Cloud Resources Placement based on Functional and Non-functional Requirements

Asma Guesmi[1], Patrice Clemente[2], Frédéric Loulergue[1] and Pascal Berthomé[2]
[1]*LIFO EA 4022, Université d'Orléans, - Bâtiment IIIA*
Rue Léonard de Vinci, B.P. 6759, F-45067 Orléans Cedex 2, France
[2]*LIFO EA 4022, INSA Centre-Val de Loire, 88 boulevard Lahitolle, CS 60013, 18022 Bourges cedex, France*

Keywords: Broker, Cloud computing, Security, Requirements.

Abstract: It is difficult for customers to select the adequate cloud providers which fit their needs, as the number of cloud offerings increases rapidly. Many works thus focus on the design of cloud brokers. Unfortunately, most of them do not consider precise security requirements of customers. In this paper, we propose a methodology defined to place services in a multi-provider cloud environment, based on functional and non-functional requirements, including security requirements. To eliminate inner conflicts within customers requirements, and to match the cloud providers offers with these customers requirements, we use a formal analysis tool: Alloy. The broker uses a matching algorithm to place the required services in the adequate cloud providers, in a way that fulfills all customer requirements. We finally present a prototype implementation of the proposed broker.

1 INTRODUCTION

With the rapid growth of cloud Computing, the cloud customer now faces many choices to fit her needs. The user now wants to deploy distributed applications on federations of clouds. Basically, those can be sets of virtual machines on clusters of several providers. Cloud brokers have thus appeared, designed to match the user's requirements against cloud providers offers, in order to establish "Service Level Agreements" between them. There exist however very few approaches aiming at providing configurable security, allowing the customers to express precisely their security requirements. In this work, we propose a cloud brokering process to deeply analyze the user's model, including abstract needs and requirements, i.e., functional requirements (services, resources) and non-functional ones (connections, security properties and policies). We also propose to confront these requirements with the providers' offers, in order to automatically generate a deployment architecture for each customer. This architecture responds to every functional and non-functional requirement of the user. Our solution uses formal methods to analyze the security properties and to ensure that they are well described and will be well implemented onto the Cloud. This avoids malformed security policies. We also use formal verification tools to check the possibility of deploying resources and ensuring the security requirements of the customer before the deployment in the cloud. The resulting generated deployment architecture is then given back to the customer, who can accept it, or ask for the next one, if possible.

The rest of the paper is organized as follows. Section 2 describes the various steps of the brokering process we propose, and details the specification of providers offers, customers models and requests, and matching of both. Section 3 describes the current prototype implementation. Section 4 compares our proposal with related works. Section 5 presents perspectives and concludes the paper.

2 METHODOLOGY

The general process consists of two big independent steps. On one hand, cloud vendors describe their offers and publish them. An offer contains templates of resources such as Virtual Machines (VMs) and non-functional offers such as security mechanisms. On the other hand, the customer describes a system she wants to have deployed in the cloud. In this paper we do not detail the syntax of the specification language. But we can use extensions of the languages presented in existing works such as (Guesmi and Clemente, 2013) or (Bleikertz and Groß, 2011). The system specification

Guesmi A., Clemente P., Loulergue F. and Berthomé P..
Cloud Resources Placement based on Functional and Non-functional Requirements.
DOI: 10.5220/0005552503350342
In *Proceedings of the 12th International Conference on Security and Cryptography* (SECRYPT-2015), pages 335-342
ISBN: 978-989-758-117-5

defined by the customer is sent to the cloud broker, as a request. The requested system contains the description of resources (VMs for example) along with their characteristics, location constraints and interconnection. It also describes security properties constraints. This customer's specification will be referred as the CIM **(Customer Initial Model)**.
The novelty of our approach is that the CIM has to be formally analyzed using verification techniques. If a property in the system is found not consistent, the broker returns a counter example to the customer. The latter must then modify her requirements in order to be consistent, if possible. If the system requirements are consistent, the broker tries to match them with the vendors offers and find a suitable resources placement. This is described in Section 2.3. The broker has to select the providers that can host the required resources and fulfill the requirements. To achieve that, the broker also uses verification techniques to provide some rigorous and formal cues to ensure that the automatically generated placement is safe and sound. This is described in Section 2.4.

2.1 Providers Offers

Each provider describes its offers **to the broker**. Offers may include:

- A set of **clusters**: (possibly virtual) groups of **physical hosts** (that host the virtual resources), their geographic locations, and a set of VM **templates**: OS, CPU, storage and some basic security assurance facilities (encryption, certification).
- A set of **physical and virtual networks, abstracted as information flows and guardians** that interconnect the clusters. Guardians are intrusion protection systems and filtering systems, such as network firewalls, integrity checkers, etc.
- A set of **conflicts or other company related relations** between clusters or providers: we call conflicting clusters, those hosting data of different critical domains. For example, a cluster hosting DoD data is conflicting with any other kind of cluster. The conflicting providers/clusters should not participate in the same cloud federation.

In this paper, we do not analyze the providers offers. We assume the existence of two types of third party certifications: The QoS certification (Rajendran et al., 2010) and the CSA STAR certification (Cloud Security Alliance, 2011b).

2.2 Customer Requirements

The CIM includes the elements that we classify as the following:

A. Functional requirements
Compute nodes (or VMs) and their characteristics such as: the CPU power, the RAM size, the storage size and the data transfer rate.
Storage nodes and the characteristics such as: the storage size, the RAM size and the data transfer rate.

B. Non-functional requirements, i.e. system constraints and security requirements:

(a) The *security level* of some nodes: some nodes may be labelled with a given security level, e.g. public, normal (or semi-public[1]), private (including: secret, top_secret, etc.). By default, the security level is set to secret (private).

(b) High-level *security relations/properties* between nodes specified by the customer:
- *Collaboration*: the customer may need some nodes to collaborate, i.e. to share information and/or computing services, etc.
- *Concurrency* (or conflict): the customer may need some nodes to work in concurrency, e.g. the concurrency between all secret nodes.
- *Access (read/write) privilege*: For example, the customer should express that some normal or secret compute nodes must be able to send information to secret storage nodes; and that some secret compute nodes can read information from some secret storage ones.
- *Isolation*: the customer may need some nodes to be fully isolated from others, e.g. all top_secret nodes have to be isolated.
- *Confidentiality/Integrity*: the confidentiality/integrity of some nodes must be ensured.

These high-level relation constraints implicitly include security properties and policies.

(c) The high-level *(security) placement properties*: *Grouping constraint*: the customer may desire to host some nodes on the same cluster or provider, e.g, collaborating nodes (in order to increase the bandwidth, security trust, etc).
Separation constraint: the opposite of the previous one. *Physical isolation*: the customer may ask to physically isolate (groups of) nodes from others. Those nodes should then be hosted "alone" on a cluster or several clusters of the same provider.

2.2.1 Derived Constraints

In our solution, the cloud broker derives these high-level (b) relations and placement constraints (c) into

[1]Public means publicly available including unknown nodes, whereas semi-public means only available from known (owned) nodes.

more concrete ones, such as information flow constraints between nodes, clusters, and providers, using system and network security protection mechanisms (i.e., guardians), location, etc. Some possible derivations rules are described hereafter. The user can be given the choice of which derivation rule may be chosen if available.

- *Collaborating nodes*: The connectivity of such nodes must be ensured. They should be linked by information flows[2].
- *Conflicting nodes*: (for example) are not linked by any information flow path (being directly or transitively), or are separated or isolated.
- *Separated/Isolated nodes*: must be separated/isolated physically (geographically) or virtually (using guardians).
- *Node confidentiality (resp. integrity)*: no outgoing (resp. incoming) information flow is possible from (resp. to) this node.

2.3 Analyzing the Customer's Requirements – Preliminary Phase

In order to avoid functional anomalies, vulnerabilities and security violations, the CIM should be consistent before deploying the corresponding services in the cloud. Indeed, the customer can make mistakes when describing the requirements. Thus, the customer specifications may contain conflicts or errors. The formal analysis of the designed system is a rigorous way to detect such conflicts.

In this preliminary phase, the customer functional and non-functional requirements are analyzed in order to detect conflicts such as: various specifications for one single resource (e.g., two different disk volumes for one single VM, various locations or encryption mechanisms for the same resource), resources defined as concurrent and collaborating at the same time, conflicting permissions (e.g., a rule authorizing a given node to read a specified storage node and another rule to prohibit the same fact.) Every conflict is notified to the customer who should make the necessary modifications on his specifications. This phase is repeated until the customer corrects the inconsistent properties of the CIM.

Once the customer requirements are consistent, the broker starts to match them with the vendors offers to select the suitable ones.

[2]Although, guardians can appear on paths between collaborating nodes, in order to block unauthorized flows or transitive flows from other nodes.

2.4 Generation of Configurations

During this step, the cloud broker tries to find matches between the customer needs (the CIM) and the providers offers. It tries to place the nodes on clusters and to interconnect them in a way to fulfill all requirements. Figure 1 summarizes the algorithm of clusters selection, nodes placement and the generations of configurations, as explained hereafter.

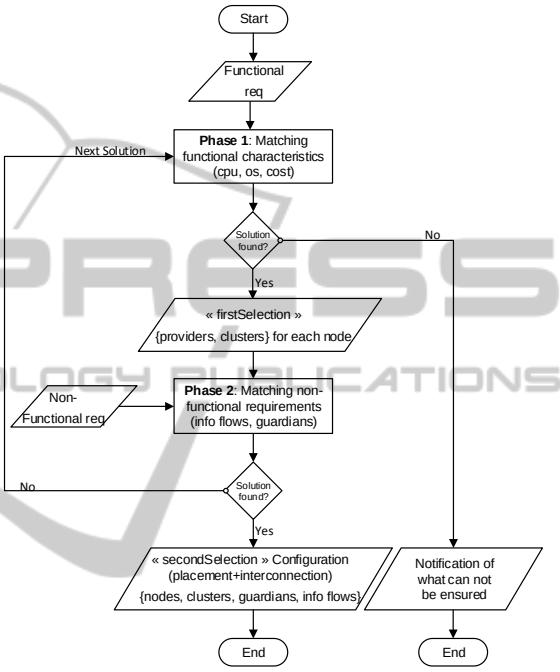

Figure 1: The flowchart of the placement algorithm.

2.4.1 Functional Requirements – Phase 1

This phase consists in matching the customer functional requirements with the providers offers and select the compatible offers.

A node is placed on a cluster only if the required node characteristics (e.g. CPU, RAM, Disk size, data transfer rate) are provided by this cluster. Other criteria (e.g., cost, location, encryption and authentication mechanisms) may also be matched at this step.

The broker forms a firstSelection set containing a selected cluster that satisfies the requirements for each node. These are potential hosts of the customer required resources.

If no match is found, the broker notifies the customer about the criteria that cause problems. For example: there is no cluster that provides Windows OS or there is not enough storage space in the Denmark area.

Figure 2 presents a possible and simple use-case of the customer requirements and the providers offers. Each provider offers a set of clusters with spe-

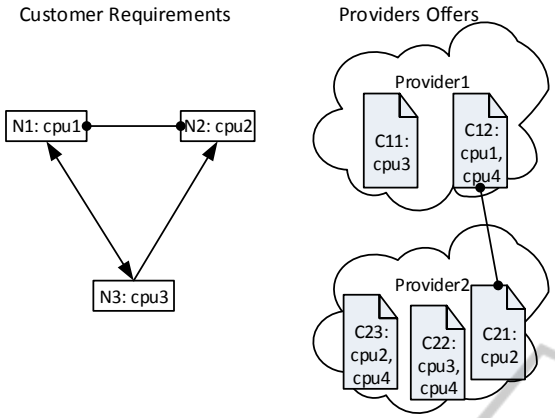

Figure 2: Example: The requirements and offers.

cific characteristics. The clusters C12 and C21 are conflicting clusters, and they cannot host nodes of the same customer. The customer requirements consist in three nodes N1, N2 and N3 with the functional constraints (for reasons of simplicity and space limitation, we consider only the CPU speed criterion) CPU1, CPU2 and CPU3 respectively. These also include the non-functional requirements:

1. Security relations: N1 and N2 are concurrent nodes, N1 and N3 are collaborating nodes.
2. Security properties: N3 has write privilege on node N2.
3. Derivated rules: concurrent nodes should reside in different physical locations and should not communicate.

Figure 3(a) shows all the possible node-to-cluster assignments corresponding to the use-case above. The broker returns only one solution at a time. Figure 3(b) shows the firstSelection set returned after the functional requirements matching by the broker. For each node, a cluster that provides fulfills functional requirements is selected as a possible host. For example: cluster C21 provides the CPU speed CPU2. Therefore, node N2 can be allocated on this cluster.

Possible assignments
N1 -> C12
N2 -> C21, C23
N3 -> C11, C22

(a) Possible assignments

firstSelection
N1 -> C12
N2 -> C21
N3 -> C11

(b) The firstSelection set

Figure 3: An example of the firstSelection set.

2.4.2 Non-functional Requirements – Phase 2

In the previous phase, the broker has selected a set of clusters that provide the required functional characteristics for all nodes.

In this second phase, the broker places the customer's nodes on corresponding clusters from the firstSelection set, then tries to interconnect nodes and clusters, in a way that the connectivity between nodes and clusters ensures the global customer's system to work properly. Otherwise, The nodes and clusters should be interconnected so that the necessary information exchanges are possible and the undesired ones are controlled and rejected. The broker also proposes a placement of a set of guardians where necessary in order to block the undesired communications.

The broker generates one placement of the nodes and interconnections. It tests and checks the configuration so as to ensure the security properties that the customer selects from the list described in Section2.2. The broker generates one configuration of placements and interconnections and checks its consistency. Finally, if the configuration ensures the functional and non-functional requirements of the customer, the broker produces a secondSelection set. This set contains the generated configuration. It represents the placement of nodes on clusters, the interconnections between nodes and clusters, the placement of guardians between nodes and clusters.

The broker notifies the customer with the proposed configuration. If the customer agrees, she confirms the broker that she is ready for the deployment and to set up the contract with the selected providers. If the broker fails to generate a placement configuration that guarantees the security properties, it generates another firstSelection solution. We call it next solution, which is different from the previous firstSelection solutions already treated. If such a solution exists then the broker retries to match the non-functional requirements. If there is no further solution, then the broker sends a notification to the customer about the properties that cannot be ensured. For example: the number of clusters in the firstSelection set is not sufficient to host the isolated nodes separately; the customer prefers that information cannot flow between concurrent nodes without using guardians, but the only clusters that can host these nodes, are already linked by flows.

Let us illustrate phase 2 with the previous example (cf. Figure 3). After that the broker matches the functional needs, it matches the non-functional and security requirements with the clusters of the firstSelection set (Figure 3(b)). In this case, cluster C12 is selected to host node N1 because it is the only one that provides CPU1. But cluster C12 is conflicting with C21, and conflicting clusters cannot host nodes of one customer. Thus, node N2 cannot be allocated on cluster C21. This solution is not possible and does not fulfill the non-functional requirements.

The broker then generates a next firstSelection solution. It is shown in Figure 4. Nodes N1, N2 and N3 are allocated on clusters C12, C23 and C11 respectively. The broker tries again to match the non-

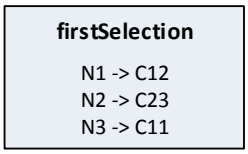

Figure 4: Example: The firstSelection set (next solution).

functional requirements using the assignments of the new firstSelection set. Figure 5 shows the generated configuration which fulfills all customer requirements. To ensure communication between nodes, the

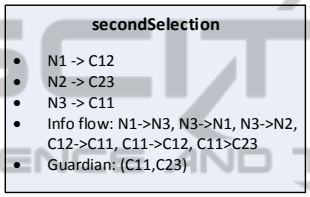

Figure 5: Example: The secondSelection set.

information flows should be set up between the collaborating nodes N1 and N3, and a flow from node N3 to N2. Thus the corresponding hosts should be linked by flows.

Guardians are placed in order to block the undesired flows. A guardian is placed between cluster C11 and cluster C23 to block the unauthorized flows from node N2 to N3 and to block the transitive flows from node N1 to node N2 via node N3 (node N1 and node N2 are concurrent and should not communicate). This configuration is shown in Figure 6. The process of placement stops here and the broker notifies the customer with this proposed configuration.

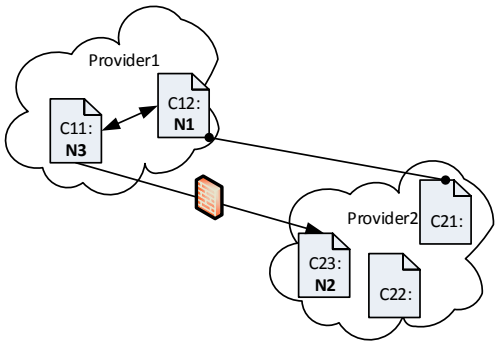

Figure 6: Example: The first configuration of the secondSelection.

2.4.3 The Formal Analysis and Matching

The broker generates a placement configuration based on the security properties described below. Then it checks the consistency of the generated configuration. Therefore, it verifies the assurance of all the required properties using the information flows that interconnect the nodes.
To be able to do that, the broker transforms any non-functional or security properties specified by the customer (cf. Section 2.2) into a set of information flows between nodes called the $Flows$.

Formal Model – The broker has a formal view of both providers offers and customers requirements. We describe a simplified subset of this formal model hereafter. Formally, it consists of the following elements:

A. Sets

- $Nodes$: the set of nodes. Nodes are actives and/or passive resources.
  There exist subsets of $Nodes$:
  - $Isolation \subset Nodes$: nodes that must be isolated
  - $Confidentiality \subset Nodes$: nodes that must stay confidential
  - $Integrity \subset Nodes$: nodes that must keep integrity
- EO: the set of Elementary (primitive) access Operations on the system. Seen at very low level (close to the infrastructure system level), these elementary operations are simple write or read access privilege from one active resource (node) to another.

B. Relations

- $Flows : Nodes \times EO \times Nodes$: the global set of information flows.
- $DirFlows \subset Flows$: The set of direct information flows.
  $\forall n_i, n_j \in Nodes, eo_k \in EO, (n_i, eo_k, n_j) \in DirFlows$ means the existence of an information flow from n_i to n_j. A 'write' access privilege is equivalent to a forward flow. A 'read' access privilege is equivalent to a backward flow. Such a direct flow between n_i and n_j will be noted $n_i \rightarrow n_j$.
- $InDirFlows \subset Flows$: the transitive closure of the $DirFlows$ relation.
  $\forall n_i, n_j \in Nodes, eo_k \in EO, (n_i, eo_k, n_j) \in InDirFlows$ means there exists an indirect (passing through other nodes) information flow between n_i and n_j.

Such an indirect flow between n_i and n_j will be written: $n_i \xrightarrow{*} n_j$.

Obviously, $Flows = DirFlows \cup InDirFlows$.

- Let *Concurrence*, *Conflicting*, and *Access*, be subsets of $Nodes \times Nodes$ relations, to represent the concurrent nodes, the conflicting nodes and the write/read privileges respectively.
- Let *Guardians* be a set of $Nodes \times Nodes$ relation, which represents the placement of guardians between nodes.

Analysis – The broker generates a configuration and check its consistency. This consists in verifying the specifications derived or expressed using information flows such as the following rules (i.e. predicates in the Alloy language).

i. Conflicting nodes should not be interconnected:
$\forall n_i, n_j \in Nodes, (n_i, n_j) \in Conflicting \implies (n_i \xrightarrow{*} n_j \notin Flows \wedge n_j \xrightarrow{*} n_i \notin Flows)$

ii. There is no information that flows into or outside isolated nodes:
$\forall n_i \in Nodes, n_i \in Isolation \implies \nexists n_j \in Nodes, ((n_i \rightarrow n_j)) \in Flows \vee (n_j \rightarrow n_i) \in Flows)$

iii. Isolated nodes can be interconnected with other nodes but protected by guardians:
$\forall n_i \in Nodes, n_i \in Isolation \implies \nexists n_j \in Nodes, \exists n_k \in Nodes, ((n_i \rightarrow n_j \in Flows \vee n_j \rightarrow n_i \in Flows) \wedge ((n_i, n_k) \in Guardians \vee (n_k, n_i) \in Guardians))$

iv. Nodes confidentiality:
$\forall n_i \in Nodes, n_i \in Confidentiality \implies \nexists n_j \in Nodes, (n_i \rightarrow n_j \in Flows)$

v. Nodes integrity:
$\forall n_i \in Nodes, n_i \in Integrity \implies \nexists n_j \in Nodes, (n_j \rightarrow n_i \in Flows)$

3 IMPLEMENTATION

We have developed the cloud broker including the previous described steps in one JAVA application. This application uses an external formal analysis tool: Alloy (MIT, 2004). Figure 7 presents the workflow of the application. The providers send their offers to the broker via a specific interface.

The customer describes its requirements on the **User Interface**. This interface is designed in a way that the customer defines easily its functional and non-functional needs, including security properties, as described in §2.2.

These requirements are translated automatically to Alloy specifications via the **text2Alloy** module. The Alloy specifications are then sent to the module that is dedicating to **analyzing the system requirements**. This module receives the Alloy customer specifications, analyzes them using the Alloy JAVA API at backend. If it detects some inconsistent properties, it sends a notification containing counterexamples to the customer. The **Alloy2text** module translates the counterexamples to visual and XML formalisms so to be easily read by the customer. The latter is supposed to modify its specifications to avoid the inconsistencies.

The formal analysis using the Alloy API are transparent to the customer. Therefore, the customer can easily modify its requirements and consult the outputs of the analysis and the placement without needing to know Alloy.

When the requirements specifications are consistent, they are transmitted to the **placement** modules. The first module consists on **matching the functional requirements** with the providers offers. This module returns the firstSelection of node to clusters assignments. The second module consists on generating the secondSelection set. This includes the placement of nodes across clusters, and setting the possible information flows and guardians between nodes and clusters so as to fulfill the functional and non-functional requirements, as described in §2.4.

If a placement is found, the corresponding configuration is sent to the client and visualized via the Alloy JAVA API. If the customer agrees then she notifies the broker with a **deployment confirmation**. The last step consists in deploying the resources such as compute and storage nodes. We did not develop this step but we can draw inspiration from existing mechanisms such as the compatibleOne broker (Yangui et al., 2014).

Alloy Example – We illustrate in this paragraph an example on how we use Alloy to specify the customer requirements. We show also how to enforce the security properties in the model. Consequently how these constraints are considered in the matching phase.
The following example shows how to enforce the security property ii (cf. §2.4.3). This property is specified by the following rule: there is no information that flows into or outside all isolated nodes(VMs in this example).
First we define some Alloy components we use in this example: A *sig* (signature) represents the basic entity of the model and contains a set of atoms. A *fact* defines the constraints that must be satisfied all the time by the described model.

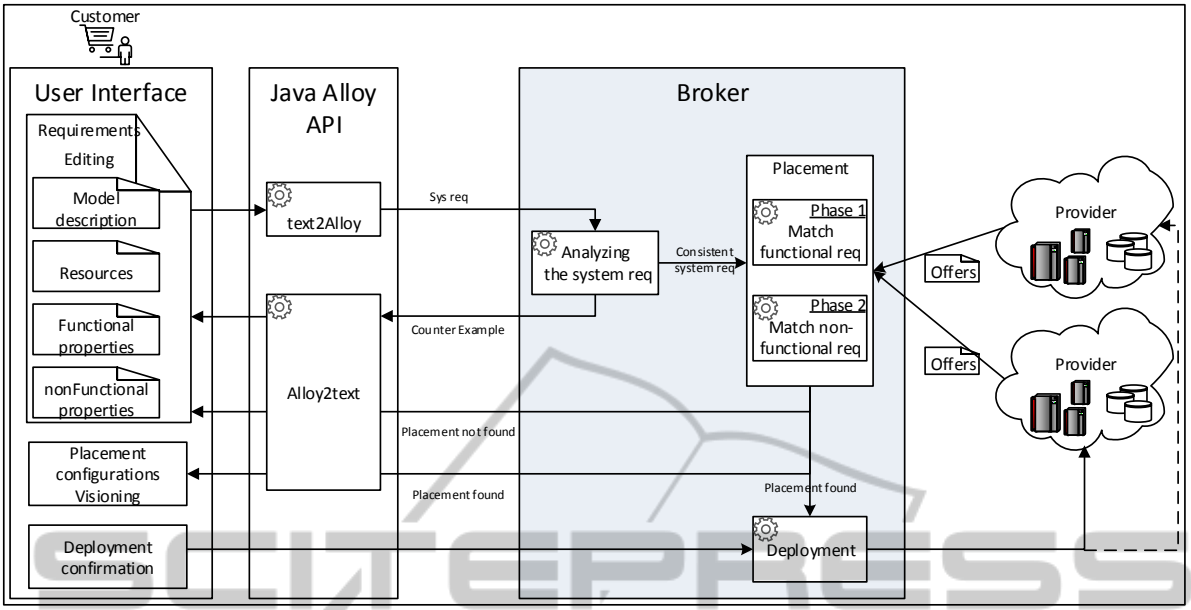

Figure 7: The workflow of the broker application.


```
sig Customer{vms: some VM}
sig VM{attributes: set Attribute}
sig Attribute{}
sig DirFlows{vm_flow: set VM→VM}
sig Isolation{isolated: set VM}

fact fact_vm_flow_isol { ∀ vi:VM | ∄ vj: VM |
vi ∈ Isolation.isolated ∧
(vi→vj ∈ DirFlows.vm_flow ∨ vj→vi ∈ DirFlows.vm_flow)}
```

The core model of this example contains the signatures Customer, VM , Attribute, DirFlows and Isolation. A customer requests some VMs. A VM is characterized by a set of Attributes (cpu, ram, etc. which we do not detail here). The signature DirFlows contains the set of VM $\times$ VM vm_flow which gathers all the direct flows of the model. The signature Isolation contains a set of isolated VM.

The fact_vm_flow_isol consists in enforcing the fact that for any VM vi such as vi is an isolated VM, there exists no VM vj such as (vi $\rightarrow$ vj) or (vj $\rightarrow$ vi) belong to the set DirFlows.vm_flow.

When the broker sets up all the flows defined in the DirFlows.vm_flow set, it also enforces all the constraints defined as facts. This is how our broker ensures that the proposed placement fulfills all the customer requirements.

4 RELATED WORK

Brokerage & Security Requirements – Very few brokerage works consider the non-functional requirements such as QoS and security. The CSA proposed a detailed set of questions (CAIQ) (Cloud Security Alliance, 2011a) to assess the security capabilities of cloud providers. A trust management system for the cloud is proposed in (Habib et al., 2014). This system allows to assess the trustworthiness of cloud providers using the the CAIQ repository. Almorst et al. proposed to evaluate security offers of providers based on the confidentiality, integrity and availability metrics (Almorsy et al., 2011) . In (Luna Garcia et al., 2012), (Garcia et al., 2013), the authors proposed methods for quantitative evaluation and ranking of the security level of cloud providers. A generic algorithm is proposed in (Jhawar and Piuri, 2013). It considers users hight level requirements to allocate applications and tries to optimize both performance and availability. Authors in (Jhawar et al., 2012) proposed a heuristics-based approach to consider security requirements for the resource management in cloud.

Formal Verification – Principally, formal methods are used to check access control policies. Several works have described how to check the consistency of access control policies and related security properties. In papers (Schaad and Moffett, 2002; Schaad, 2003) and (Toahchoodee and Ray, 2008; Toahchoodee and Ray, 2009), the authors use the Alloy language to specify access control policies and security specifications. They use Alloy analyzer to check that the access control policies are well implemented.

Bleikertz et al (Bleikertz and Groß, 2011) propose a formal language to express high-level security goals

such as the isolation management in cloud environment. They introduce in (Bleikertz et al., 2011) an approach for analyzing static virtualized cloud infrastructures. They verify the correctness of the deployment in the cloud given by a configuration snapshot.

5 CONCLUSION

In this paper, we propose a cloud brokering process based on functional and non-functional requirements. We propose a matching algorithm which matches the customers requirements with the providers offers and propose a corresponding placement configuration to the customer. We use the Alloy language and analyzer to specify the provider offers, the customer requirements (and their analysis) and the matching algorithm. Alloy generates a placement configuration which is analyzed and validated in a way to fulfill the customer requirements.

However, some elements of our global brokering process remain to be done. In particular, the numerical part of the non-functional matching phase is currently under development, using a finite domain solver, because Alloy is not suited for this kind of task. This part concerns the matching of quantities of resources (e.g. RAM or disk size, CPU frequency, number of VMs).

REFERENCES

Almorsy, M., Grundy, J., and Ibrahim, A. S. (2011). Collaboration-based cloud computing security management framework. In *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pages 364–371. IEEE.

Bleikertz, S. and Groß, T. (2011). A virtualization assurance language for isolation and deployment. In *Policies for Distributed Systems and Networks (POLICY), 2011 IEEE International Symposium on*, pages 33–40. IEEE.

Bleikertz, S., Groß, T., and Mödersheim, S. (2011). Automated verification of virtualized infrastructures. In *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*, pages 47–58. ACM.

Cloud Security Alliance (2011a). Consensus Assessments Initiative Questionnaire. https://cloudsecurityalliance.org/research/cai/, accessed on March 15, 2013.

Cloud Security Alliance (2011b). STAR Certification. https://cloudsecurityalliance.org/star/certification/, accessed on January 2015.

Garcia, J. L., Vateva-Gurova, T., Suri, N., Rak, M., and Liccardo, L. (2013). Negotiating and brokering cloud resources based on security level agreements. In *CLOSER*, pages 533–541.

Guesmi, A. and Clemente, P. (2013). Access control and security properties requirements specification for clouds' SecLAs. In *Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on*, volume 1, pages 723–729. IEEE.

Habib, S. M., Ries, S., Mühlhäuser, M., and Varikkattu, P. (2014). Towards a trust management system for cloud computing marketplaces: using CAIQ as a trust information source. *Security and Communication Networks*, 7(11):2185–2200.

Jhawar, R. and Piuri, V. (2013). Adaptive resource management for balancing availability and performance in cloud computing. In *SECRYPT*, pages 254–264.

Jhawar, R., Piuri, V., and Samarati, P. (2012). Supporting security requirements for resource management in cloud computing. In *CSE*, pages 170–177.

Luna Garcia, J., Langenberg, R., and Suri, N. (2012). Benchmarking cloud security level agreements using quantitative policy trees. In *Proceedings of the 2012 ACM Workshop on Cloud computing security workshop*, pages 103–112. ACM.

MIT (2004). Alloy: a language and tool for relational models.

Rajendran, T., Balasubramanie, P., and Cherian, R. (2010). An efficient WS-QoS broker based architecture for web services selection. *International Journal of Computer Applications*, 1(9):79–84.

Schaad, A. (2003). A framework for organisational control principles. In *PhD thesis, The University of York, York, England*.

Schaad, A. and Moffett, J. D. (2002). A lightweight approach to specification and analysis of role-based access control extensions. In *Proceedings of the seventh ACM symposium on Access control models and technologies*, pages 13–22. ACM.

Toahchoodee, M. and Ray, I. (2008). On the formal analysis of a spatio-temporal role-based access control model. In *Data and Applications Security XXII*, pages 17–32. Springer.

Toahchoodee, M. and Ray, I. (2009). Using alloy to analyse a spatio-temporal access control model supporting delegation. *IET Information Security*, 3(3):75–113.

Yangui, S., Marshall, I.-J., Laisne, J.-P., and Tata, S. (2014). Compatibleone: The open source cloud broker. *Journal of Grid Computing*, 12(1):93–109.