Towards Multi-Clouds Engineering

(Invited Paper)

Dana Petcu*, Elisabetta Di Nitto†, Danilo Ardagna†, Arnor Solberg‡, Giuliano Casale§
*Institute e-Austria Timişoara and West University of Timişoara, Romania,
†Politecnico di Milano, Italy, ‡SINTEF, Norway, §Imperial College London, UK

***Abstract*—Multi-Clouds are nowadays motivated by the needs of Cloud service consumers to ensure a certain level of service quality. Main requirements and challenges of the developing Multi-Clouds are analyzed in this paper. A particular attention is payed on the application and service portability. As complementary solution to the current approaches of using uniform application programming interfaces or standard protocols, a model-driven engineering approach dealing with the quality of service in Multi-Clouds is discussed.**

I. Introduction

Current infrastructure and platform services exposed by Cloud providers are generic and designed in the idea that one size should fit all needs. Consequently, the service consumers are forced to adapt their applications, designed to consume these services, to the available stack of software. In this context, the sequential or simultaneous usage of services and resources from multiple Clouds is pushed forward by the consumer needs. A sequentially use of services from multiple Clouds is usually related to the migration from one Cloud to another for example driven by business level needs such as emergencies, cost reductions or back-ups. A simultaneous use of services from different Clouds can provide several benefits like high availability and fault tolerance.

Despite the fact that the concept of multiple Clouds has emerged at the same time as the Cloud computing concept (due to the diversity of the practical implementations of the concept), the multiple Clouds field is still in an infancy stage. For example, an automated guidance through the variety of service offers, based on monitoring tools for the quality of services, is not yet available. Moreover, the considerable differences between the current service interfaces are hindering the easy composition or configuration of service to be consumed from multiple Clouds. Furthermore, technical barriers like interoperability and portability, data and services mobility, middleware openness need to be handled. However, approaches and tools that can manage an exhaustive set of scenarios exploiting services from multiple Clouds are still lacking.

Two basic delivery models are generally accepted for the multiple Clouds usage scenarios: Cloud Federations and Multi-Clouds. In the first case, the Cloud providers are in agreement with each others to provide the Federation aiming to enhance the service offer to their service consumers. In the second case, a third party is building a unique entry point for multiple Clouds, without a prior agreement with and between the Cloud providers. Federations are mostly implemented in the academic institutions where the agreements between the infrastructure providers are easily established, following the experience of Grid computing. Multi-Clouds are attractive for the commercial sector as being non-intrusive for the Cloud providers and bringing an added value to the third party which is gathering the services. A particular case is the Hybrid Cloud, a Multi-Cloud that connects two or more Clouds (Public and Private): in a Cloud bursting scenario external Public services are exploited when the Private services are not sufficient.

Taking into account the current need to design comprehensive Multi-Cloud support systems, we discuss in Section II the technical requirements of Multi-Clouds and the challenges generated by them. Then we consider three particular requirements highly relevant for the Multi-Cloud consumers: portability, Cloud agnostic services, and quality of service control. We argue that a model-driven engineering can deal with the heterogeneity of Cloud services in Multi-Clouds and can help in ensuring the control of the quality of the services requested by the consumers. As proof-of-concept, Section III presents the preliminary results in applying a model-driven approach for the design and execution of applications on Multi-Clouds. The last section is dedicated to conclusions.

II. Requirements and Challenges of Multi-Clouds

A. Types of Multi-Clouds Support Tools

In order to identify the requirements we first look into the current Multi-Clouds incarnations.

The authors of [1] are classifying the Multi-Clouds as library-based or service-based. In the first case, a library facilitates a uniform way to access multiple services and resources, as well as the provisioning of services and resources from multiple Clouds. The most known library-based approaches are jclouds[1], libcloud[2], δ-cloud[3]. We considered in [2] that service-based approaches for Multi-Cloud can also be classified in two categories: hosted or deployable. The most known hosted services are the commercial offers of RighScale[4] and Kavoo[5], offering management platforms for the control and administration of distributed applications deployed in different Clouds and the workloads, focusing on infrastructure services. These commercial solutions are able to deploy applications in various Clouds, but not yet able to migrate the running ones. Several deployable services for Multi-Clouds, focusing on the management and governance of Cloud services, are the results of open-source projects, like Aoleus[6], OPTIMIS[7], mOSAIC[8] and Cloud4SOA[9].

In the most complex cases of service-based Multi-Clouds, a special service act as broker between multiple services, based

[1] http://www.jclouds.org
[2] http://libcloud.apache.org
[3] http://deltacloud.apache.org
[4] http://www.rightscale.com
[5] http://www.kaavo.com
[6] http://aeolusproject.org
[7] https://github.com/optimistoolkit
[8] https://bitbucket.org/mOSAIC
[9] https://github.com/Cloud4SOA

978-1-4799-3088-3/14/$31.00 ©2014 IEEE

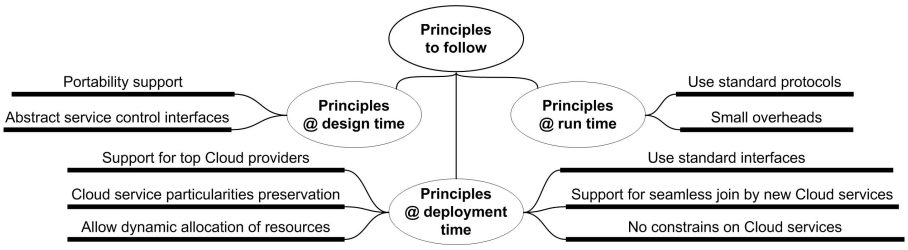

Fig. 1. Minimal Principles to be followed by a Multi-Clouds Supporting Service or Software

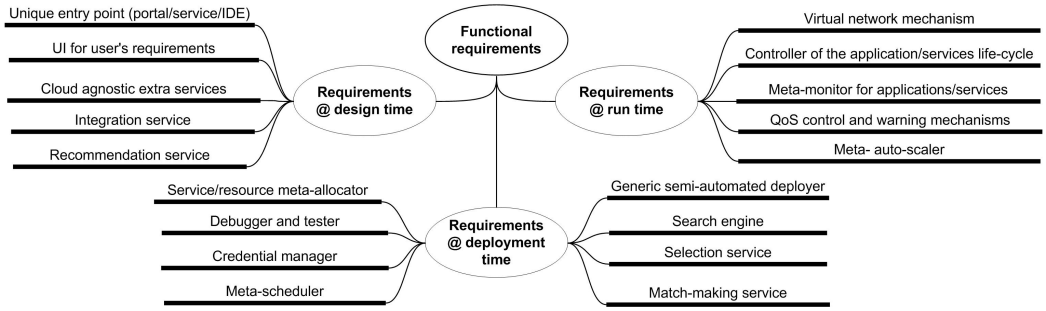

Fig. 2. Functional Requirements to be followed by a Multi-Clouds Supporting Service or Software

on service level agreements (SLAs) or provisioning rules. The well-known Cloud-independent services are SplotCloud[10], Scalr[11] and Stratos[12]. The Multi-Clouds should also offer service marketplaces, service matching with consumer's requirements, single sign-on, or monitoring of resource consumption, performance or SLA fulfillment. An example of such features is provided by Dell's Cloud Manager[13].

B. Technical Requirements

The development of a Multi-Cloud support service or software requires to find solutions to multiple problems, from business strategies and semantics of the calls and responses, to the virtual machines, data, and network orchestration. In what follows we are interested to investigate the technical requirements and the current challenges in satisfying them.

We identified the common features of the existing tools and products for Multi-Clouds support and we augmented the list with the most cited expectations from the service consumer point of view. Then we classified the features and expectations in three categories, according to the stage they appear in the life-cycle of a Cloud application or service (design, deployment, runtime). The results are resumed in Figures 1 and 2.

Figure 1 lists the minimal principles to be followed when a new service or software is designed to support Multi-Clouds. Figure 2 summarizes the requirements that we identified for the Multi-Clouds. These requirements are partially fulfilled by the current tools and products (mentioned in the previous section), but none of them is complying with all the requirements. An analysis of the degree of fulfillment of the requirements by each tool and product is out of the scope of this paper. However, we are interested to identify the challenges in fulfilling the identified requirements. We discuss here only two requirements and one principle: monitors, recommender systems, and portability support (subjects for the solution proposed in Section III). Other challenges, like the low adoption of Cloud standards or loosing the service particularities through uniformity that provides a common denominator of the underlying services, were identified and discussed in [2].

The diversity of Cloud services is a challenge for any service selection mechanism (expected to act at the deployment stage), as well as for establishing quantifiable quality of a service (QoS). A methodology to compare Cloud service must be based on multiple criteria like cost, policies, performance and so on. Moreover, optimal matching between the customer requirements (in form of SLAs or rules to be applied) with the available service should take into account functional and non-functional parameters. Even for the simple case of performance measurements, independent observer services are needed, in the context that the Cloud providers are reluctant in providing monitoring services or data about their services' performances. Moreover, the few Cloud monitoring services are heterogeneous and a meta-monitoring service at the level of Multi-Cloud is difficult to be build.

The outages of Public Clouds and their security breaks have brought into discussions the trustfulness of the Public Clouds. At design time a decision maker or a Cloud application or service developer is expecting to be guided which parts (if any) of the application or services are to be ported in a Public Cloud (if possible with which risk and which cost). A solution can be a recommendation system. Due to the complexity of the decision making process and the not yet quantified trust in Cloud services, few research prototypes of such a recommendation system are currently available.

C. Portability Requirements

Portability is needed for at least two reasons: the protection of the end user investments in developments and the need to develop a Cloud service eco-system and market.

A common understanding on a certain action or feature of

[10]http://spotcloud.com
[11]http://scalr.net
[12]http://wso2.com/cloud/stratos
[13]http://enstratius.com

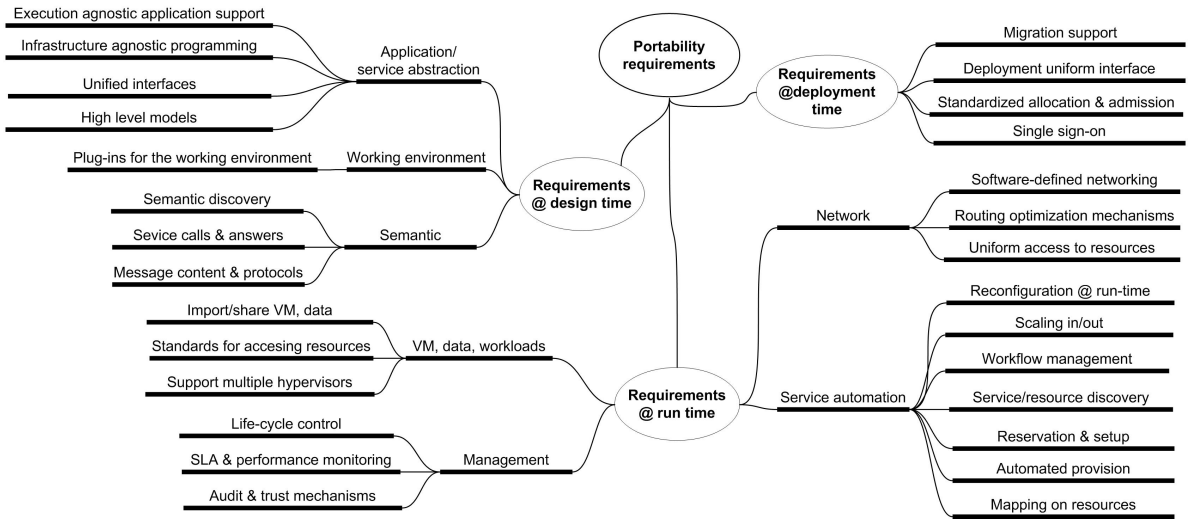

Fig. 3. Portability Requirements

a Cloud service has not yet been reached, as the emerging standards are not widely adopted and the providers are interested to differentiate themselves by unique offers. Consequently, the third party supporting the Multi-Clouds needs to understand each particular service interface in order to plug it in their service or software. This fact is not compliant with the requirement of a seamless joining procedure for a new service. If the conditions that the interface requires to comply with is constantly increased as new service providers are announced to enter in the market, the compliance with them is considered a moving target, that is difficult to reach. In this context, to ensure the portability of the services and applications that are consuming the Cloud services is a challenging task.

The current technical solutions for portability can be split in two categories: (1) low level, via open APIs, open protocols, standards; (2) high level via abstraction frameworks, semantic repositories, domain specific languages. Several open APIs were mentioned earlier (the library-based solutions). We present in the next section an approach that uses a domain-specific languages for modeling provisioning and deployment.

Figure 3 lists several requirements in order to ensure the portability in Multi-Clouds. These requirements are split into three categories according in which stage of the life-cycle is the application or service. In the following, we discuss in particular the challenges (overcome by the solution exposed in the next section) which are derived from only two requirements: infrastructure agnostic programming and standards in accessing the resources. Other challenges like the ones related to importing and sharing virtual machines and data were discussed in [3].

In order to move a running Cloud application from one Cloud to another Cloud, an inspection of the source code is needed to identify the specific API calls or to build a model or representation of the code. Tools that can do that are only in early stage of prototyping and not yet ready for production.

A classical way to ensure the portability is the adoption of standards. A classification of the standards at infrastructure service level was done in [4] and refers to access mechanism, virtual machines, storage, network, security and SLA. New ones are emerging nowadays, like TOSCA[14]. However there are several gaps in the set of available standards, for topics such as Cloud metrics and real-time monitoring, interfaces for security(-as-a-)services, and accountability associated with transparency and responsibility.

III. Case Study: Engineering QoS in Multi-Clouds Applying the MODAClouds' Approach

The quality of cloud services is currently a widely debated subject in conjunction with multiple Clouds usage scenarios. Its modeling at application design time and its evaluation at the application run-time are currently subjects for intensive research and developments. Model-driven engineering techniques can help in finding a proper solution.

We discuss in this section the early results of an initiative for providing an open source, integrated and model-driven development environment for the high-level design, early prototyping, semi-automatic code generation, and automatic deployment of applications on Multi-Clouds, with guaranteed quality of service.

A. Context

Model-driven development combined with model-driven risk analysis and quality prediction can enable application developers to specify models of Cloud services that are vendor agnostic and which are enriched with quality parameters. A model-driven framework can support in performing quality prediction. At run-time, models can be kept alive. Run-time monitoring and optimization can provide valuable information to the design time environment and hence help in filling the gap between design and run-time. To prove that this idea can work in practice is the main aim of a recently started research initiative, namely the MODAClouds project[15].

MOdel-Driven Approach for design and execution of applications on multiple Clouds (shortly MODAClouds) aims to provide methods, a decision support system, an open source IDE (Integrated Development Environment) and run-time environment for the high-level design, early prototyping, semi-automatic code generation, and automatic deployment of applications on Multi-Clouds with guaranteed QoS. The

[14]http://docs.oasis-open.org/tosca/TOSCA/v1.0/os/TOSCA-v1.0-os.html

[15]http://www.modaclouds.eu

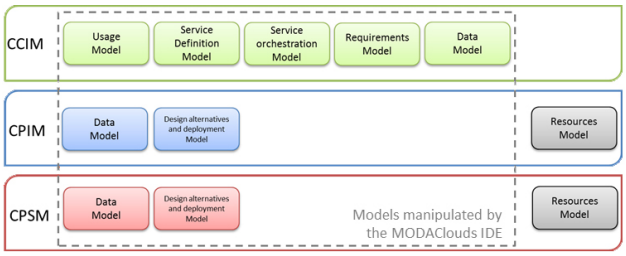

Fig. 4. Models of MODACloudsML

project is partially funded by the European Commission during October 2012 until September 2015 (ten companies and research institutions are participating).

MODAClouds concept with a motivating scenario was presented for the first time in [5]. Several more recent papers (referred in the following section), provide details about particular solutions and components. The following sub-section intends to provide an overview of the MODAClouds concept and especially the status of the concept implementation after one year of the project with a particular accent on QoS management.

B. Concepts

In order to ensure a service vendor agnosticism in a market in which diversity of the service interfaces is hindering the adoption by customers, a certain level of abstraction is needed at the application design time. In MODAClouds the Cloud services are represented and handled in a specific modelling language, namely MODACloudML [6]. This is an extension of CloudML[16], which provides a domain-specific modelling language along with a run-time environment that facilitate the specification of provisioning, deployment, and adaptation concerns of multi-cloud systems at design-time and their enactment at run-time.

The MODACloudML architecture is inspired by the OMG Model-Driven Architecture[17] (MDA), a model-based approach for the development of software systems. The specific MDA relies on three types of models for three layers of abstractions. These layers, from the more abstract to the more detailed, are CCIM, CPIM and CPSM: Cloud Computational Independent Model (CCIM) describes at a high level what the system does, hiding all the technical details related to the implementation of the system. Cloud Platform Independent Model (CPIM) describes views of the systems in a platform independent and Cloud provider agnostic manner so that it can be mapped and executed on several Cloud environments. Cloud Platform Specific Model (CPSM) refines the CPIM with technical details required for specifying how the system can use a specific platform. The models manipulated by the MODACloudML are grouped as shown in the dashed box of Figure 4. More details can be found in [7], [8].

The MODAClouds' software stack is build from several components (Figure 5). The most recent paper explaining in details the MODAClouds architecture is [9].

MODACloudML is supported by the *MODAClouds IDE*, which provides the functional, operational and data modelling environments, as well as some modules enabling the analysis of non-functional characteristics of a Multi-Cloud. From the

[16]http://www.cloudml.org
[17]http://www.omg.org/mda/

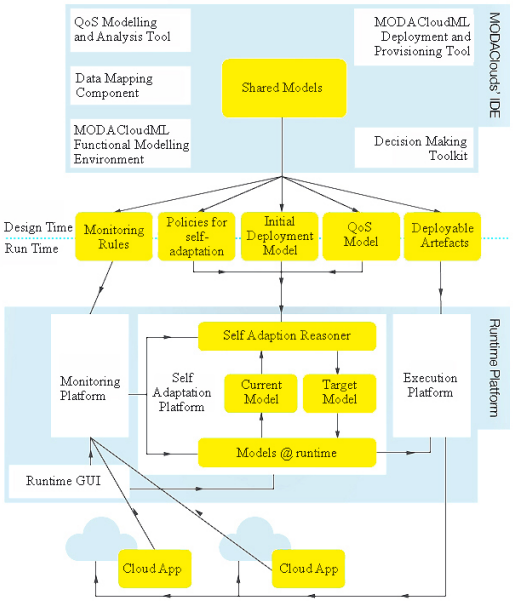

Fig. 5. MODAClouds Software Stack

point of view of the user, the IDE is the main piece of software to be used at design time. It realizes the model-driven engineering approach proposed by the MODAClouds, and its main output is the set of models and artefacts required by the runtime components to deploy, monitor and adapt cloud-based applications. The IDE includes also decision support tools to guide the construction of the application models. Furthermore, it provides a common access point for a set of tools (its core functionality), and a set of accessory functionalities to aid the use of the other tools. The IDE is composed of five tool sets:

1) The *QoS Modelling and Analysis Tool* set allows a quality of service engineer to model the QoS requirements of the application, to assess the performance of a given deployment and to identify the deployment configuration that minimizes costs. The results of the analysis can be used by the application developer to improve the design of the code and its data and by the application provider to improve the deployment.
2) The *Functional modelling Tool* set implements the MODACloudML meta-model and provides a user interface allowing the editing and storing of a MODACloudML model. It provides transformation, reverse engineering, traceability and document and code generation capabilities.
3) The *Decision Making Toolkit* allows a feasibility study engineer to model the costs and risks of an application architecture. His or her decisions will impact on application deployment, QoS and design.
4) The *Deployment and Provisioning Tool* interfaces with the runtime components and allows the application provider to deploy the application.
5) The *Data Mapping Component* allows the application developer to design the data that is manipulated and stored by the application. Data structures are analyzed from the point of view of the best runtime performance.

C. Implementation Status

The first two tool sets of the IDE are currently available for testing (see bellow), the next two are in test phase (the decision tool is presented in [10]), while the last one is under development (a proof of concept is described in the deliverable D4.4.1[18]). The *QoS Modelling & Analysis Tool* is composed by LINE and SPACE4Cloud.

LINE[19] is a tool for the performance analysis of cloud applications. It has been designed to automatically build and solve layered queuing network performance models, and is able to provide accurate estimates of relevant performance measures such as application response time or server utilisation. It can therefore be used at design time to diagnose whether the deployment characteristics are adequate to attain the desired QoS levels. LINE stands apart from other tools available for performance modelling for a number of reasons: (1) in addition to provide average performance measures, it has been designed to compute response time distributions, which can be directly used to assess percentile service level agreements; (2) features describing the specificities of cloud applications (e.g. random environments modeling reliability and multi-tenancy, or general service times representing the resource demands posed by the very broad range of cloud applications); (3) able to compute transient performance measures, capturing the effect of temporary conditions, such as workload spikes or failures of the application components.

SPACE4Cloud[20] (System PerformAnce and Cost Evaluation on Cloud) [11] is a tool for the specification, assessment and optimisation of QoS characteristic of cloud applications. It allows users to describe the architecture of their application by means of models both at CCIM and CPIM level following the model-driven paradigm. These models are then evaluated against a user-defined workload in order to assess both performance and cost of the modelled solution. The tool is built on top of the Palladio Bench modelling environment[21], and enriches its modelling capabilities, allowing more expressiveness in the definition of the resource environment and the specification of the workload. Moreover, it implements advanced meta-heuristic techniques to effectively and efficiently explore the space of possible alternative configurations. For each configuration involved in the search process the tool is also capable of evaluating the overall operative cost (it makes use of LINE for the performance evaluation).

As stated earlier, the *Functional modelling tool* implements the MODACloudML metamodel and provides a user interface allowing the editing and storing of a MODACloudML model. It uses and enhance Modelio (as detailed in [12]). Modelio[22] is a combined UML/BPMN modeler supporting a wide range of models and diagrams. Its main features are: Java code generation, scripting language support, and extensibility (it can be extended for any language, methodology or modeling technique just by adding modules).

For what concerns the status of *Runtime Platform* development, the first versions of the *Monitoring* and *Execution Platforms* are available, while the *Self-Adaptation Platform* is under development (it will be responsible for observing the monitoring data and statistics and deciding at runtime for corrective actions that can improve QoS, e.g. the deployment of new virtual machines to scale out, or dually reducing application execution cost through scale in).

The main components of the *Monitoring Platform* are the *data collectors* and the *data analyzers*. The first ones gathers information at different levels to evaluate the QoS, e.g., the consumption of system resources (like CPU, memory) and other QoS metrics (like response time of a request). The second ones processes the information gathered by the monitoring collectors and generates high-level statistics from them (e.g., analyze the correlation between different metrics to infer the cause of a given error in the system). They also performs statistical inference to estimate the parameters needed for the runtime QoS management models used in the *Self-Adaptation Platform*. Monitoring rules are received from the IDE and define what metrics to collect and how to measure them, such as the monitoring time granularity. The platform monitors them and provides triggers and data streams to the *Self-Adaptation Platform* and sends feedback to the IDE. Developers and cloud system administrators can interact with the platform through graphical user interface (part of IDE; the link between the platform and the IDE is presented in [13]) or command-line interface (a typical usage scenario considered in MODAClouds is a system administrator who wants to visualize the historical performance offered by a cloud application). Data collectors are responsible for collecting monitoring data from cloud resources and applications and to associate semantic information to the data. The data is collected periodically and the monitoring period is defined again in the monitoring rules. The monitoring data is encoded in RDF format and is processed the continuous query engine of C-SPARQL engine[23]. Two different kinds of data analysers are featured: deterministic and statistical data analysers. The first ones process at high-speed the RDF monitoring data coming from the data collectors and try to detect on-the-fly patterns that emerge directly from the data, without the need of major transformations of the data itself. The second ones extract hidden information from the data using statistical methods and generate predictions. Details about the platform functionality are provided in [14] and the public deliverables[18] related to the specifications (D6.2) and the first implementation (D6.3.1), while the source codes are available on MODAClouds repositories[24].

The *Execution Platform* handles the deployments and execution on Cloud environments (offering IaaS and PaaS), allowing the user to gather a unified experience regardless of the selected Cloud environments. Its main components are: (a) *platform system*, build from the services which are tightly related with the MODAClouds environment; (b) *infrastructure system*, build from services which handle the management of the cloud service, potentially enhancing them, and providing critical services required by other components; (c) *coordination system*, offering services like distributed communication, coordination, data storage. In order to ensure the portability of the applications, the last two components are based on mOSAIC PaaS[25] at IaaS level and Cloud4SOA[26] at PaaS level (for deployments in various Clouds without code changes).

[18] http://www.modaclouds.eu/publications/public-deliverables/
[19] https://code.google.com/p/line/
[20] http://home.deib.polimi.it/gibilisco/SPACE4Cloud/SPACE4Cloud.zip
[21] http://www.palladio-simulator.com/tools/source_code/
[22] http://forge.modelio.org/projects/modelio/files
[23] http://streamreasoning.org/download
[24] http://www.modaclouds.eu/software/Monitoring/
[25] https://bitbucket.org/mosaic
[26] https://github.com/Cloud4SOA/Cloud4SOA

D. Related Approaches in Service Quality Management

An extensive analysis of the state of the art was done in the frame of MODAClouds' public deliverables[18], relative to the modeling of costs and benefits (D2.2), the Cloud service modelling (D4.1), the QoS and performance modelling at design time (D5.1), and the Cloud monitoring techniques, QoS management at run-time and automatic deployment in Cloud environments (D6.1). We mention here only the most significant and closest initiatives.

The Service Level Agreement (SLA) is widely used today for managing QoS in Cloud environments. A QoS attribute in a SLA describes a specific measurable aspect of the quality of service. QoS management steps are discussed, e.g., in [15]. The technologies of interest for SLAs and QoS specifications in MODAClouds are SLAang[27] and QML [16]. SLA@SOI[28] framework is based on a SLA-enabling reference architecture suitable for both new and existing service-oriented systems and Cloud infrastructures. It consists of a suite of open-source software and components to implement SLA-aware solutions. It covers most of the levels of the service provisioning process; however it is not standards-compliant and defines a proprietary SLA stack [17]. Later on, Cloud4SOA[26] adapted the SLA@SOI framework for PaaS environments. Similarly, in mOSAIC[25], SLA@SOI concept was used at IaaS level. SLAs in Contrail[29] project are used to specify QoS for performance, availability, and quality of protection (QoP) for security guarantees for infrastructure components. The starting point is again the SLA@SOI framework, and SLA template creation, browsing and querying, SLA execution planning and adjustment were implemented. The PaaSage project[30] is providing novel techniques to evaluate the distributed application deployments in Multi-Clouds. The most recent project, SPECS[31] aims to enhance the security definition in SLAs, to monitor the SLA compliance and to ensure its enforcement. Despite the fact that all of these initiatives are exposing incipient forms of monitoring systems and feedback mechanisms, none of the them have currently comprehensive technical solutions for QoS management as MODAClouds currently provides.

IV. Conclusions and Future Work

The complexity of a Multi-Clouds support system, delivered as service or software, is tackled by various approaches. Ironically, these solutions intending to hide the heterogeneity of Cloud services, are heterogeneous. However, their requirements to cover certain service consumer are the same. The degree of these requirements fulfillment is different. In this paper we intended to identify the basic requirements as well as various challenges in satisfying them.

An uncovered segment in what concerns the portability requirement in Multi-Clouds is related to the domain-specific languages. Dealing with the long cycle from the high level language for service management and governance to the infrastructure service control is a complex task and multiple tools need to be build. The MODAClouds approach demonstrates that model-driven engineering techniques can provide a feasible solution to this complex task. A collection of tool sets ensuring service quality management and Cloud service agnosticism were presented shortly in this paper. The proof-of-the-concept implementation shows that the model-driven approach is applicable to the control of the service quality. The Cloud agnostic models can help Cloud application developers and providers to optimize their activities.

The implementation of MODAClouds concepts, from MODACloudML to the runtime platform is not yet finalized. In order to close the loop between the design and run-time to ensure a certain QoS further components must be developed. Moreover, the number of QoS parameters that are monitored should be increased.

[27] http://uclslang.sourceforge.net/

[28] http://www.sourceforge.net/apps/trac/sla-at-soi/

[29] http://contrail-project.eu/sla

[30] http://www.paasage.eu/

[31] http://specs-project.eu

Acknowledgment

The work reported in this paper is partially funded by the grants EC-FP7-ICT-2011-8-318484-MODAClouds (§III) and RO-PN-II-ID-PCE-2011-3-0260-AMICAS (§II).

References

[1] N. Grozev and R. Buyya, "Inter-cloud architectures and application brokering: taxonomy and survey," *Software: Practice and Experience*, 2012.

[2] D. Petcu, "Consuming resources and services from multiple clouds," *Journal of Grid Computing*, pp. 1–25, 2014.

[3] ——, "Multi-cloud: Expectations and current approaches," in *2013 MultiCloud*, 2013, pp. 1–6.

[4] R. Teckelmann, C. Reich, and A. Sulistio, "Mapping of cloud standards to the taxonomy of interoperability in iaas," in *3rd IEEE CloudCom*, 2011, pp. 522–526.

[5] D. Ardagna, E. Di Nitto, G. Casale, D. Petcu, P. Mohagheghi, S. Mosser, P. Matthews, A. Gericke, C. Ballagny, F. D'Andria, S. Nechifor, and C. Sheridan, "Modaclouds: A model-driven approach for the design and execution of applications on multiple clouds," in *4th MiSE*, 2012, pp. 50–56.

[6] N. Ferry, F. Chauvel, A. Rossini, B. Morin, and A. Solberg, "Managing multi-cloud systems with cloudmf," in *NordiCloud*, 2013, pp. 38–45.

[7] N. Ferry, A. Rossini, F. Chauvel, B. Morin, and A. Solberg, "Towards model-driven provisioning, deployment, monitoring, and adaptation of multi-cloud systems," in *6th IEEE CLOUD*, 2013, pp. 887–894.

[8] M. Miglierina, G. P. Gibilisco, D. Ardagna, and E. Di Nitto, "Model based control for multi-cloud applications," in *MiSE*, 2013, pp. 37–43.

[9] E. Di Nitto, M. Almeida, D. Ardagna, G. Casale, C. D. Craciun, N. Ferry, V. Muntes, and A. Solberg, "Supporting the development and operation of multi-cloud applications: The modaclouds approach," in *15th SYNASC*, 2013.

[10] A. Omerovic, V. Muntes-Mulero, P. Matthews, and A. Gunka, "Towards a method for decision support in multi-cloud environments," in *4th CLOUD COMPUTING*. ThinkMind, 2013, pp. 162–180.

[11] D. Franceschelli, D. Ardagna, M. Ciavotta, and E. Di Nitto, "Space4cloud: A tool for system performance and costevaluation of cloud systems," in *2013 MultiCloud*, 2013, pp. 27–34.

[12] M. A. Almeida, A. Abhervé, and A. Sadovykh, "From the desktop to the multi-clouds: The case of modeliosaas," in *15th SYNASC*, 2013.

[13] J. Perez and G. Casale, "Assessing sla compliance from palladio component models," in *15th SYNASC*, 2013.

[14] P. Bar, R. Benfredj, J. Marks, D. Ulevinov, B. Wozniak, G. Casale, and W. J. Knottenbelt, "Towards a monitoring feedback loop for cloud applications," in *2013 MultiCloud*, 2013, pp. 43–44.

[15] A. Dastjerdi and R. Buyya, "A taxonomy of qos management and service selection methodologies for cloud computing," in *Cloud Computing*, L. W. et al, Ed. CRC Press, 2011.

[16] S. Becker, "Quality of service modeling language," in *Dependability Metrics*, ser. LNCS, vol. 4909. Springer, 2008, pp. 43–47.

[17] P. Wieder, J. Butler, W. Theilmann, and R. Yahyapour, Eds., *Service Level Agreements for Cloud Computing*. Springer, 2011.