SeaClouds Project

D5.2.1 - Design of the User Interface

Project Acronym	SeaClouds
Project Title	Seamless adaptive multi-cloud management of service-based applications
Call identifier	FP7-ICT-2012-10
Grant agreement no.	610531
Start Date	1st October 2013
Ending Date	31st March 2016

Work Package	WP5, Integration, infrastructure delivery and GUI
Deliverable code	D5.2.1
Deliverable Title	Design of the User Interface
Nature	Report
Dissemination Level	Public
Due Date:	M12
Submission Date:	10th October 2014
Version:	1.0
Status	Final
Author(s):	Francesco D'Andria, Román Sosa González (Atos); Jesus Gorroñogoitia Cruz (ATOS)
Reviewer(s)	Diego Perez (Polimi), Pengwei Wang (UPI)

Dissemination Level

Project co-funded by the European Commission within the Seventh Framework Programme		
	Public	X
	Restricted to other programme participants (including the Commission)	
	Restricted to a group specified by the consortium (including the Commission)	
	Confidential, only for members of the consortium (including the Commission)	

Table of Contents

List of Figures

List of Tables

Executive Summary

This deliverable presents the design of the User Interface for the SeaClouds platform.

In this deliverable, SeaClouds adopts a dashboard metaphor for the interface inspired by the direct manipulation principles and WIMP (Window, Icon, Menu, Pointing device) interfaces.

The SeaClouds User Interface will be focused in implementing an easy to use topology editor, something that traditionally was quite complicated. The utilization of Eclipse Winery, an existing Web-based environment to graphically model TOSCA topologies, is under evaluation.

1 Introduction

This deliverable has been inspired by the work already started from ATOS in the Cloud4SOA project [2].

In the context of the SeaClouds project, the deliverable describes the design of the SeaClouds User Interface (UI), which constitutes the uppermost layer in the SeaClouds architecture. The high-level specification of this interface was defined in D2.2, according to the requirements elicitation defined in D2.1.

A background on UI design is introduced in Section 2, where the models for User Interaction Design are presented, emphasizing the different models that comprise the Conceptual Model: Data Model, Interaction Model and Process Model.

The sequence diagrams corresponding to the Process Model are located in Annex A.

Section 3 presents the Interaction Model, exposing a set of good practices in the implementation of User Interfaces, and proposes a dashboard metaphor for the interface.

Section 4 explains the objectives of the SeaClouds interface, which is focused in developing an easy to use topology editor, and reviews existing projects with interfaces that include that kind of editor.

1.1 Glossary of Acronyms

Acronym	Definition
DM	Direct Manipulation
KPI	Key Performance Indicator
OSS	Open Source Software
SLA	Service Level Agreement
QoB	Quality of Business
QoS	Quality of Service
UI	User Interface
WIMP	Windows, Icons, Menus and Pointing

Table 1: Glossary of acronyms

2 User Interface Design

The concept of user interaction refers to the two-way communication in a human-computer interaction context. Users interact with a computer system via a user interface (UI).

Traditionally, designing and evaluating a good user interface, which is easy to use, easy to understand, meets the needs of users, and supports them in the tasks, requires a good understanding of the interaction between the user and the system as well as a good knowledge of the potential end users and of human communication skills [1] and [2].

Recently, the Web has provided a sort of standard for online user interfaces: web browsers have become a part of the computer literacy of most users, and they are also able to reproduce desktop-like interfaces online.

Therefore users have a common understanding and a shared expectation about how elements in a user interface should behave. This common background alleviates the task of interface design since the user should already be familiar with a particular interaction model, although this can be a limitation for very innovative designs that propose alternative models.

In this section we introduce the discipline of user interaction design and user interaction modelling, and we discuss how Web interfaces in general and SeaClouds interaction model in particular are positioned within those disciplines.

2.1 A Dynamic and Iterative Process

User interaction is a dynamic process in which available technology is exploited to improve the user-system interaction [3].

User interaction design is also an iterative process [4]. It involves different design cycles at different levels of detail such as understanding the user's needs, coming up with possible conceptual models, prototyping and evaluating them with respect to usability and user experience goals, and so on.

The iterative user interaction process is also important for achieving a good user-centered interaction design which both involves users throughout the design and development process [5] and requires the iteration of design solutions [6]. Understanding users' need is an ongoing activity and requires a great deal of attention not only in the beginning of the user interaction design process, but also in the whole process itself. This activity refers to the **User Model** which is an explicit representation of the properties of a particular user [7]. User Model includes the user's knowledge of both the tasks and the system while taking into account the users' needs, abilities and preferences.

User involvement into this iterative process occurs by using different levels of prototypes which are tested in the evaluation phase. Both **low**- and **high-fidelity prototyping** techniques play a significant role in evaluating user interaction of a system in the planning stages [7]:

- Design sketches and low-fidelity mock-ups are used to visualize the new system and the flow within this system before making any programming.
- High-fidelity prototypes use the software packages to construct the user interface prototype and have a look and feel because they are close to the final design.

This dynamic and iterative design process supports the evolution of an initial prototype toward a final system. In SeaClouds this will imply that we will evaluate the design at different stages of the development.

2.2 Models for User Interaction Design

Traditionally, an effective methodology for designing interactive systems should define four models [1] and [2]:

- cognitive model
- conceptual model
- structural model
- perceptual model

In the following we explain what these are and how they fit in the SeaClouds design.

2.2.1 The Cognitive Model

The cognitive model includes the analysis of the user's tasks, which leads to the other stages of the interaction design. A cognitive process starts by understanding users' needs which are represented in scenarios and use cases and goes on by evaluation of the steps required to perform a task. Task analysis evaluates the way in which people perform their tasks [5] and employs a user population performing those tasks to test a tool [9]. A cognitive model should focus on the users and tasks together, and include both user and task analyses in the design process in order to analyse users' preferences, tasks and cognitive abilities successfully.

In SeaClouds we focus on two types of users: application designers, who develop the applications to be deployed on the cloud; and application administrators, who decide about the planning and observe the performance of the application on the cloud. The tasks analysis was carried out by means of use cases derived by the SeaClouds scenarios (see D2.1; section 7).

2.2.2 The Conceptual Model

The conceptual model consists of organizing the results obtained from the cognitive model in order to describe how end users would build their own representation of the system [1].

Conceptual Models provide user interface designers with several benefits. They force designer to be more precise than using natural language. They help to understand the user interaction model as well as making consistency check easier because of the visualization of the process. Conceptual modelling of a user interface is achieved through establishing the

underlying structure of an interface [5]. Such structure is obtained by taking into account the functional requirements represented with use cases and mostly relying on brainstorming of designers and comparisons of alternatives [10].

The process of conceptual modelling an interface comprises a *data model*, a *process model* and an *interaction model*.

A **data model** refers to the data and relations between data and has a well-known data modelling techniques such as the Entity-Relationship (E-R) model [11].

In SeaClouds the most significant data structure is a meta-model that englobes the full application profile and the cloud profile (see D3.1; section 2).

A **process model** is used to fix the functionality that the system is supposed to provide. For the user interface, we defined UML sequence diagrams (see Annex A).

An **interaction model** can be seen as the final stage of a conceptual model. It can be defined as the set of principles, rules and properties which guide the design of an interface [12].

Unfortunately, there is not a widely accepted user interaction model in contrast with the data and process models. Nonetheless, '**best practices**' can be used to simplify and facilitate the process of user interaction modelling.

The interaction model is further discussed in the following chapter.

2.2.3 The Structural Model

The next step in UI design is the structural or architectural model which refers to software architecture and its implementation. The structural model aims to describe the functional elements and their relationships at the interface implementation stage. Since the software architecture will be the focus of D2.2, here it suffices to say that we chose to implement a modular user interface.

2.2.4 The Perceptual Model

The perceptual model describes how end users build their own representation of the system. Actually, the model is defined by conducting operator tests once a prototype or an intermediate version of the system has been implemented.

Ideally, the perceptual model is the same as the conceptual model. In that case, the users can anticipate the behaviour of the system and use it in an efficiently way [1].

On the other hand low- and high-fidelity prototypes can be used for the testing. As already mentioned, low-fidelity prototypes are used to visualize the new system and the flow within this system by using paper-based materials. High-fidelity prototypes use the software packages to construct the user interface prototype and help to visualize the final design.

Even if [1] recommends low-fidelity and high-fidelity prototyping for the evaluation of the design, other insights from the literature review as well as the practical usage emphasize on the significance of the prototyping at early stages of the UI design [8] and [5].

3 User Interaction Model

In the scope of user interface design, the most problematic issue of the system is modelling the interaction [3].

Designing the user-system interaction is a complex process because of the two facets of human-computer interaction: coping with technical constraints as well as human factors [1]. In general, a model of an interaction can be seen as a way to understand and improve the usability of the interface [13].

The model indicates the transaction across the system boundary and describes how to combine interaction methods in a meaningful and consistent way while achieving the look and feel of the interaction from the users' perspective.

The goal of this chapter is to define the User Interaction Model. We adopt the following working definition [12]:

"An interaction model is a set of principles, rules and properties that guide the design of an interface. It describes how to combine interaction techniques in a meaningful and consistent way and defines the "look and feel" of the interaction from the user's perspective. Properties of the interaction model can be used to evaluate specific interaction designs."

In this chapter we will introduce the general metaphor for our model which will provide the general context for the "*set of principles, rules and properties*" from the definition above.

3.1 Direct Manipulation Interfaces

When we look at how a user enters her input with an interface, there are several interaction models, such as command line, menu selection, form-fill, direct manipulation and anthropomorphic [5].

Direct manipulation (DM), as the generic interaction model, is the most used model in user interaction design. DM interfaces enable users to interact directly with the user interface objects and to execute some actions such as dragging and dropping these objects in a user interface. A DM interface has several advantages [12]:

- Presents the concepts visually: the user can see the objects and act on them directly. In DM interactions the objects can be manipulated by physical actions such as clicking or dragging and therefore there is little syntax to remember. In other words, users are allowed to interact with objects in a more natural and familiar way. DM interfaces use icons and metaphors, and this helps users to develop a mental model easily.
- Easy to learn: even novice users can learn the functionality of this kind of interfaces easily. Experienced users also can work rapidly on a wide range of tasks.
- Easy to remember how to use: because objects are visually displayed, even the infrequent users do not need to remember hidden commands.

- Avoids errors and allows easy recovery from errors if they occur: in DM interfaces, operations are rapid, incremental and reversible. The user can easily see if her actions can accomplish her goals or not. If not, she can simply change the direction of the activity. This also implies fewer error messages and therefore less stress on the user.
- Encourages exploration: because the user has a direct involvement with objects rather than communicating with an intermediary agent, she feels confidence and in control and therefore can predict the system's response.

However, DM interfaces require graphic displays and present the danger of misrepresenting icons and metaphors for different user groups. Therefore, in order to avoid misinterpretation, those icons and metaphors should be evaluated in the evaluation stage of the interaction and in case revised.

3.1.1 Direct Manipulation principles

DM has the following principles [12]:

1. Continuous Representation of objects of interest: object of interests should be visible or easily accessible at all time.
2. Physical actions on objects vs. complex syntax: DM privileges the former over the latter, being the former more immediate and easy to understand.
3. Fast, incremental and reversible operations with an immediately and apparent effect on the objects of interest: short temporal distance between an action and its results, visible results with possibility to correct errors early in the process.
4. Layered or spiral approach to learning: easy transition from beginner to power user.

3.2 WIMP Interfaces

WIMP (Windows, Icons, Menus and Pointing) interfaces are the most common interfaces since they are well established, having being introduced at Xerox in 1982. Designers find it faster and easier to stick with a small set of well understood techniques. Similarly, developers find it more efficient to take advantage of the extensive support for WIMP interaction provided by current development tools [12]. WIMP interfaces are used mostly for desktop applications, but currently some Web development environments such as JavaScript Ext JS are capable of replicating most of desktop application interface functionality directly within the browser (see for example http://www.sencha.com/examples/desktop.html). Therefore, design guidelines thought for the desktop can be applied to the web, and this is thanks to the technological advances.

Particularly, since SeaClouds interface needs to support user interaction and asynchronous display of events such as monitoring, a **dashboard metaphor** seems to be appropriate.

Web dashboards are designed to layout large amounts of information into a single page so end-users can easily find, retrieve, understand and process all that information at a glance. The information is normally pre-processed (e.g., classified, filtered, etc.) before being

displayed, whereby end-users can easily manage it. For instance, information is structured in a pyramid hierarchical way, so top information (and supporting operations) includes summary reports (for instance, SeaClouds summaries on deployed applications), while bottom information contains low-level details (for instance, performance data about a particular deployed application).

An effective dashboard design would emphasize those areas end-users care most about. Hence, dashboard layouts should change depending on users' role. This encourages a modular design. Depending on users' role, only top level information may be displayed in the dashboard widgets, while low-level information and operations may be hidden.

In the context of Management Information Systems, key benefits of the dashboard design are: visual representation of performance measures, easy identification and correction of negative trends, inefficiencies and efficiencies measurements, detailed reporting, ability to execute human based and machine based decisions, real-time analysis of system functioning, etc.

3.3 Design Elements

Web dashboards use different visual elements (e.g., controls) and layouts to render collected information and offer supporting operations concerning that information. Control and layout elements that appear frequently are:

- Boolean state indicators: representing on/off state information. Rendered with text, icons, check/radio buttons, etc.
- Multi-state indicators: representing discrete multi-state information, e.g., application life cycle (started, maintenance, stopped). Rendered with text, combo-boxes, etc.
- Continuum state indicators: representing continuum multi-state information. Rendered with bar-graph, gauges, etc.
- Information connections, which relate rendered information with further available ones. Rendered with links (e.g., anchors), buttons, etc.
- Information supporting operations, which operate upon displayed data. Rendered with buttons, icons, menu entries, etc.
- Panels, intended to display together similar information and related operations.
- Tab panels, intended to distribute large amount of information among topics.
- 2D/3D data visualization controls, intended for representing visual information: pictures, performance graphs, content types, etc. Rendered as icons, pictures, etc.
- Specialised widgets, rendering calendars, lists, tables, and menus.

Well design existing dashboards in general follow common dashboard design principles, some of which are referenced in the literature [14]:

- Avoid scrolling information. Keep useful data visible at once (this is analogous to the first principle of Direct Manipulation.
- Give context to data, required to understand information in its correct context.
- Avoid too detailed information. Provide just high level information to give quick overview.
- Choose the right measure and display to render the data (for instance, render long structure data in a chart bar, pie, etc., rather than in table).
- Avoid meaningless colour coding.
- Highlight important data.

4 SeaClouds Usage Scenario and Review of Existing Interfaces

In the SeaClouds project, we are going to develop a user interface to interact with all the components that have been implemented in the platform, in order to facilitate the use of the software capabilities.

But the intention is to move away from the full integrated management application, focusing on innovative aspects. In our case, we will focus on the Application/Cloud Resource orchestration, making easy something that traditionally was quite complicated.

Therefore, the development of the User Interface will produce a set of modules, each one interacting with a different SeaClouds capability: Application Topology Design, Management, SLA, Monitoring, etc. The implementation of these modules will prioritize the functionality, leaving the goal of obtaining a uniform look&feel as a less important topic.

While each capability in the SeaClouds platform should be supported by the User Interface, in the context of this document we would like to give special emphasis to the Design of the Application Topology (an early version of the User Interface of other capabilities of SeaClouds are described in [19])

In fact during the redaction of this document, the consortium is still analysing and evaluating the existing technologies that support the desired behaviour.

The proposed GUI follows the same concept and approach of modern web applications, which try to put together most of the functionality of the application on a main view, while retaining an easy and intuitive usage. Some examples are as follows:

- www.websequencediagrams.com
- www.gitter.im
- www.draw.io

Actually, there are some initiatives with this approach in the field of designing cloud applications. For example, **OpenStack** [17], one of the most prominent free and open-source cloud computing software platform, is developing its interface following the same approach through Curvature by Cisco[18].

Curvature is an intuitive graphical user interface, designed to take advantage of the OpenStack Quantum service to provide the user with a powerful visualisation tool for designing virtual application topologies Figure 1.

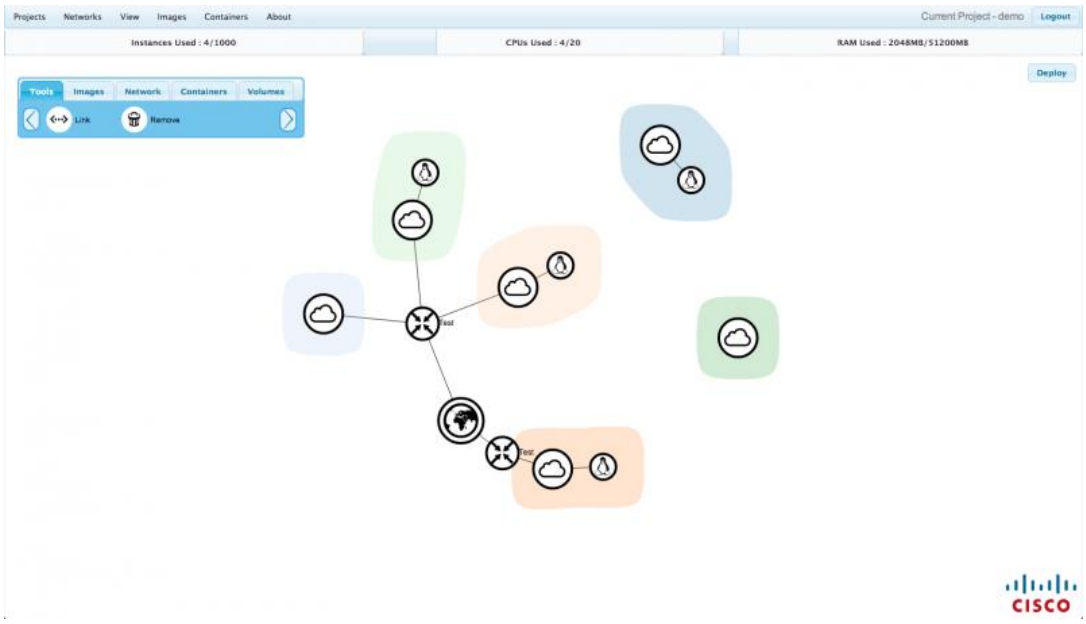

Figure 1: Example of the Curvature Dashboard

Another example is Juju [16], an open source service orchestration management tool developed by Canonical Ltd. Juju allows software to be deployed, integrated and scaled on a wide choice of cloud services or servers.

The Juju GUI allows non-technical users to create complex software stacks via drag-and-drop Figure 2.

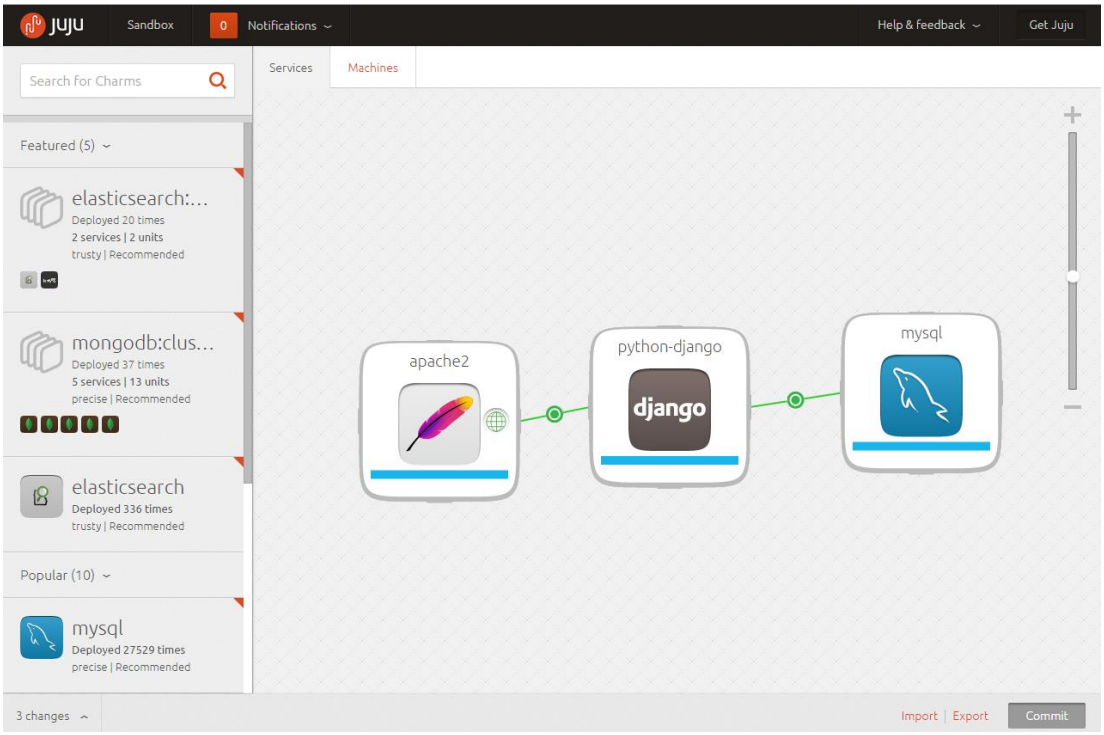

Figure 2: Example of Juju Dashboard

The idea behind these two projects relies on providing a main screen where the user creates and manipulates graphs of nodes. In the context of SeaClouds, a node is an application component, and an arc between two nodes is a relation of dependency from one node to the other one. Each node has properties that describe functional and non-functional requirements: QoS, Technology, Cost etc.

To design a complex application, the application provider starts with a blank screen, where she adds modules and sets its properties. From one node, a line can be drawn to another node in order to represent a relationship between the two nodes. With a finished design, further actions may be executed.

The actions on the graph are context-dependent. If the application has not been deployed, the available operations may be only modify the design or deploy the application. Once the application has been deployed, a set of new operations are then available: view monitoring metrics, view SLA status, manage the application, etc.

Finally, since the consortium decided to leverage TOSCA for the definition of the Application Topology, we are evaluating the chance of adopting an already existing TOSCA solution like Winery [15].

Winery is a Web-based environment to graphically model TOSCA topologies and plans managing these topologies. The project is very interesting for SeaClouds, because of its visual topology editor, which produces the designed topology as a TOSCA specification.

The adoption of Winery involves an effort in order to adapt it to the requirements of SeaClouds project. The needed effort is being evaluated by the consortium. The main issues that Winery presents are:

- Some features are not stable yet,
- Integration issues

The other possibility is to implement the Topology Design UI from scratch. This possibility entails more coding efforts, but the integration risks are less than using Winery.

5 Conclusions

In this deliverable we described the design of the User Interface we plan to develop in SeaClouds project. The starting point was the high-level specification as described in D2.2, according to the requirements elicitation defined in D2.1.

According to the good practices exposed when evaluating the Interaction Model, SeaClouds adopts a dashboard metaphor for the User Interface, focusing in developing an easy to use topology editor.

During the writing of this deliverable, emerged the possibility of using Winery, a graphical topology editor that generates TOSCA specification files, and which can save a lot of coding efforts. The utilization of this software is still being evaluated.

Annex A. Sequence Diagrams

The following sequence diagrams show the flow of interaction of the use cases:

Initial Planning

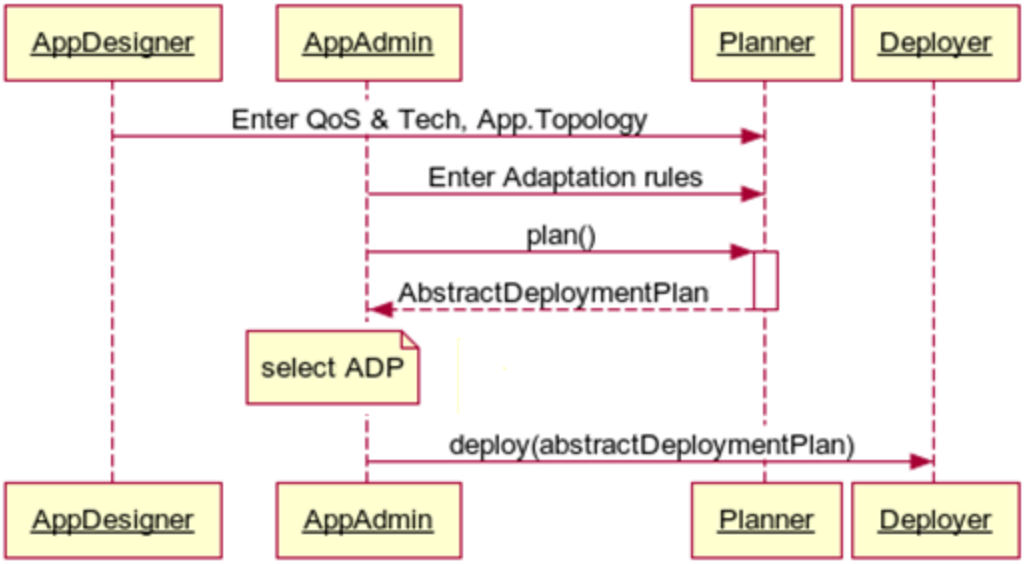

Figure 3: Initial planning

The planner needs the following inputs from the users Figure 3:

- Application Designer: QoS, technology requirements and the application topology.
- Application Administrator: Adaptation rules.

Using these inputs and the discovered capabilities and SLAs coming from the Discoverer component, the planner generates an Abstract Deployment Plan describes the feasible distribution of application modules onto available clouds, and satisfies all the QoS properties and technology requirements required by the Application Designer.

The abstract plan and related SLAs are returned to the Application Administrator, to support his decision.

Once the Application Administrator decides in favour of using the abstract deployment plan, it is then passed to the Deployer, which will instantiate a concrete plan to actually deploy the application modules.

Replanning

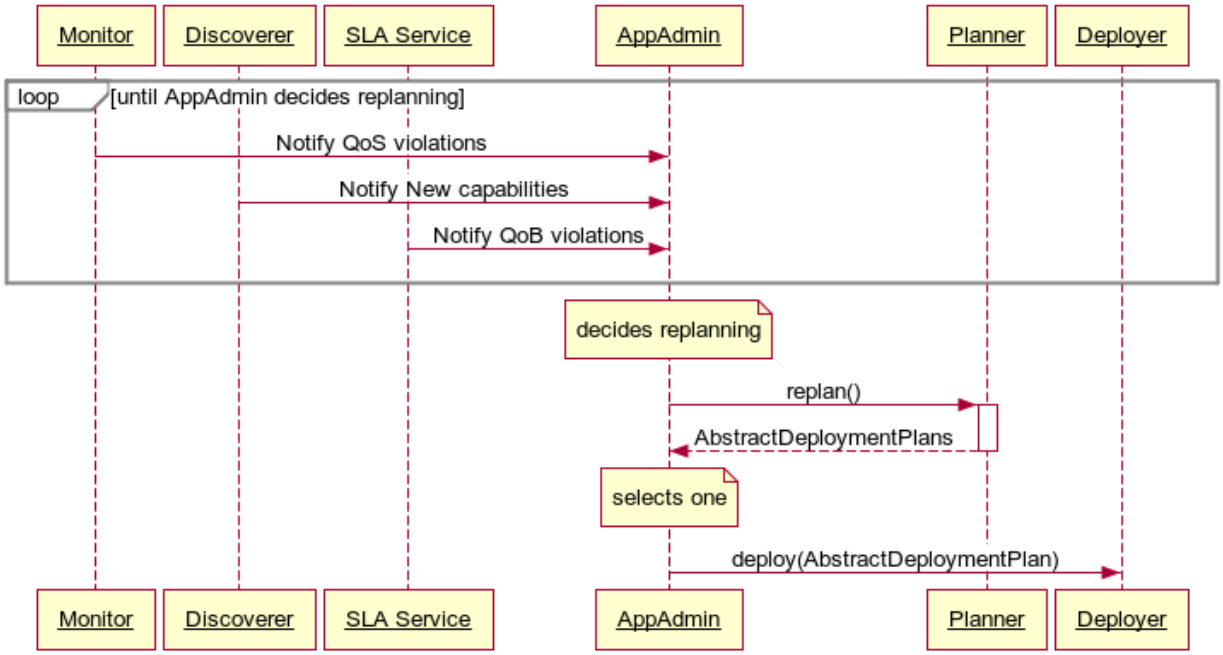

Figure 4: Replanning

During runtime, the Application Administrator will receive notifications from Figure 4:

- the Monitoring Manager, notifying any QoS violations,
- the SLA Service, notifying about QoB violations,
- the Discoverer component, notifying new cloud capabilities discovered by the component.

As shown in Figure **4**, with this information, the Application Administrator can decide whether to accept to do replanning. If so, the replanning trigger will be passed to the planner. Then, the planner tries to do replanning, generates a new set of abstract deployment plans and passes them to the Application Administrator again.

As in the Initial Planning case, the Application Administrator selects an abstract deployment plan, and passes it to the Deployer, which will instantiate a concrete plan to redeploy the application modules.

Update module artefacts

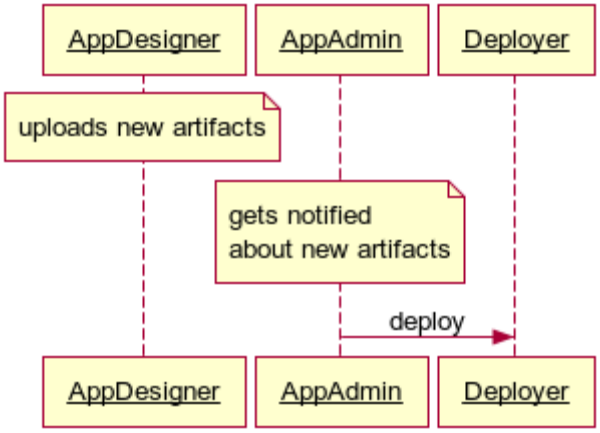

Figure 5: Update module artifacts

The deployed application needs to be updated as new features are added and new bugs are fixed. The Application Designer must select the module/s to update and upload the new artifact/s. Once the module/s has been uploaded, the application is ready to redeploy. Figure 5 depicts these interactions.

Manage application lifecycle

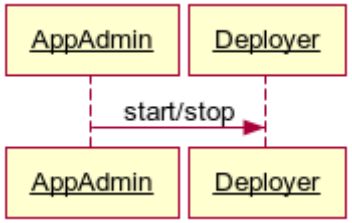

Figure 6: Manage application lifecycle

At any time, during runtime, the Application Administration can decide to stop the application (e.g. for a maintenance process), and later resume the execution of the application.

When he decides it, the desired signal is sent to the Deployer (as shown in Figure **6**), which actually performs the operation.

View real-time monitoring metrics

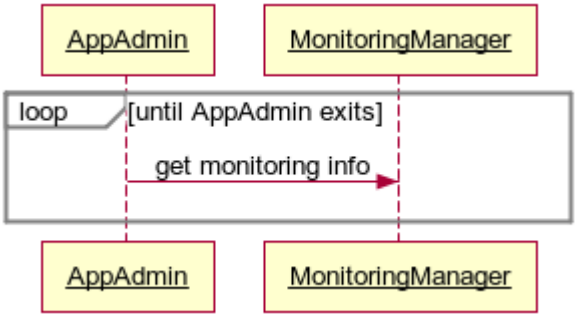

Figure 7: View real-time metrics

A very interesting feature is the ability to show the several metrics being observed by the Monitoring Manager.

There are two approaches in this view:

- show real-time metric data, querying the Monitoring Manager with a reasonable frequency the retrieve the new samples,
- show historic metric data, querying the Monitoring Archive the desired time interval.

The view should show a set of metrics related to the application and a set of time ranges. Once the user has selected the desired metric and time range, the selection is used to query the Monitoring Manager and plot and plot on the screen periodically, as shown in Figure **7**.

More complex graphs can be plotted if we consider several metrics or applications at the same time.

View SLA status

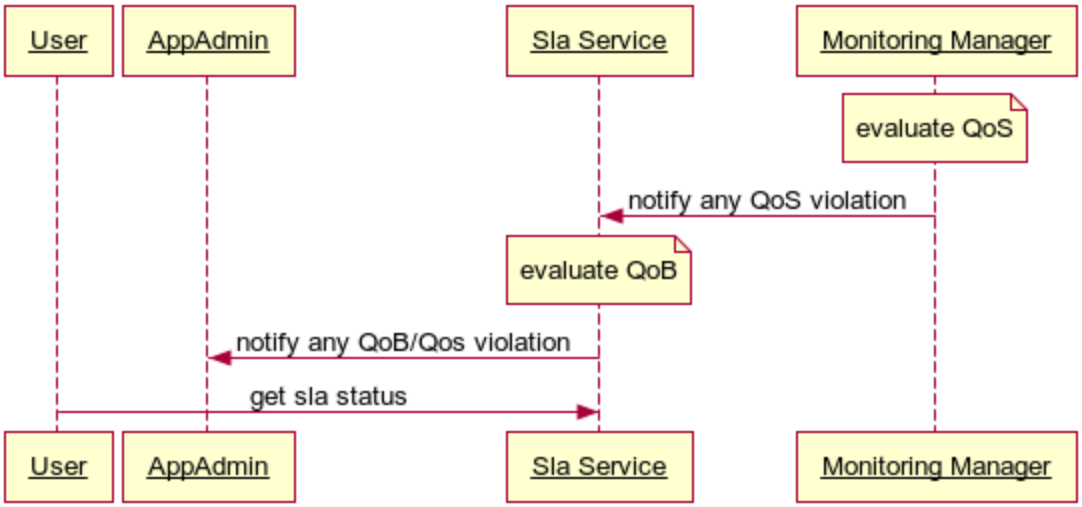

Figure 8: View SLA status

During runtime, the cloud provider must fulfill the constraints that were specified by the Application Designer as QoS. Any QoS constraint not satisfied produces a QoS violation, which is notified to the SLA Service to evaluate the associated QoB rules. A QoB rule not satisfied produces a QoB violation Figure 8.

The Application Designer or the Application Administrator may at any time retrieve the current status of the SLA. The information to be shown is:

- The status of the agreement: fulfilled/non fulfilled,
- The status of individual guarantee terms: fulfilled/non fulfilled,
- QoS violations,
- QoB violations,
- Overall penalization applied.

This information may be displayed as a simple list. More complex visualizations with graphs can be achieved if the results are grouped by application module, date, etc.

References

[1]. M. Beaudouin-Lafon, “An overview of human-computer interaction,” Biochimie, vol. 75, no. 5, pp. 321 - 329, 1993.

[2]. Stefano Bocconi, Cyntelix and Yosu Gorroñogoitia (ATOS), Cloud4SOA D3.1 “User Interaction Model”

[3]. G. Lee, C. M. Eastman, T. Taunk, and C.-H. Ho, “Usability principles and best practices for the user interface design of complex 3D architectural design and engineering tools,” International Journal Human-Computer Studies, vol. 68, no. 1-2, pp. 90-104, 2010.

[4]. Y. Rogers, H. Sharp, and J. Preece, Interaction design: Beyond human-computer interaction. John Wiley & Sons, Inc., 2002.

[5]. D. Stone, C. Jarrett, M. Woodroffe, and S. Minocha, User Interface Design and Evaluation. Elsevier, 2005, p. 704.

[6]. ISO 9241-210, Ergonomics of human-system interaction -- Part 210: Human-centred design for interactive systems. 2010, p. 32.

[7]. L. Strachan, J. Anderson, M. Sneesby, and M. Evans, “Minimalist User Modelling in a Complex Commercial Software System,” User Modeling and User-Adapted Interaction, vol. 10, no. 2-3, p. 109--146, 2000.

[8]. S. Oviatt, “User-centered modeling and evaluation of multimodal interfaces,” in IEEE User-centered modeling and evaluation of multimodal interfaces, 2003, pp. 1457 - 1468.

[9]. K. Höök, “Steps to take before intelligent user interfaces become real,” Interacting with Computers, vol. 12, no. 4, pp. 409 - 426, 2000.

[10]. X. Li, M. Gunal, and J.-Y. Shiau, “Computational Modeling for Improving Usability Design Workflow,” 2009 IEEE International Conference on Networking Sensing and Control ICNSC 2009 March 26 2009 March 29 2009, pp. 679-684, 2009.

[11]. S. España, J. I. Panach, I. Pederiva, and Ó. Pastor, “Towards a holistic conceptual modelling-based software development process,” in *25th International Conference on Conceptual Modeling (ER2006)*, 2006, pp. 437 - 450.

[12]. M. Beaudouin-Lafon, “Instrumental interaction: an interaction model for designing post-WIMP user interfaces,” in *Proceedings of the SIGCHI conference on Human factors in computing systems*, 2000, pp. 446 - 453.

[13]. D. Benyon and D. Murray, “Adaptive systems: from intelligent tutoring to autonomous agents,” *Knowledge-Based Systems*, vol. 6, no. 4, pp. 197-219, Dec. 1993.

[14]. S. Few, *Information Dashboard Design*. O’Reilly, 2006, p. 211

[15]. Winery https://projects.eclipse.org/projects/soa.winery.

[16]. Juju https://jujucharms.com/

[17]. OpenStack http://www.openstack.org/

[18]. Curvature CISCO http://docwiki.cisco.com/wiki/Curvature_for_OpenStack

[19]. SeaClouds Deliverable D4.2 “Cloud Application Programming Interface”: http://www.seaclouds-project.eu/deliverables.html