Chapter 12
Modelio Project Management Server Constellation

Antonin Abhervé and Marcos Almeida

12.1 Introduction

SOFTEAM is a French middle-sized company that provides the Modelio modelling tool. Modelio.[1] is an enterprise-level open source modelling solution delivering functionality for business, software and infrastructure architects. It is a comprehensive MDE workbench tool supporting the UML2.x standard. Modelio provides a csentral IDE which allows various languages (represented as UML profiles) to be combined in the same model. Modelio proposes various extension modules, enabling the customization of this MDE environment for different purposes and stakeholders.

The **Team Work Manager** is SOFTEAM's solution to team collaboration in Modelio. It allows Modelio users, after a minimal software and hardware investment, to efficiently share and work together on models stored in a central repository accessible in a local network or in the Internet. It automates version control and configuration management, making sure every developer has access to the last version of the shared model and works on a uniform configuration. From the point of view of the developer, a repository is divided into Projects, which contain: Model elements, Extension modules used by the user and Configuration information. A repository needs to be installed, configured and maintained by the users in private machines. A SVN repository may store different projects and different teams may work in the same repository. Developers use the Modelio desktop client to access a central repository on a SVN like workflow: committing modifications to model elements,

[1] http://www.modelio.org.

A. Abhervé (✉) · M. Almeida
Softeam Cadextan, 21 Avenue Victor Hugo, 75016 Paris, France
e-mail: antonin.abherve@softeam.fr

M. Almeida
e-mail: marcos.almeida@softeam.fr

E. Di Nitto et al. (eds.), *Model-Driven Development and Operation of Multi-Cloud Applications*, PoliMI SpringerBriefs,
DOI 10.1007/978-3-319-46031-4_12

receiving updates from other users and using merges/locks to deal with concurrent work.

By its participation on the MODAClouds project, SOFTEAM intended to move its modelling services to the Cloud in order to relieve the burden for our clients in supporting the necessary infrastructure. During the MODAClouds project, we developed a new version of this tool called **Constellation** [1, 2]. This service is based on a Service-Oriented Architecture under which the TeamWork Manager is provided as a service on the Cloud. By the beginning of the third year of the project we started providing commercial services based on Constellation.

We hope that the "potentially infinite" resources available on the Cloud will make tasks such as scaling the servers of a project up and out and moving between different Cloud providers very easy to our customers. Additionally, activities such as monitoring and adapting the installation hopefully will be able to be executed without specialized knowledge in systems administration.

The MODAClouds provided features have an important role in fulfilling these objectives. As we are going to present in the following sections, the role of MODAClouds in Constellation is two-fold. At design time, MODAClouds should support design and implementation in a Cloud provider independent way, reducing development costs, and increasing its flexibility. At run time, it should support the monitoring and adaptation of the application to support its desired QoS levels.

This chapter is organised as follows. Section 12.2 presents the proposed architecture of Constellation. Section 12.3 presents how we used MODAClouds components in building our case study. Finally, Sect. 12.4 presents our conclusions.

12.2 Proposed Architecture

In order to simplify this migration, the architecture of our Cloud solution relies on the implementation of a component called **Administration Server** (Fig. 12.1). The Administration Server allows clients to create and manage user accounts, define roles, and create modelling projects and associate users and roles to specific projects. The Administration Server is designed as a JEE application which provides a web accessible user interface support implemented with Java Server Faces 2 and service behaviour supported by Entity Java Beans components. This application is linked with an relational database to ensure persistency of application data.

The Administration Server can provision computing resources in order to maintain the established level of quality of service. Cloud Services managed by a Administration Server are delivered as Cloud-enabled applications. These applications are deployed on the provisioned Cloud resource. Once deployed in Cloud resources, services usually need to be configured and accessed by clients. The Administration Server needs to make sure that the necessary projects, users and permissions have been created and set up once a Cloud agent has been installed. Standard protocols are used for both activities. Web Services enable the deployed agents to be configured. Moreover, TCP/IP protocols will allow Modelio desktop based clients to connect to an agent, independently from which Cloud it has been deployed.

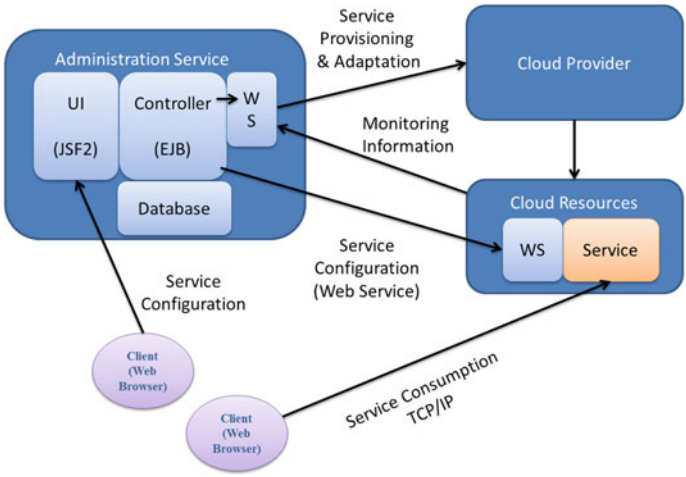

Fig. 12.1 The architecture of the administration server

External agents are independent applications that provide specific high resource consuming services to Prototype of Constellation. Agents can be deployed on demand on specific Cloud instances (IaaS or PaaS depending on their implementation). The number of deployed agents may change in real time depending on the application workload. Each agent implements a variable number of services called **Workers**, which are executed when an agent receives a command from the Administration server.

The only dependency of this design to the specific Cloud provider is the communication between the Administration Server and the Cloud provider in order to deploy, monitor and eventually migrate services. The actual code to interact with the Cloud provider is however encapsulated in a Web Service usually installed on the Administration Server. This Web Service translates actual requests from the user into specific requests to MODAClouds runtime components.

12.3 Use of MODAClouds Design and Runtime Components

12.3.1 Modelling with Creator 4Clouds

We used MODAClouds Creator 4Clouds Functional Modelling tool to describe the architecture of Constellarion's Administration Server along with its modelling services. We have also used this model as input to other design and runtime tools. During the first MODAClouds phase we considered two kinds of services: **SVN** and **HTTP**

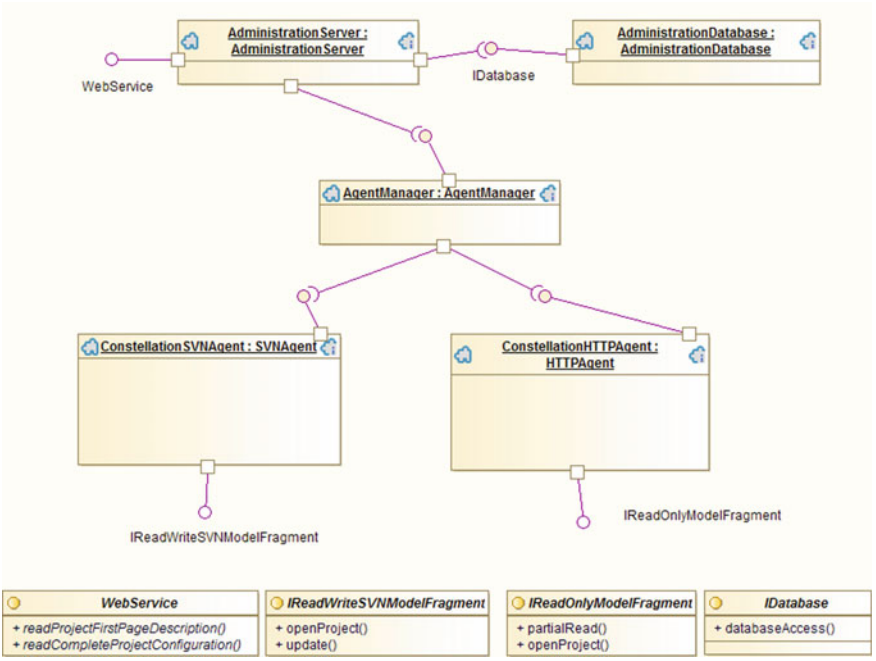

Fig. 12.2 Case study CCIM modelling on the IDE

fragments. The first one provides a read-write model that is edited collaboratively, while the second one provides read-only models that are shared among different teams.

Figure 12.2 depicts the functional architecture of Constellation specified with the MODAClouds IDE as a Cloud Computation Independent Model.

At the highest level, the CCIM shows the services that compose Constellation: the Administration Server and the Administration Database connected by an interface provided by the Administration Database and required by the Administration Server.

Still at the CCIM level, Fig. 12.3 shows the QoS constraints associated with the most important operations provided by the Constellation modelling services. For

	metric	unit	aggregationType	rangeMin	rangeMax
HTTPAgentReadModelAverage	ResponseTime	ms	Average		5000
HTTPAgentReadModelPercentile	ResponseTime	ms	Percentile(thPercentile=85)		12000

	metric	unit	aggregationType	rangeMin	rangeMax
SVNAgentReadModelAverage	ResponseTime	ms	Average		15000
SVNAgentReadModelPercentile	ResponseTime	ms	Percentile(thPercentile=85)		30000
SVNAgentWriteModelAverage	ResponseTime	ms	Average		60000
SVNAgentWriteModelPercentile	ResponseTime	ms	Percentile(thPercentile=85)		300000

Fig. 12.3 CCIM QoS constraints on MODAClouds IDE

SVN fragments, 15 s is the target average time for reading model modifications, and 60 s is the target average time for writes. This considers that users make large commits (i.e., containing a great number of model changes, and therefore expect to obtain large change sets when they update). For HTTP models, 5 s is the average time for reading parts of the model, considering that users make infrequent accesses to subparts of shared read-only models. Constraints on the 85th percentile are used to define acceptable upper bounds for response times. These are set to 12 s for HTTP reads, and to 30 s and 5 min for SVN reads and writes, respectively.

CPIM and CPSM models describe the deployment of the application at different levels of abstraction, first in a Cloud provider independent way, and then in a Cloud provider specific way. Figure 12.4 presents excerpts of the Constellation application model described in MODACloudML at the three levels of abstraction in order to illustrate the correspondence between the CCIM and the CPIM and CPSM models.

12.3.2 Multi-cloud Deployment with CloudML 4Clouds

The deployment model at CPIM level allows us to model the deployment of our application by identifying the various components of our application deployment.

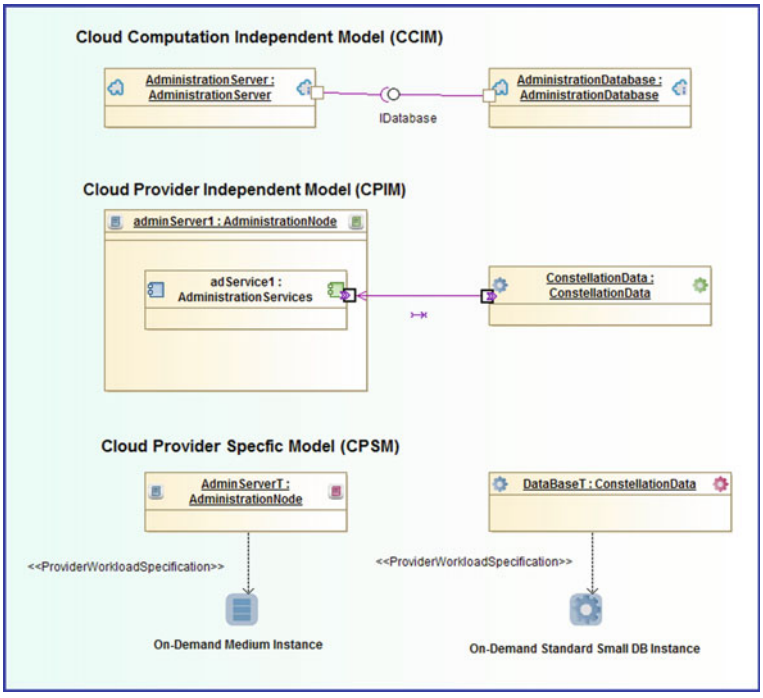

Fig. 12.4 Three levels in IDE

In this experiment, our efforts focused on better use of Cloud platforms through the integration of PaaS services and the migration to a multi-Cloud deployment solution. In a second step, we sought to take advantage of the support of multi-Cloud environments allowed by the MODAClouds project. We studied the best deployment configuration for our application and selected three Cloud providers: Amazon EC2, Flexiant and Amazon RDS.

Figure 12.5 describes the deployment of Constellation in a multi-Cloud context. It shows an Administration Server and two agents, both of them in IaaS Cloud nodes. The former in Amazon, the later in Flexiant. The database that stores administration data is stored on a PaaS database, provided by Amazon RDS.

This development brings the following benefits:

- Allows us to scale the compute and storage resources available to our database to meet Constellation needs.
- Provides the best reliability to our application with automated backups, DB snapshots and automatic host replacement capabilities.
- Provides predictable and consistent performance for I/O intensive transactional database workloads.

12.3.3 Cost and Performance Analysis with SPACE 4Clouds

As part of MODACloudsML CCIM models, we provided models of how users interact with Constellation, and of the performance of Constellation services when actually deployed on a virtual machines. We used SPACE4 Clouds to assess the costs and

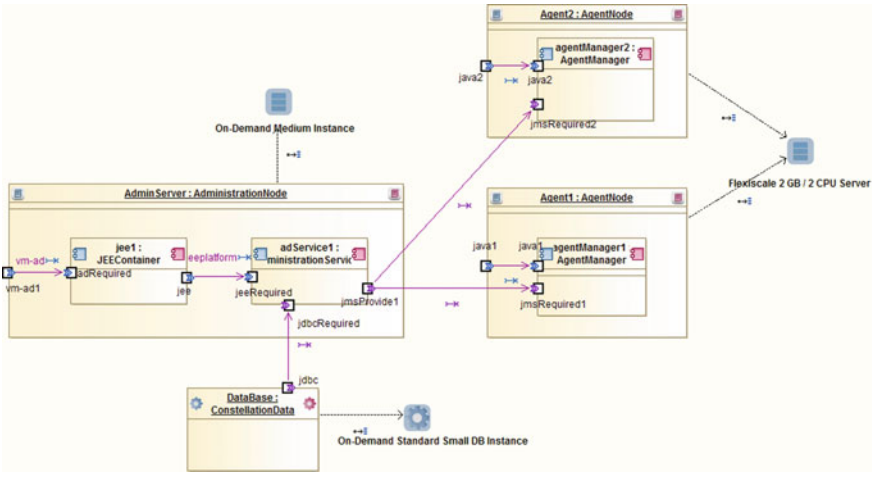

Fig. 12.5 Constellation deployment in multi-cloud environments

QoS the current architecture is able to provide on different Clouds, and in particular, the maximum number of clients we can serve with the modelled architecture.

In addition, we devised a trial architecture for a new modelling service called **Conference Service** to be implemented during the last year of the project, and compared its QoS characteristics with the one implemented in the first two years of the project. Differently from a SVN service, the conference service decouples the reading and writing load on the system in different VMs that can be load balanced and Cloud bursted independently. This is a typical example of advanced deployment configurations Constellation needs to support. Our experiments showed that the Conference Service is more scalable than the current solution.

The Fig. 12.6 presents the usage model of our users, obtained through observation of typical users. It considers users that connect to modelling through their full workday. Five percent of the time they interact with Constellation, they connect to an existing project, which is translated onto the sequence of calls we see on the top of the figure. Ten percent of the time, they read updates from an SVN model, seventy-five percent of the time they get data from HTTP fragments and ten percent of the time they perform SVN commits.

In addition to usage models, we provided models of user workload throughout the day (see the Fig. 12.7). We represented a typical business office workload, with most of it concentrated around commercial working hours (8–12 h and 13–17 h).

SPACE 4Clouds allowed us to discover the peak number of users supported by this architecture. Figure 12.8 shows the result of this analysis. We can see that the SVN service supports around 250–300 users without breaking QoS constraints, while the Conference service scales to almost the double number of users without breaking constraints.

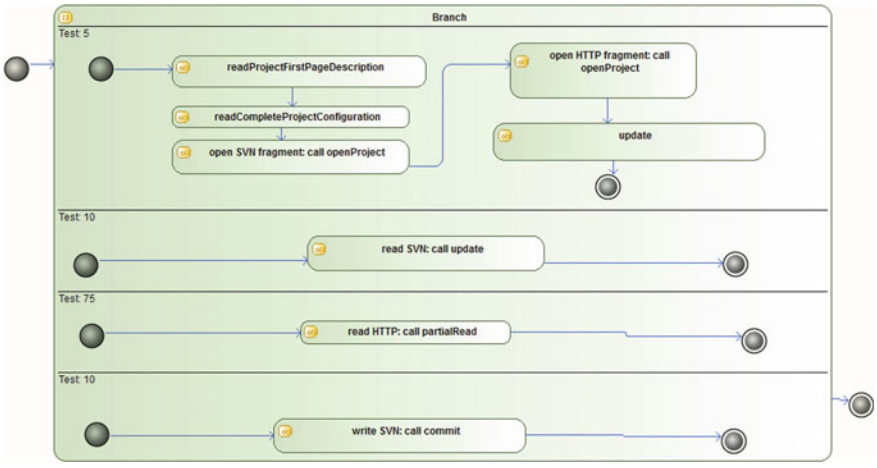

Fig. 12.6 Modelling constellation user's behavior

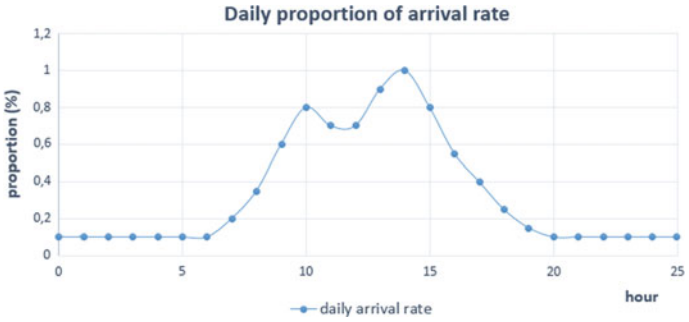

Fig. 12.7 Modelling constellation user's workload

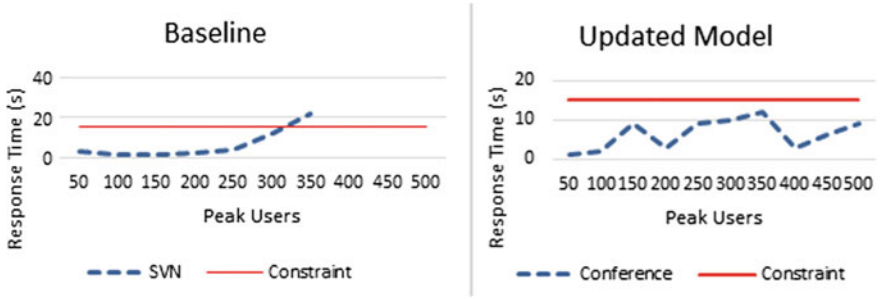

Fig. 12.8 Response time bottleneck estimations for SVN and conference services

12.3.4 Multi-cloud Monitoring and Management with Energizer 4Clouds

Energizer 4Clouds provides valuable services for our case study, such as the management of the execution, intended as the set of operations to instantiate; run and stop services on the Cloud; the monitoring of the running application and the self-adaptation of the application, to ensure the fulfilment of the QoS goals.

When defining the final design of the Constellation case study, we were interested in the best way to integrate the features provided by the platform into our application. In the context of the Constellation case study, we are interested in the integration of three aspects of Energizer 4Clouds: the monitoring platform, the self-adaptation platform and the execution platform. Figure 12.9 presents the deployment model of the Constellation case study including runtime platform components.

The Monitoring Platform allows us to monitor specific metrics collected from business components of our case study deployed on different Cloud platforms. To achieve this goal, we integrated five components into our architecture: three componentsfrom the monitoring platform and two components developed using the API

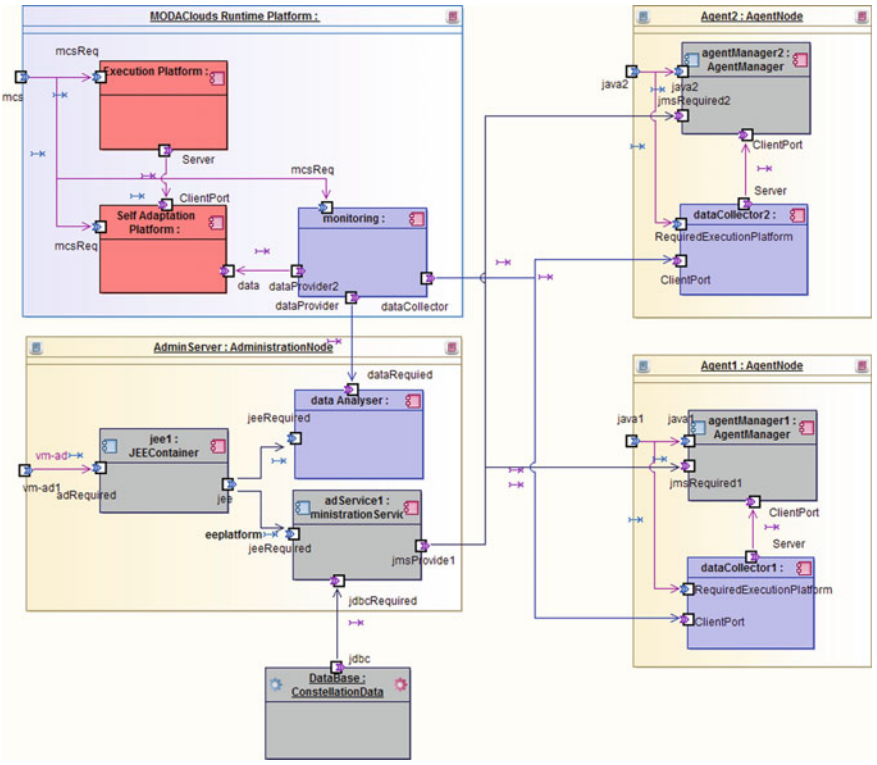

Fig. 12.9 MODAClouds runtime platform integration

provided by platform components. The role of these is to exploit monitoring data in our application.

To exploit the monitoring platform, we have integrated two components based on the API provided by the monitoring platform. These components ensure the intermediation between the monitoring platforms and business components of Constellation. They allowed us to implement a Cloud vendor independent agent monitoring user interface, and to integrate it to our commercial offering.

- **Constellation Data Collector**: To collect business metrics from Constellation agents, we integrated into our architecture this extension of the monitoring platform. Based on MODAClouds Data Collector API, this programme will collect data about CPU, RAMS and Access Disk of each process managed by agents.
- **Constellation Data Analyzer**: Based on the REST API of MODAClouds Monitoring Manage, Constellation will incorporate a component to analyse, store and display monitoring data according to a business point of view. This service will be integrated into the Administration Server.

12.4 Conclusion

Constellation can be presented as an advanced repository which stores the models defined using the Modelio CASE tool and which provides several high-time consuming services on the Cloud. Among its services, we find the creation of collaborative projects, the hosting of model fragments allowing teamwork, the management of a Model Library catalogue or monitoring services applied to all these elements.

In this chapter, we presented the final version of the Project Management Server, renamed, for commercial reasons to Constellation. The development of Constellation started with the beginning of the MODAClouds project and by the end of it we have a first version that started to be commercialized. The current commercial version of Constellation is restricted to deployment on customer premises. We are confident that, thanks to MODAClouds, its architecture is ready to the Cloud.

The Constellation case study integrated both design time and runtime components from MODAClouds in its design. At design time, MODAClouds supported the design of the architecture of the application, and its early QoS analysis, in order to identify bottlenecks. At runtime, MODAClouds supported the multi-Cloud deployment, management and monitoring of Constellation.

References

1. Almeida Da Silva MA, Abhervé A, Sadovykh A (2013) From the desktop to the multiclouds: the case of ModelioSaaS. In: 15th international symposium on symbolic and numeric algorithms for scientific computing (SYNASC), 23–26 Sep 2013, pp 462–472
2. Desfray P (2015) Model repositories at the enterprises and systems scale the Modelio constellation solution. In: 2015 3rd international conference on model-driven engineering and software development (MODELSWARD), Feb 2015, pp IS–15