On MODAClouds' Approach for Application Design and Execution on Multi-clouds

Elisabetta Di Nitto[1], Arnor Solberg[2] and Dana Petcu[3]

[1]Politecnico di Milano, Italy
[2]SINTEF, Norway
[3]Institute e-Austria Timişoara and West University of Timişoara, Romania

Abstract. The design, deployment and control of new applications in Cloud environments still requires a deep understanding of the underneath technologies. Moreover, the interfaces and tools that are provided by the resource owners or third parties are highly coupled to the various exposed services to a level that hinders their easy adoption. Abstraction layers in form of models, semantic repositories or uniform interfaces are therefore build today. In particular the adoption of the model-driven engineering concepts has recently been proved to be beneficial for Cloud computing. In this paper we present the latest achievements of an initiative which has adopted a model-driven oriented approach for the design and execution of Cloud applications.

1 Introduction

Model-driven development combined with model-driven risk analysis have been recently introduced to Cloud computing application cycle. The vision of the marriage of model-driven architecture and Cloud computing includes the application developers that will be able to specify in a vendor agnostic manner the models of Cloud services in which they are interested, as well as to be able to enrich these models with quality parameters. Moreover, to close the current gap between the design and run-time stages of an application, both performing quality predictions, as well as 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 vision can be applied in practice is the main aim of a recently started research initiative, the MODAClouds project[4]. MOdel-Driven Approach for design and execution of applications on multiple Clouds (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 quality of service (QoS). The project is partially funded by the European Commission during October 2012 until September 2015 (ten companies and research institutions are participating).

MODAClouds concept and a motivating scenario were presented for the first time in [1], while the solution architecture was detailed in [2]. Several recent papers (complete

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

Di Nitto E., Solberg A. and Petcu D.
On MODACloudsâĂŹ Approach for Application Design and Execution on Multi-clouds.
DOI: 10.5220/0006183100490060
In *European Project Space on Information and Communication Systems (EPS Barcelona 2014)*, pages 49-60
ISBN: 978-989-758-034-5

list available at [5]) provided details about particular solutions and components. This paper intends to provide an overview of the MODAClouds concept and its implementation status at the half way through of the project duration.

The paper is organized as follows. Section 2 is describing shortly the state-of-the-art in model-driven Cloud engineering. Section 3 is describing the MODAClouds' approach and the implementation status. Section 4 is providing some hints on the next steps to be followed.

2 First Steps Towards a Model-driven Cloud Engineering

Cloud computing is a computing model enabling ubiquitous network access to a shared and virtualised pool of computing capabilities (e.g. network, storage, processing, and memory) that can be rapidly provisioned. However this conceptual model is implemented by a large number of service providers in very different ways. The current software stacks of Cloud services are heterogeneous and the provided features that are often incompatible between different service providers. This diversity is an obstacle with respect to demands such as promoting interoperability and preventing vendor lock-in.

The model-driven approach, often summarized as model once, generate anywhere, is particularly relevant when it comes to provisioning and deployment of applications and services across multiple Clouds, as well as migrating the source code and data from one service provider to another. The model-driven engineering (MDE) approach allows the developers to build the system at various level of abstractions. In the Cloud context, three levels are envisioned: CCIM, the Cloud enabled Computation Independent Model to describe an application and its data; CPIM, the Cloud-Provider Independent Model to describe Cloud concerns related to the application in a Cloud agnostic way; and CSPM, the Provider Specific Model to describe the Cloud concerns needed to deploy and provision the application on a specific Cloud.

CCIM describes Cloud applications at the service level. Hence, for a given application it contains the description of the services that compose it, the public interfaces of each service, the business processes that describe their orchestration and the domain model of the data exchanged by them through their public interfaces. Services Oriented Architecture related technologies which define applications as sets of services that communicate through well-defined interfaces can be used to define Cloud-enabled applications without going into the fine details of deployment. The time services are usually modeled by means of general purpose languages such as UML. Service-specific languages have also been designed for SOA (e.g. SoaML[6]). USDL language [3] goes even further, by allowing designers to specify, beside services and their interfaces, non-functional aspects on these services (e.g. pricing, legal, certification, documentation), however not Cloud specific. Other approaches are related to the specific concept of Web Service: WSDL language[7] which enable the specification of a list of services, interfaces, data types and orchestration processes at a syntactical level, or OWL[8] as semantic Web

[5] http://www.modaclouds.eu/publications/papers/
[6] http://www.omg.org/spec/SoaML/1.0/Beta2/
[7] http://www.w3.org/TR/wsdl
[8] http://www.w3.org/Submission/2004/SUBM-OWL-S-20041122/

language which enable the specification of the semantics of the services, besides their syntax. The main drawback of these approaches is that they do not allow for the description of non-functional requirements and constraints. However, at the modeling level, the OMG UML profile for QoS, QFTP[9], allows a designer to specify QoS requirements and to connect them to service descriptions. Reusing and extending such language is part of the scope of MODAClouds.

Modeling concepts and technologies for the provisioning, deployment and adaptation of applications in the Cloud (CPIM/CPSM) has been done until recently using the uniform interfaces provided by various libraries for application development, deployment and control at run-time. We can mention here the most successful ones: jclouds[10], libcloud[11], δ-cloud[12] or fog[13]. For example the jclouds model includes: description of node with metadata such as CPU, RAM, security policy; an abstract representation of a node with parameters such as minCPU, OS type; group of nodes to be managed together; set of command to be executed on nodes; information about the provider. Most of these libraries are language-dependent providing a common access to multiple Clouds. Typlically they provide provide a code-based model of the infrastructure and do not offer any mechanism for automatic provisioning and deployment of applications on the Clouds. Moreover, working at the infrastructure-as-a-service level, applications and services are not modeled.

Recently developed frameworks for managing Multi-Clouds that provide capabilities for the provisioning, deployment, monitoring, and adaptation without being language-dependent, have appeared (a survey is evalaible in [4]). We mention here three of them: Cloudify[14], Scalr[15] and CloudFoundry[16]. For example, the Cloudify model for deploying applications includes: a recipe for information like required infrastructure and how it should be used; the cluster of service instances that make up an application tier; the configuration (including provisioning and scaling rules) of an application and the services it is made of; the set of services working together; probes used to monitor the status of the system. These frameworks are important to optimise performance, availability, and cost of Multi-Cloud systems. However, they do not come with any structured approach, and the provided methods and tools are at a technical level, thus, the developer will typically be left hacking at code level rather than engineering Multi-Cloud systems following a structured tool supported methodology.

The models@runtime [5] paradigm proposes to leverage models during the execution of adaptive software systems to monitor and control the way they adapt. This approach enables the continuous evolution of the system with no strict boundaries between design-time and runtime activities. Models@runtime provide an abstract representation of the running system causally connected to the underlying state of the system

[9] http://www.omg.org/spec/QFTP/
[10] http://jclouds.apache.org
[11] http://libcloud.apache.org
[12] http://deltacloud.apache.org
[13] http://fog.io
[14] http://www.cloudify.org
[15] http://scalr.com
[16] http://www.cloudfoundry.org

which facilitates reasoning, simulation and enactment of adaptation actions. A change in the running system is automatically reflected in a model of the current system. A modification applied to this model can be enacted on the running system on demand. Thanks to the use of models, well-defined interface are provided to monitor the system and adapt it. The models also provide a way to measure the importance of changes in the system and analyse the delay before their enactment on the running system. Note that the libraries and frameworks described above do not provide such level of abstraction.

The concepts manipulated by the above mentioned solutions are serving as a basis for the definition of MODACloudML models and metamodels. The usage of models@runtime concept is promoted by MODAClouds in order to tame the complexity of adaptation and ease the reasoning process for self-adaptation.

3 MODACloud's Concepts, Approach and Implementation Status

The MODAClouds approach enables users to (a) specify service-independent models enriched with quality parameters, (b) implement these models, (c) perform quality prediction, (d) monitor applications at runtime, and (e) optimise them based on the feedback (filling the gap between design and runtime). More specifically, MODAClouds solution targets system developers and operators by providing them with supporting tools for the following software system life-cycle phases:

Feasibility Study and Analysis of Alternatives: a special tool enabling developers to analyze and compare various Cloud solutions.

Design, Implementation and Deployment. MODAClouds IDE supports a Cloud-agnostic design of software systems, the semi-automatic translation of design artefacts into code, and their deployment on the selected target Clouds.

Runtime Monitoring and Adaptation. The runtime layer (i) enables system operators in overseeing the execution of the system on multiple Clouds; (ii) automatically triggers some adaptation actions (e.g., migrate some system components from one IaaS to another that offers better performances at that time); and (iii) provides runtime information to the design-time environment that can inform the software system evolution process

The expected stakeholders involved in interaction with the MODAClouds solutions are:

Feasibility Study Engineer: in charge of analysing the high-level requirements for Cloud-based applications and of identifying an initial set of risks and costs deriving from the adoption of Cloud technologies;

Application Developer: takes care of defining all functional aspects of the Cloud-based application;

QoS Engineer: in charge of describing QoS constraints and of analyzing whether and how these constraints can be fulfilled given the available set of potential providers;

Application Provider: given the definition of the design information concerning the application, orchestrates the execution of the application deployment on the selected Clouds;

Cloud App Admin: oversees the correct operation of the Cloud application.

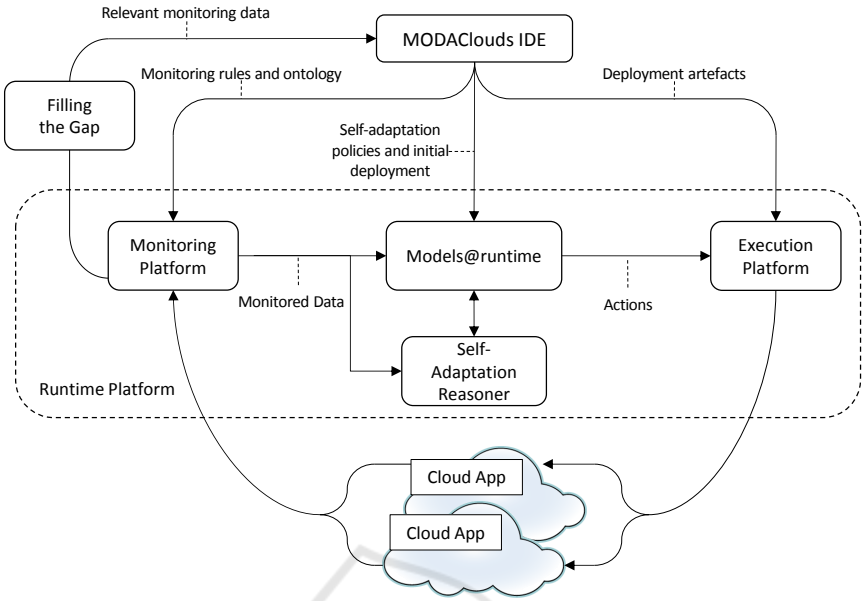

Fig. 1. High level architecture of MODAClouds.

The high level architecture of MODAClouds consists of two distinct software levels: the Runtime platform and the MODAClouds IDE (Figure 1).

The main component of the Runtime platform are the Monitoring Platform, the Self-Adaptation Reasoner, the Models@runtime engine, and the Execution Platform:

Monitoring Platform: is responsible for collecting data from the running application and from the underlying infrastructure, for analyzing them and for defining summary measures such as application health, QoS, etc. This platform executes monitoring rules that define the QoS constraints to be checked (together with some frequency of the check) and the actions to be executed in case some constraints are violated. Such actions can either trigger the self-adaptation Reasoner or results in the activation of additional monitoring rules. Other components subscribe to monitored data generated by the monitoring platform.

Self-adaptation Reasoner: implements the decision-making process responsible for identifying suitable changes in the current configuration of the Cloud application to satisfy the QoS constraints. The self-adaptation reasoner is activated by the monitoring platform in response to specific monitorid events and operate on the application through the Models@runtime APIs. Note that, the initial deployment decisions, provided by the MODAClouds IDE, are used to initialize this component.

Models@runtime Engine: is responsible for maintaining an updated version of the application model developed at design-time by interacting with the monitoring platform. This model is then used by the Self-adaptation Reasoner to build adaptation actions and by the execution platform to retrieve new deployment configurations. The second important task of this component consists in controlling the underlying Execution Platform.

Execution Platform: provisions all services that are needed to deploy applications on Multi-Cloud environment including the supporting services such as the Monitoring platform. The initial deployment is provided by the IDE through the Models@runtime component and translated to the target platform. The execution platform also offers the services and the API, needed to maintain and change the operational state of the deployment at runtime, typically triggered by of the Self-adaptation Reasoner.

The MODAClouds IDE offers the functions to support the development life-cycle of a Cloud-based application, from the early assessment of risks and costs to the actual deployment. The IDE supports the functional and non-functional design of the application at the three levels of abstraction (CCIM, CPIM, CPSM). Moreover, it supports the identification of potential Cloud candidates and the analysis of non-functional characteristics of the application. As a result of the analysis, the IDE is also able to propose optimized architectural configurations for the application. The IDE is built out of the integration among six components:

Decision Support System: supports the Feasibility Study Engineer in identifying the main risks and advantages in adopting specific Cloud solutions and in determining a first estimate of costs associated to these solutions. The estimated cost will then be refined by the QoS modelling and analysis tool.

MODACloudML Functional Modelling Environment: supports the modelling of Cloud applications and of the data they manipulate. It supports the CCIM, CPIM and CPSM models and the generation of the deployment artefacts from them.

QoS Modelling and Analysis Tool: allows the QoS Engineer to perform the following operations: modeling of QoS requirements; quality prediction and analysis of QoS requirements fulfillment; generation of monitoring rules; definition of the goals and constraints driving the execution of self-adaptation.

Data Mapping Component: is able to analyze the data that are going to be manipulated by the applications and the constraints on them (and on their storage and retrieval). The result of such analysis should allow the user to decide upon the best Cloud specific data representation formats and tools that fulfill the application requirements.

MODACloudML Deployment and Provisioning Component: supports the model specification of deployment and resource provisioning including deployment and provisioning constraints, and interfaces with the MODAClouds runtime components in order to effectively deploy the Cloud application. The deployment and provisioning model exploits at runtime to enact adaptation of the application using Models@runtime technologies [6].

Filling the Gap Design-time Manager: is responsible for retrieving information from the runtime platform where the Cloud application is deployed. This information is used for two main purposes: first, to update the parameters of the design-time models, allowing these to better capture the actual behaviour of the application; and second, to provide the Cloud App Admin and the QoS Engineer with information about the behaviour of the application at runtime.

The shared models (between the above six components of the IDE) are organized in the three abstraction layers (CCIM, CPIM, and CPSM) as shown in Figure 3. We

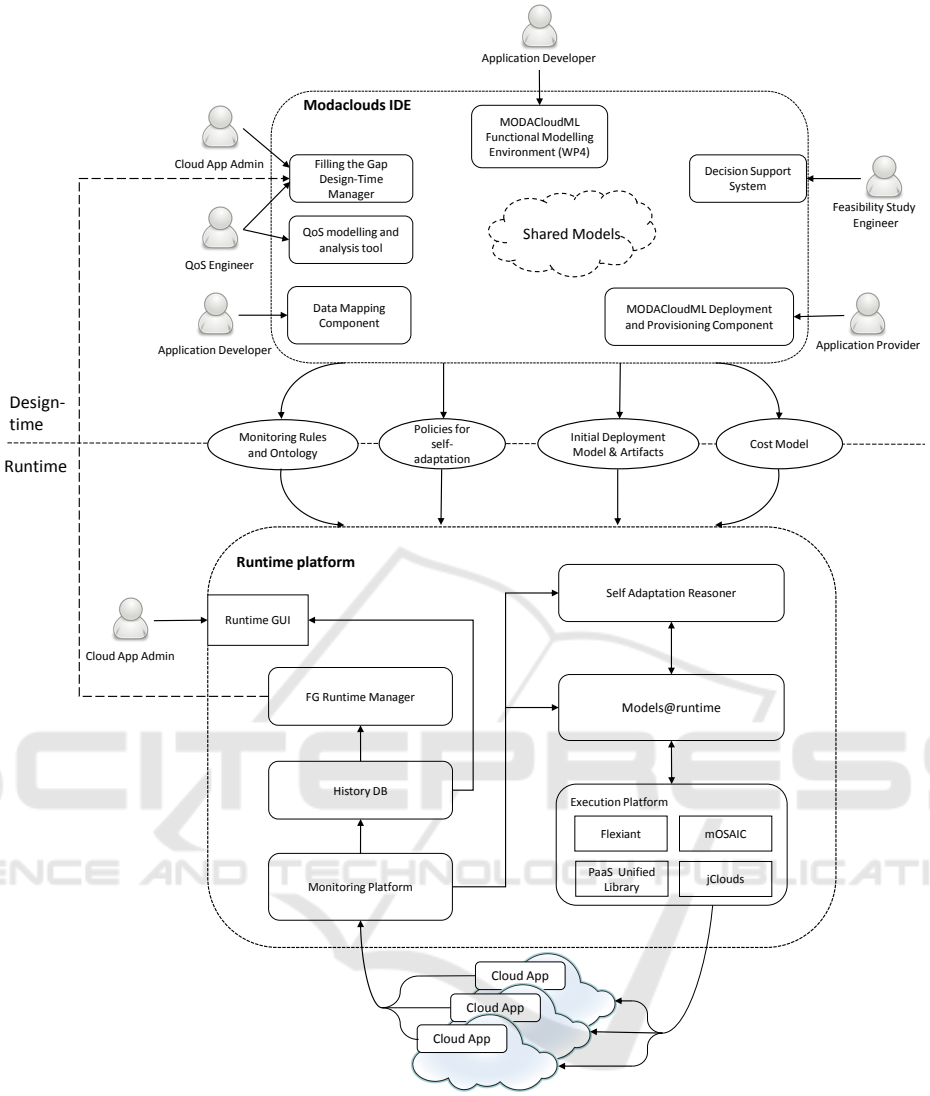

Fig. 2. Detailed architecture of MODAClouds.

take here only the example of data models to ilustrate the differences encountered at the three abstraction layers:

CCIM Data Model: describes the main data structures associated to the software to be. It is expressed in terms of a typical Entity Relationship diagram and enriched by a meta-model that specify functional and non-functional data properties.

CPIM Data Model: describes the data model in terms of models being typically offered by Cloud providers, without referring to a particular one. Since the majority of available data storage software provides services that include distributed file

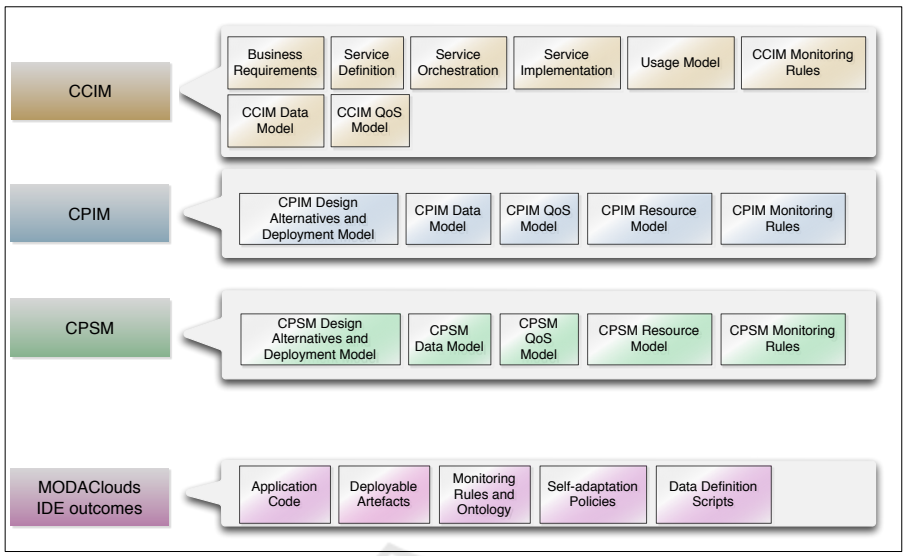

Fig. 3. Shared meta-models.

system, NoSQL solutions, and blobs, these three models are considered at the CPIM level that are suitable to represent these different cases. They are: Graph Data Model, Hierarchical Data Model and Flat Data Model.

CSPM Data Model: describes data structures implemented in the Cloud providers. The following families are considered: Relational Data Model family, Hierarchical Data Model family, Flat Data Model family, Key-Value Data Model, Column-based Data Model family.

The application development workflow based on MODAClouds IDE follows the approach depicted in Figure 4. Four stakeholders are considered, namely the Feasibility Study Engineer, the Application Developer, the QoS Engineer and the Application Provider. These actors play different roles in the development of the application interacting with specific tools, represented by blue circles, the interactions between the tools is performed by a set of shared models, represented by white rectangles. The design phase ends with the generation of some artefacts, represented by gray rectangles, that can be used to deploy and manage the application.

The development process starts at the CCIM level with the Application Developer building a functional model of the application using the MODACloudML Functional Modelling Environment. The Application Developer is also in charge of creating the data model with the Data Mapping tool. The architectural description of the application is enriched by the Feasibility Study Engineer defining performing a feasibility study with the help of the Decision Support System, the result of this action provides guidelines to the QoS engineer that adds information useful to predict the performance of the designed application with the help of the QoS Modelling tool. These actors work in an agile way by re-iterating over the shared models adding incrementally their contribution to the model of the application.

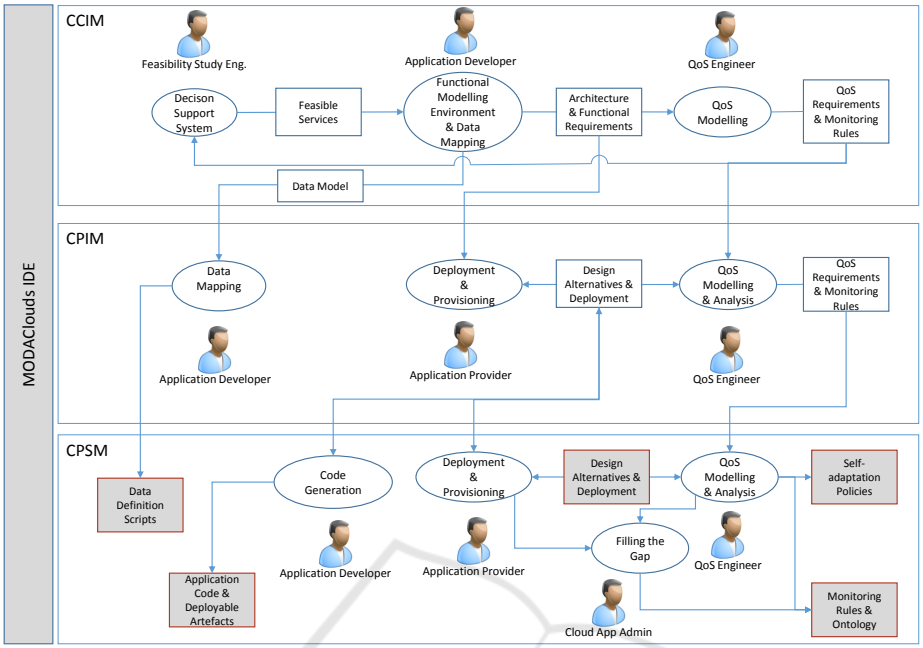

Fig. 4. IDE workflow.

The Application Developer and the QoS engineer also add functional and non-functional requirements related to the architecture and the behaviour of the application, from these requirements an initial set of monitoring rules can be derived.

This set of initial models is then refined by adding some Cloud-dependent information. This process is performed by the Application Developer that refines the Data Model and by the cooperation of the QoS engineer and the Application Provider. These two actors interact with the MODACloudML Deployment and Provisioning Component and the QoS Modelling and Analysis tool, respectively. The interactions between the two actors is aimed at providing an initial deployment model that maps the functional components modelled at the CCIM level to Cloud resources. Note that at this abstraction level no information on specific Cloud providers is available. When the deployment model has been built, the QoS engineer can refine the non-functional constraints specified at the CCIM level or build new ones. The monitoring rules generated at the previous level can now be refined. The last modeling phase is aimed at adding Cloud provider specific information to the previously built models and derive artefacts used to develop and deploy the application. The Data Mapping component generates a set of Data Definition Scripts specific to the data storage technology and the Cloud provider. The application developer makes use of the code generation capability of the MODACloudML functional modeling environment to derive the structure of the application code. The Application provider and the QoS engineer continue their interaction in order to define the final Design Alternatives and Deployment model. This model is used by the QoS engineer to map the CPIM non-functional requirements and derive the final Self-adaptation Policies. The QoS Engineer and the Cloud Application Adminis-

Table 1. MODAClouds' components availability at the half time of the project.

Component	Status	Public deliverables
Monitoring Platform	Final release	D6.3.2
Self-Adaptation Reasoner	Under construction	D6.4
Models@runtime engine	Proof of concept	-
Execution Platform	Initial release	D6.5.1
Decision Support System	Proof of concept	D2.3.1
MODACloudML Functional Modeling Env.	Proof of concept	D4.3.1
QoS modeling and analysis tool	Initial release	D5.2.1
Data Mapping Component	Under construction	D6.6
MODACloudML Deployment & Provisioning	Proof of concept	D4.3.1
Filling the Gap Design-Time Manager	Initial release	D5.3.1
Integrated solution	Proof of concept	D3.5.1

trator interact with the Filling the Gap Design-Time Manager to configure the Filling the Gap Analysis, which is translated in a set of Monitoring Rules, making use of the QoS Model and the Deployment Model.

The artefacts generated at design-time by the different tools composing the IDE are deployed into the runtime platform that takes care of managing the application execution.

The status of the component implementation is exposed in Table 1. The Dx.y.z from the last column are refering to user-guides and codes available at [17, 18]). Note that most components are available already as open-source codes and can be used as stand-alone tools.

4 Conclusions and Future Steps

The MODAClouds was designed to support application designers in the development of a Multi-Cloud application and operational personnel in the execution of such application.

Compared with other current approaches for the model-driven engineering of Cloud applications, the following main novel ideas are proved by MODAClouds to be viable:

At Design Time. MODAClouds toolset allows a development team to design and implement an application in a cloud-agnostic way and supports the deployment of such application on a multi-cloud environment. Moreover, it allows designers to express QoS requirements and to run proper analyses to identify the best configuration of the application on a set of suitable Clouds. Furthermore, the tools are integrated through the sharing of a set of common metamodels. This integration approach enables an asynchronous interoperability between the various tools and allows for an easy extension of the toolset. It allows therefore to deliver, promote and adopt each of the design-time tools belonging to the MODAClouds solution separately from the others.

[17] http://www.modaclouds.eu/publications/public-deliverables/
[18] http://www.modaclouds.eu/software/

At Run Time. At runtime MODAClouds focuses on three main issues: the management of the execution, intended as the set of operations to instantiate, run, stop services on the Cloud, the monitoring of the running application and the self-adaptation of the application to ensure the fulfillment of the QoS goals. The corresponding components interact based on the Models@runtime engine. This engine keeps alive a model of the system and uses it to control all management operations on the actual execution platforms. This model is updated according to the results gathered by the monitoring platform and is modified as a result of a self-adaptation action. The innovation points of this runtime platform concern the highly configurable and scalable monitoring platform, the live model of the application that is suitable for supporting the decisions concerning self-adaptation, as well as the sophisticated optimization approaches to support self-adaptation process.

Moreover, MODAClouds offers the possibility (to be exploited in near future) for the runtime to provide data back to the design-time (feedback loop) enabling continuous design of the Cloud application and its deployment configuration and provisioning on a Multi-Cloud infrastructure. Its main advantage concerns the possibility for the design time tools to learn from previous execution experiences and to use them both to optimize the configuration of the currently operated application and to improve the analyses on applications to be developed in the future.

Acknowledgement

The work reported in this paper is partially funded by the grant EC-FP7-ICT-2011-8-318484 (MODAClouds). The paper resumes the main concepts and approaches detailed in public deliverable D3.2.2 of the project.

The list of this paper author do not reflect all the contributors to the project and the above mentioned deliverables. We express our gratitude to all the project members who have make this paper possible.

References

1. Ardagna D., Di Nitto, E., Casale, G., Petcu, D., Mohagheghi, P., Mosser, S., Matthews, P., Gericke, A., Ballagny, C., D'Andria, F., Nechifor, C.S., Sheridan, C.: MODACLOUDS: A Model-Driven Approach for the Design and Execution of Applications on Multiple Clouds. In: Procs. 2012 ICSE Workshop on Modeling in Software Engineering (MISE), doi: 10.1109/MISE.2012.6226014, IEEE (2012) 50–56
2. Di Nitto, E., Almeida Da Silva, M.A., Ardagna, D., Casale, G.; Craciun, C.D., Ferry, N., Muntes, V., Solberg, A.: Supporting the Development and Operation of Multi-cloud Applications: The MODAClouds Approach, In: Procs. 15th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), doi: 10.1109/SYNASC.2013.61, IEEE (2013) 417–423
3. Cardoso, J., Barros, A.P., May N., Kylau, U.: Towards a Unified Service Description Language for the Internet of Services: Requirements and First Developments. In: 2010 IEEE International Conference on Services Computing (SCC), doi: 10.1109/SCC.2010.93, IEEE (2010) 602–609

4. Petcu, D., Consuming Resources and Services from Multiple Clouds Journal of Grid Computing, doi: 10.1007/s10723-013-9290-3, Springer (2014)
5. Blair, G., Bencomo N., France, R.: Models@run.time. Computer 42 (10), doi: 10.1109/MC.2009.326, IEEE (2009), 22–27
6. Ferry, N., Rossini, A., Chauvel, F., Morin, B., Solberg, A.: Towards Model-Driven Provisioning, Deployment, Monitoring, and Adaptation of Multi-cloud Systems. 2013 IEEE Sixth International Conference on Cloud Computing (CLOUD), doi: 10.1109/CLOUD.2013.133, IEEE (2013), 887–894.