pattern which name (PatN) matches the requirement type (RT) and which pattern goal (PatG) matches the resource goal(RG):

$$\begin{aligned} Pat= \sigma\ (&Pats.PatN= Req_i.RT \text{ AND} \\ &Pats.PatG =Req_i.RG)(Pats) \end{aligned} \quad (10)$$

The selected pattern is used to instantiate the corresponding policy rule. According to this, a policy rule refereeing to the requirement and the resource is generated. Let R_k be the resource associated to the i^{th} requirement Req_i, (i.e. $R_k = Req_i.RR$), the policy rule which refers to this requirement and to the k^{th} resource is defined as:

$$\begin{aligned} PolR_{ik}= (&Req_i.RR,\ Pat.PatN,\ Pat.PatG, \\ &Req_i.RL,\ Req_i.RC, Pat) \end{aligned} \quad (11)$$

After discovering the 'basic policy rule' thanks to this selection process, we have to check the selected pattern's related sub-pattern to get more precise policy rules. If a selected pattern contains a related sub-pattern (i.e. when Pat.PatR is not an empty set), a refinement algorithm (see Algorithm 1) is recursively launched to precise and develop the policy rules associated to this pattern (for example, a generic "confidentiality management" pattern can be refined using authentication and authorization patterns as well as encryption sub-patterns).

```
Ref.PatR(pattern Pat)
   {int j ; //numbering pattern;
      $Pat_j$= Pat;
      If ($Pat_j$.PatR ≠ $\phi$)
      //if pattern '$Pat_j$' has related transfer pattern $Pat_j$.PatR
    {then call Ref.PatR($Pat_j$.PatR );
     //the consequence is call refinement algorithm until the pattern has no related pattern;   }
   If ($Pat_j$.PatR = $\phi$)
    //if patter '$Pat_j$' has no related transfer pattern;
   {Then int x;//numbering policy rule;
   $Pat_j$.PatCsq = $PolR_x$;
    // if $Pat_j$ has no related pattern then its consequence is generated policy rule '$PolR_x$';   }      }
```

Algorithm 1: Refinement algorithm.

At the end of this step the different policy rules associated to the requirements are generated. As for the CIM elicitation, we use a policy composition process, including the functional composition knowledge, to select, extract and compose the different policy rules attached to a resource. Each task (in the BL level) is considered as a sub-process and used to compose / derive the policies associated to same or lower-level resources composed to implement this sub-process (see algorithms 2 and 3).

```
**Comp-subPB** (Resource R, Policy-rule PolR)
{      Int k , x;
//numbering resource and policy rule;
      $R_k$=R;       $PolR_x$= PolR;
      Extract ($R_k$.RT ,  $PolR_x$. PR );
//extract policy rule's resource and the resource type:'$R_k$.RT';
//all relevant resources are in the same sub-BP;
    If ($R_k$.RT == 'Task')
    {call Comp-task($R_k$)}
//if resource type is Task call 'Task's PIM rule composition process;
    If ($R_k$.RT == 'service')
    {call Comp-service($R_k$)}
//if resource type is service call service's PIM rule composition process;
     }

**Comp-task** (task R)
{  Int num-task, k ;//numbering task and resource;
    $R_k$= R; //$R_k$ has related services
    int num-service,n-service;
//'num-service' is the related service number;
//'n-service' is the total number of related services;
     Service $_{(num-service)}$ = $R_k$'s related service;
// 0<num-service<=n-service
     Valid $R_k$'s PIM policy rules to service$_{(num-service)}$;
//valid task's policy rules to all related service resources;
     If (service $_{(num-service)}$ has related infrastructure resource)
Call Comp-service (service $_{(num-service)}$);}

**Comp-service** (service R)
{  Int num-service,k ; //numbering service and resource;
   $R_k$=R;    //$R_k$ has related infrastructure resource;
     int num-inf,n-inf;
//'num-inf' is the related infrastucture number;
//'n-inf' is the total number of related infrastructure;
     Inf$_{num-inf}$=   $R_k$'s related infrastructure;   //  0<num-inf<=n-inf
     Valid $R_k$'s PIM policy rules to Inf$_{num-inf}$;
//valid service's policy rules to all related infrastructure resources;}
```

Algorithm 2: Cross layers sub-PB PIM policy rule composition.

```
**Comp-samelayer**(provider-resource R1, consumer-resource R2)
{  Int k1 , k2;
//'k1''k2' are resource numbers;
    $R_{k1}$= R1; $R_{k2}$=R2;
     $R_{k1}$ is provider resource;
     $R_{k2}$ is consumer resource;
//Which means $R_{k2}$'s input == $R_{k1}$'s output;
   Valid $R_{k2}$'s PIM policy rules to $R_{k1}$;
    If ($R_{k1}$, $R_{k2}$'s resource type== task and they have related resources)
{Call Comp-task($R_{k1}$)and Comp-task($R_{k2}$)};
//If $R_{k1}$, $R_{k2}$'s resource type is task and they have related services, then call task composition process;
     If ($R_{k1}$), $R_{k2}$'s resource type == service and they have related infrastructures)
     {Call Comp-service($R_{k1}$)and Comp-service($R_{k2}$)};
//If $R_{k1}$, $R_{k2}$'s resource type is service and they have related infrastructures, then call service composition process;}
```

Algorithm 3: At same level's PIM policy rule composition.

4 USE CASE

Our use case aims at generating the security and delay governance policies for a business level Payment Task described Figure 3.

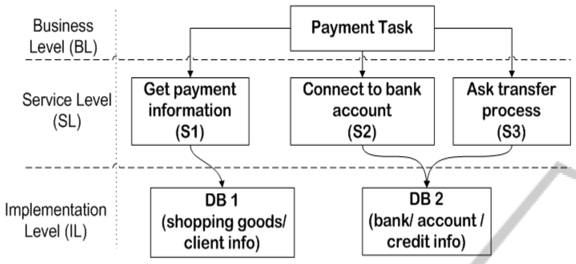

Business Level (BL)
Service Level (SL)
Implementation Level (IL)
Payment Task
Get payment information (S1)
Connect to bank account (S2)
Ask transfer process (S3)
DB 1 (shopping goods/ client info)
DB 2 (bank/ account / credit info)

Figure 3 Example of a Payment Task crossing the 3 layers.

At the first step, the user defines his requirements according to our formal model:

1) Requirement 1: Managing and governing the confidentiality at a high level (level 2)

Req_1= {"PaymentTask", ("Confidentiality", level2), "Govern", "Business layer"}

2) Requirement 2: Being able to capture the payment task execution delay at a business level to evaluate its impact to the global business performance system

Req_2 = {"PaymentTask", ("Delay"), "Govern", "Business Layer"}

Then governance patterns are identified and used to define the governance policy rule. Figure 4 presents the confidentiality and performance patterns organization.

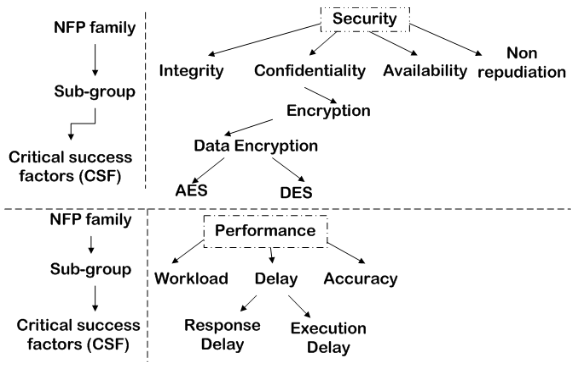


Figure 4: Organization of the Confidentiality and Performance NFP family and governance pattern.

Based on the requirement type and on the goal, the selection function is used to discover the CIM to PIM transformation pattern to apply in order to generate the policy rule associated to the payment task: As far as the security requirement is concerned, the confidentiality governance pattern is selected (Eq. 1) and checked: to identify if it contains sub-patterns (i.e. related patterns) that can be a Critical Success Factor or not. As the Conf-Pattern contains an Encryption pattern (that is not a CSF) the policy rule is refined to integrate the sub-pattern invocation (Eq. 2) and the refinement algorithm is launched is re-launched to refine the Encrypt-Pattern which is associated to CSF.

(R_1=Req_1.RR==PaymentTask)
$PolR_1(R_1)$={"PaymentTask", "Confidential", "Govern-confidentiality", " Business Layer", {PC},"Conf-Pattern"}) (Eq. 1)

$PolR_2(R_1)$=("PaymentTask", "Encryption", "Govern-encyption", "Business Layer", {PC},"Encrypt-Pattern") (Eq. 2)

As far as the performance requirement is concerned; we use a similar pattern selection process to support the CIM to PIM transformation, identifying the Delay pattern (Eq. 3) and refining it with the 2 CSF patterns (Response delay pattern and Executive delay pattern) (Eq. 4)

(R_1=Req_2.RR==PaymentTask)
$PolR_3(R_1)$=("PaymentTask", "Delay", "Govern -delay", "Business Layer", {PC}, "Delay-Pattern") (Eq. 3)

$PolR_4(R_1)$={"PaymentTask", "Delay", "Govern-Delay", "Business Layer",{PC},"Response delay"}
$PolR_5(R_1)$={"PaymentTask", "Delay", "Govern-Delay", "Business Layer",{PC},"Execution delay"} (Eq. 4)

Each policy rule is analyzed to identify the sub-resource related to the policy's "root resource".so that relevant policy rules can be composed. In our case, the resource attached to the policy is the payment-task which belongs to the business layer. This task is implemented thanks to a service chain composed of 3 services: S1, S2 and S3 deployed at the service layer. The policy rule is "propagated" to each service (see Eq. 5 which presents the security policy rule associated to S1). In a similar way, the selection process is used to identify the implementation level resources related to the service in order to generate the corresponding policy rules accordingly (see the example for DB 1 in Eq. 6)

$PolR_6$(S1)=("S1", "Encryption", "Govern-encyption", "SL", "","Encrypt-Pattern") (Eq. 5)

$PolR_7$(DB1)=("DB1", "Encryption", "Govern-encyption", "IL", "","Encrypt-Pattern") (Eq. 6)

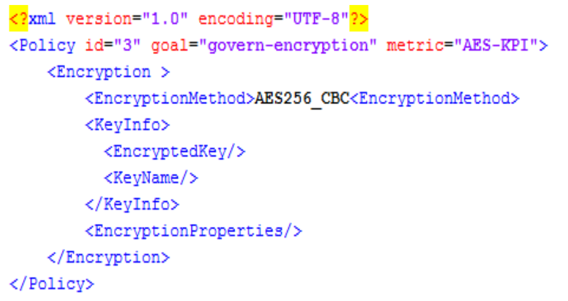
The delay governance requirement is also used to generate policy rules attached to the different services and resources (Eq. 7 and 8).

$PolR_8(S1)$=("S1", "Delay", "Govern-delay", "SL", "","ExecDelay-Pattern") (Eq. 7)
$PolR_9(DB1)$=("DB1", "Delay", "Govern-delay", "IL", "","ExecDelay-Pattern") (Eq. 8)

While deploying the task and related services, the deployment environment knowledge is used to generate platform dependant policy rules. For each platform independent policy rule, we select the "deployment" pattern depending on the PIM rule name and on the deployment platform characteristics. As we use an hybrid cloud platform to support the Payment Task deployment, the transformation pattern is selected thanks to the following criteria: PatStep is associated to the PIM to PDM transformation step, PatG fits the "Encryption level 2" goal and PatCtx fits the hybrid cloud context. This leads to select the AES-256 data encryption implementation pattern. So for each PIM rule that refers to the Encrypt-Pattern, the AES-256-KPI pattern is substituted (Eq. 9 and 10)

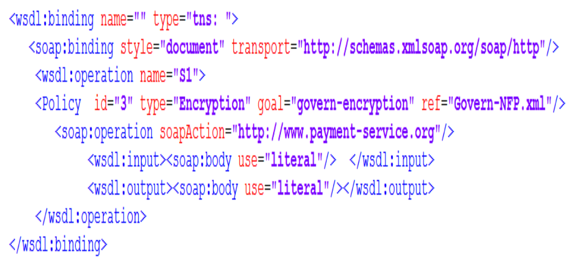
$PolR_{10}(S1)$=("S1", "Encryption", "Govern-encyption", "SL", "","AES-256-KPI pattern") (Eq.9)

$PolR_{11}(DB1)$=("DB1", "Encryption", "Govern-encyption", "IL", "","AES-256-KPI pattern") (Eq. 10)



```
<?xml version="1.0" encoding="UTF-8"?>
<Policy id="3" goal="govern-encryption" metric="AES-KPI">
    <Encryption >
        <EncryptionMethod>AES256_CBC<EncryptionMethod>
        <KeyInfo>
          <EncryptedKey/>
          <KeyName/>
        </KeyInfo>
        <EncryptionProperties/>
    </Encryption>
</Policy>
```

Figure 5: Extract of Policy file "Govern-NFP.xml".



```
<wsdl:binding name="" type="tns: ">
  <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
   <wsdl:operation name="S1">
   <Policy  id="3" type="Encryption" goal="govern-encryption" ref="Govern-NFP.xml"/>
     <soap:operation soapAction="http://www.payment-service.org"/>
         <wsdl:input><soap:body use="literal"/>  </wsdl:input>
         <wsdl:output><soap:body use="literal"/></wsdl:output>
   </wsdl:operation>
</wsdl:binding>
```

Figure 6: Policy reference added in Service S1 description (WSDL: Binding part).

Then these policy rules are used to generate the XML policy file that is attached to each service. Figure 5 shows the policy file related to the encryption whereas figure 6 shows the service annotation which refers to the policy file. At runtime, each policy rule is analysed and leads to the execution of the convenient security / governance service invocation while orchestrating the Governance KPIs. This PaymentTask's relevant KPIs' results can be aggregated into comprehensive result for business decision makers.

5 CONCLUSIONS

To support governance functions and secured deployment in a multi-cloud context we proposed to take advantage of the MDE and pattern-based engineering approaches to generate NFP management policies depending on the deployment process. Our multi-level architecture built on the top of the XaaS model allows taking advantage of the Business knowledge to derive and compose policy rules at each layer based on a single business requirement and deploy them depending on the execution context. Further works will focus on the service orchestrator component so that the policy rules will be used to compose and orchestrate the NFP management and governance services "on the fly" depending on the exact deployment context.

ACKNOWLEDGEMENTS

This work has been partly supported by the French economy ministry DGCIS under the Web Innovant Process 2.0 project grant.

REFERENCES

Alhamad, M., Dillon, T., Chang, E., 2011. a survey on SLA and performance measurement in cloud computing, *on the move to meaningful internet systems: OTM 201*, springer Berlin, 7045, 469-477.

Basin, D., Doser J., Lodderstedt, T., 2003. Model Driven Security for Process Oriented Systems, *In SACMAT '03: Proceedings of the eighth ACM symposium on Access control models and technologies*.

Clayman, S., Galis, A., Chapman, C., Toffetti, G., Rodero-Merino, L., Vaquero, L. M., Nagin, K., Rochwerger B., 2010. Monitoring Service Clouds in the Future Internet. *In: Tselentis, G and Galis, A and Gavras, A and Krco, S and Lotz, V and Simperl, E and Stiller, B and Zahariadis, T, (eds.) Towards the Future Internet - Emerging Trends from European Research.*,pp. 115 - 126

Cloud security alliance, 2012. Security Guidance for Critical Areas of Focus in Cloud Computing V3, https://cloudsecurityalliance.org/wp-content/themes/csa/guidance-download-box.php.

Clavel, M., Silva, V., Braga, C., Egea, M., 2008. Model-Driven Security in Practice: An Industrial Experience, *ECMDA-FA '08 Proceedings of the 4th European conference on Model Driven Architecture: Foundations and Application*s, pp.326 – 337.

DMTF Informational, 2009. Interoperable Clouds – A White Paper from the Open Cloud Standards Incubator

Freitas A.L., Parlavantzas N., Pazat J., 2012. An Integrated Approach for Specifying and Enforcing SLAs for Cloud Services.; *In Proceedings of IEEE CLOUD*. pp. 376-383.

Gartner, Inc. analysts, 2012. Gartner Report. *Top 10 Strategic Technologies* for 2012.

Garrison, G., Kim, S., Wakefield, R. L., 2012, Success factors for deploying cloud computing. *Commun. ACM 55(9)* pp., 62-68.

Gogouvitis, S., Konstanteli, K., Waldschmidt, S., Kousiouris, G., Katsaros G., Menychtas A., Kyriazis D., Varvarigou, T., 2012. *Workflow management for soft Real-time Interactive applications in virtualized environments*. Future Generation Computer Systems 28 (1), 193–209.

Heward, G., 2010. Assessing the Performance Impact of Service Monitoring. In Proceedings of the 2010 21st Australian Software Engineering Conference *(ASWEC '10). IEEE Computer Society*, Washington, DC, USA.

Jureta, J.I., Herssens, C., Faulkner S., 2009. A comprehensive quality model for service-oriented systems. *Software Quality Control 17 (1)*, , 65-98.

Jericho Forum, 2009. “Cloud Cube Model: Selecting Cloud Formations for Secure Collaboration,” Jericho Forum, Version 1.0, http://www.opengroup.org/jericho/cloud_cube_model_v1.0.pdf.

Jürjens, J., 2005. Model-Based Security Engineering with UML, *FOSAD 2004/2005, Springer-Verlag Berlin Heidelberg*, pp.42-77.

Jayasinghe D., Swint G., Malkowski S.,Li J., Wang Q, Park J., Pu C., 2012. Expertus: A Generator Approach to Automate Performance Testing in IaaS Clouds, *IEEE Fifth International Conference on Cloud Computing*, pp. 115-122,

Katsaros, G., Kousiouris, G., Gogouvitis, S.V., Kyriazis D., Menychtas, A., Varvarigou T., 2012. A Self-adaptive hierarchical monitoring mechanism for Clouds, *Journal of Systems and Software, 85 (5)*, 1029-1041,

Li, J., Biennier, F., Amghar, Y., 2012. Business as a Service governance in a Cloud organization. *Proceedings of the I-ESA Conferences 6, Enterprise Interoperability V*, pp. 355-365.

Loganayagi, B., Sjuatha, S., 2012. Enhance Cloud Security by Combining Virtualization and Policy Monitoring Techniques, *Procedia Engineering, 30*, 654-661,

Mell, P., Grance, T., 2011. The NIST Definition of Cloud Computing, *NIST Special Publication* 800-145.

Marcos, E., Acuria, C., Cuesta C., 2006. Integrating Software Architecture into a MDA Framework, sofware architecture, pp 127-143.

Mülle, J, von Stackelberg, S., Böhm, K., 2011. Security Language for BPMN Process Models, Karlsruhe institute of technology, Germany.

Moran, D., Vaquero, L.M., Galan, F., 2011. Elastically Ruling the cloud: specifying application’s behavior in federated clouds. in: *IEEE International Conference on Cloud Computing - CLOUD, pp*. 89-96,

Ouedraogo, W.F., Biennier, F., Ghodous, P., 2012. Adaptive security policy model to deploy business process in cloud infrastructure. *The 2nd International Conference on Cloud Computing and Services Science, CLOSER* 2012. Porto, Portugal, pp. 287-290.

Organization for the Advancement of Structured Information Standards (*OASIS), 2009.OASIS: Reference Architecture Foundation for Service Oriented Architecture*, Version 1.0.

Papazoglou M., Van Den W., Heuvel, 2006. Service-oriented design and development methodology. *Int. J. Web Eng. Technol. 2, 4*, 412-442.

Rodero-Merino L., M. Vaquero L., Gil V., Galán F, Javier Fontán J., Montero R. S., Llorente I. M., 2010. From infrastructure delivery to service management in clouds, *Future Generation Computer Systems, Volume 26, Issue 8,* 1226-1240.

Souza, A., Silva B., Lins F., Damasceno J., Rosa N., 2009. Sec-MoSC Tooling – Incorporating Security Requirements into Service Composition. *Proceeding ICSOC-ServiceWave '09 Proceedings of the 7th International Joint Conference on Service-Oriented Computing*, pp 649 – 650.

Torres, V., Giner, P., Pelechano, V., 2012. Developing BP-driven web applications through the use of MDE techniques, *Software & Systems Modeling, Springer-Verlag Volume 11, Issue 4*, pp 609-631

Uzunov, A. V., Fernandez, E. B., Falkner, K., 2012. Securing distributed systems using patterns: A survey, *Computers & Security, Volume 31, Issue 5*, pp.681–703.

Van Der Straeten, R., Mens, T., Van Baelen, S., 2009. Models in Software Engineering: challenges in Model-Driven Software Engineering, *Springer-Verlag Berlin*pp. 35 - 47 .

Vaquero L. M., Morán D., Galán F., Alcaraz-Calero,J. M., 2012.Towards Runtime Reconfiguration of Application Control Policies in the Cloud, *Journal of Network and Systems Management,Volume 20, Issue 4*, pp 489-512.

Wolter, C., Menzel M., Schaad A., Miseldine P, 2009. Model-driven business process security requirement specification, *Journal of Systems Architecture JSA*, 211–223.

Yoshioka, N., Washizaki, H., 2008. A survey on security patterns, *Progress in Informatics, No. 5* pp. 35-47.

Zhu Q., Tung, T., 2012. A Performance Interference Model for Managing Consolidated Workloads in QoS-Aware Clouds, *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference*, pp.170-179,

Zhang, Q., Cheng, L., Boutaba R.., 2010. Cloud Computing: state-of-the-art and research challenges, *J: Internet Services and Applications,1(1)*, 7-18