Using a Lifecycle Model for Developing and Executing Real-Time Online Applications on Clouds

Dominik Meiländer[1], Antonio Bucchiarone[2], Cinzia Cappiello[3], Elisabetta Di Nitto[3], and Sergei Gorlatch[1]

[1] University of Muenster, Germany
[2] Fondazione Bruno Kessler, Italy
[3] Politecnico di Milano, Italy
d.meil@uni-muenster.de

Abstract. We describe how the generic Lifecycle Model developed in the S-Cube project for the design and management of service-based applications (SBA) can be utilized in the context of Cloud Computing. In particular, we focus on the fact that the Infrastructure-as-a-Service approach enables the development of Real-Time Online Interactive Applications (ROIA), which include multi-player online computer games, interactive e-learning and training applications and high-performance simulations in virtual environments. We illustrate how the Lifecycle Model expresses the major design and execution aspects of ROIA on Clouds by addressing the specific characteristics of ROIA: a large number of concurrent users connected to a single application instance, enforcement of Quality of Service (QoS) parameters, adaptivity to changing loads, and frequent real-time interactions between users and services. We describe how our novel resource management system RTF-RMS implements concrete mechanisms that support the developer in designing adaptable ROIA on Clouds according to the different phases of the Lifecycle Model. Our experimental results demonstrate the influence of the proposed adaptation mechanisms on the application performance.

1 Introduction

Service-oriented applications are developed for constantly changing environments with the expectation that they will evolve over time. Several service-oriented system engineering (SOSE) methodologies have been proposed aiming at providing methods and (sometimes) tools for researchers and practitioners to engineer service-oriented systems. SOSE methodologies are more complex than traditional software engineering (TSE) methodologies: the additional complexity results mainly from open world assumptions, co-existence of many stakeholders with conflicting requirements and the demand for adaptable systems. A number of service lifecycle models have been proposed by both industry and academia. However, none of the proposed models has either reached a sufficient level of maturity or been able to fully express the aspects peculiar to SOSE. Within the S-Cube project [1] a new Lifecycle Model was designed that combines existing techniques and methodologies from TSE and SOSE to improve the process through which service-based applications will be developed.

This paper extends our previous work [2] on studying how the S-Cube Lifecycle Model can be applied on the emerging and challenging domain of *Real-Time Online Interactive Applications (ROIA)* including multi-player online games, high-performance simulations, e-learning applications, etc. In particular, we study how to use server resources economically, which is difficult due to continuously changing user numbers.

Cloud Computing with its *Infrastructure-as-a-Service (IaaS)* approach offers new opportunities for ROIA execution and promises a potentially unlimited scalability by distributing application processing on an arbitrary number of resources given suitable adaptation mechanisms. Clouds allow for adding/removing resources on demand. This opens for ROIA an opportunity to serve very high numbers of users and still comply with QoS demands. Despite a variable number of users, Cloud resources can be used efficiently if the application supports adding/removing resources during runtime. Hence, using Cloud Computing for resource provision and the Lifecycle model for implementing adaptable ROIA complement each other.

This paper studies how Cloud Computing and the S-Cube Lifecycle Model can be utilized for ROIA applications. We illustrate how the Lifecycle Model expresses the major design and execution aspects of ROIA on Clouds and present our novel resource management system RTF-RMS that implements concrete mechanisms for ROIA development and adaptation according to the Lifecycle. We report experimental results on the influence of the proposed adaptation mechanisms on the application performance.

The paper is structured as follows. We introduce the S-Cube Lifecycle Model in Section 2, followed by a description of the challenges in ROIA development and execution on Clouds in Section 3. Section 4 illustrates how the Lifecycle Model is applied for ROIA development on Clouds using RTF-RMS. Section 5 reports experimental results on the adaptation of a sample ROIA, and Section 6 concludes the paper.

2 Lifecycle Model

Many of the existing development methodologies for service-based applications (SBA) are based on the results carried out in the fields of classical software and system engineering and do not facilitate SBA adaptation [3,4,5]. Some of the reported SBA development approaches such as SOUP (Service Oriented Unified Process) [6] or the approach by Linner et al [7] do support some level of adaptation, however, they lack sufficient process details. Lane and Richardson [8] carried out a systematic literature review of SBA development approaches, they identified 57 such approaches of which there were only eight that specifically dealt with adaptation. Only four of these eight approaches target the adaptation of SBAs, the others target the adaptation of services.

Each of the four approaches shows interesting features, but even those that enable the definition of various adaptation strategies lack a coherent approach to support designers in this complex task. Moreover, they focus on the implementation process without considering what impact adaptation has on the rest of the development and operational lifecycle [9,10,11]. Finally, they also tend to focus on particular types of adaptation, such as adaptation due to requirement violations [12], or service substitution due to application constraints [13], so it is difficult to elicit generic adaptation mechanisms from them. In summary, each of these approaches focused on the analysis and design processes without consideration for any other development or runtime processes.

The Lifecycle Model proposed in the S-Cube project (see Fig. 1) aims to support the design of adaptable SBAs [14]. It provides a solid reference [15] for practitioners who are planning to develop adaptable SBAs since that it has advantages over similar

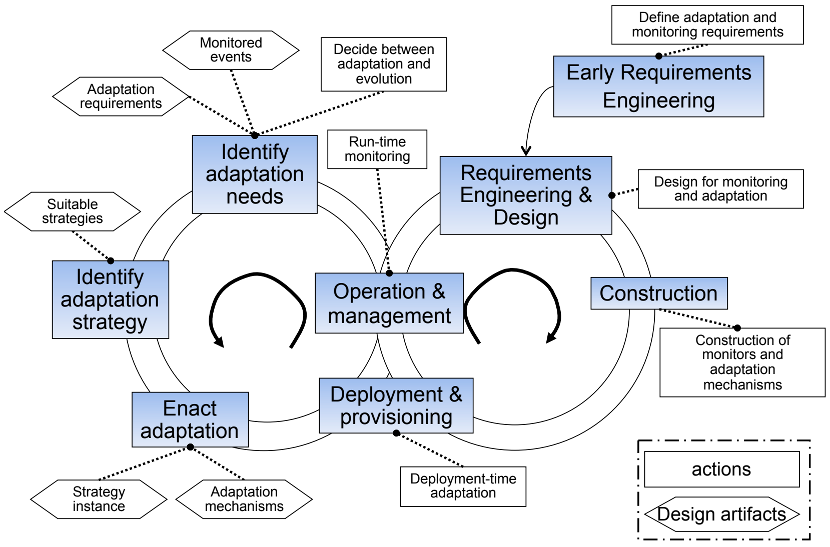
Define adaptation and monitoring requirements
Monitored events
Decide between adaptation and evolution
Adaptation requirements
Early Requirements Engineering
Identify adaptation needs
Run-time monitoring
Requirements Engineering & Design
Design for monitoring and adaptation
Suitable strategies
Identify adaptation strategy
Operation & management
Construction
Construction of monitors and adaptation mechanisms
Enact adaptation
Deployment & provisioning
Deployment-time adaptation
Strategy instance
Adaptation mechanisms
actions
Design artifacts

Fig. 1. Lifecycle for adaptable service-oriented systems

approaches in that it focuses on software process rather than the specific adaptation mechanism implementation techniques. It highlights not only the typical design-time iteration cycle, but it also introduces a new iteration cycle that is performed at runtime when the application needs to be adapted on-the-fly. Both cycles coexist and support each other during the lifetime of the application. In particular, the design time activities allow for *evolution* of the application, i.e., introduction of permanent and, usually, important changes, while the runtime activities allow for temporary *adaptation* of the application to a specific situation. At design time, it is important to analyze functional and non-functional requirements, context and machine characteristics in order to (i) identify the types of changes that trigger self-adaptation, (ii) define mechanisms to monitor the environment and the system behavior, and (iii) develop strategies for self-adaptation.

In the *(Early) Requirement Engineering and Design* phase, the relevant context dimensions and the application and system characteristics are considered in order to elicitate adaptation and monitoring requirements and in particular define the types of changes that trigger self-adaptation. Subsequently, during the *Construction* phase, the corresponding monitoring and adaptation mechanisms are designed, developed and then refined until the *Deployment and Provisioning* phase. At runtime (*Operation and Management* phase), the system is continuously monitored in order to support the detection of the relevant context and system changes. When changes occur, the system might require evolution or adaptation interventions. Evolution is performed if the system needs to be redesigned and thus it requires the reiteration of the described cycle starting from the requirements engineering and design phase.

In the *Identify adaptation need* phase, specific situations that demand for adaptation are identified. Each adaptation need has to be associated with particular *Adaptation Strategies* that are able to satisfy the corresponding adaptation requirements. Based on the current situation, the knowledge obtained from previous executions, and the available adaptation strategies, a reasoner (e.g., a resource management system) selects the most suitable adaptation strategy that will be performed in the *Enact adaptation* phase. Fig. 1 highlights for each phase the various adaptation- and monitoring-specific actions (boxes) carried out throughout the lifetime of an SBA and the main design artifacts that are exploited to perform adaptation (hexagons).

3 ROIA Development and Execution on Clouds

In Real-Time Online Interactive Applications (ROIA), there are typically multiple users who concurrently access a common application state and interact with each other within one virtual environment. The users access the application from different client machines and control their avatars that can influence other users' avatars. Since ROIA have very high performance requirements, the application state processing is performed on multiple servers: the virtual environment is divided into disjoint areas (*zones*) and each server is processing clients inside a particular zone, i.e., the overall workload of the application is distributed on multiple resources. Hence, ROIA are highly distributed applications with challenging QoS demands, such as: short response times (about 0.1-1.5 s), high update rate of the application (up to 50 Hz), large and frequently changing number of users in a single application instance (up to 10^4 simultaneously).

An important challenge is that the number of users participating in a ROIA session and, thus, the workload, is often subject to daytime-dependent changes. As a consequence, expensive up-front investments are made to build a server pool which is able to handle peak user numbers but will be underutilized most of the time when the load is below the peak. Hence, dynamic adaptation of ROIA sessions during runtime is crucial.

We address this challenge by using Cloud Computing for resource provision. Cloud Computing allows to add/remove resources on demand and promises a potentially unlimited scalability by distributing frequent state computations on an arbitrary number of resources given suitable adaptation mechanisms. Despite a variable number of users, Cloud resources can be used efficiently if the application provides suitable adaptation mechanisms. Hence, using Cloud Computing for resource provision and the Lifecycle Model for implementing adaptable ROIA complement each other.

In order to support ROIA development and adaptation on Clouds, we develop the RTF-RMS resource management system [16] on top of the Real-Time Framework (RTF) [17]. RTF-RMS implements the following mechanisms for ROIA development on Clouds:

1. *Monitoring* of application-specific data, e.g., update rate, number of entities, etc.
2. *Distribution handling* for the dynamic adaptation of application sessions by adding/ removing Cloud resources using particular adaptation strategies (described below).
3. *Application profiles* for implementation of application-specific adaptation triggers.
4. High-level *development tools* for communication and application state distribution.

4 Using the Lifecycle Model for ROIA Development on Clouds

This section describes how RTF-RMS supports the developer in designing adaptable ROIA according to the different phases of the Lifecycle Model. In [2], we showed how the Lifecycle described in Section 2 can be applied for ROIA development. In this section, we demonstrate how the Lifecycle Model is used for ROIA development in Cloud environments by exploiting RTF-RMS.

The design of an adaptive application requires the definition of suitable adaptation strategies and adaptation triggers. In the following, we illustrate how RTF-RMS supports the developer in designing adaptable ROIA in Cloud environments according to the different phases of the Lifecycle model.

In the "Requirement Engineering" phase, the application developer must identify suitable adaptation requirements for his application. For ROIA, the mechanisms for monitoring and adaptation should be non-intrusive, i.e., take place in parallel with the application execution, such that users are not aware of changes inside the application.

For the "Construction" phase, RTF-RMS provides the developer with a C++ library of high-level functions for optimized communication handling (client-server and inter-server) and efficient application state distribution in a multi-server environment. By using RTF-RMS for communication and distribution handling, monitoring mechanisms are automatically integrated in the application processing. These monitoring mechanisms are used in the next phase of the Lifecycle to implement adaptation triggers.

In the "Deployment and Provisioning" phase, trigger rules for each adaptation trigger are defined. We describe the implementation of adaptation triggers in Section 4.1.

In the "Operation and Management" phase, the application is running and monitoring data are checked continuously against the trigger rules to detect changes in the context or in the system that could require adaptation.

In the "Identify adaptation need" phase, RTF-RMS detects a violation of trigger rules which indicates the demand for adaptation.

In the "Identify adaptation strategy" phase, RTF-RMS analyzes the number of application servers and their current workload to choose an adaptation strategy. A detailed description of adaptation strategies provided by RTF-RMS is given in Section 4.2.

In the "Enact adaptation" phase, RTF-RMS enacts the chosen adaptation strategy and changes the distribution of the application processing accordingly.

4.1 Adaptation Triggers for ROIA on Clouds

In general, the adaptation is triggered when some changes occurs. Such changes may affect the component services or the context of the considered application. ROIA require an adaptation when one of the following changes occurs:

1. *Change in Quality of Service*: QoS violations may be caused by unreliable Cloud resources. QoS violations for ROIA may be related to changes in update rate, response time, throughput, resource usage, packet loss and service availability.
2. *Change in computational context*: application requirements sometimes change on the basis of variations of the computational context such as variations of CPU and memory load or incoming/outgoing bandwidth.
3. *Change in business context*: it refers to changes in user preferences which were not predicted in advance, e.g., too many concurrent users connected to the application or the increase of the number of requests per application.

RTF-RMS provides a generic mechanism to implement adaptation triggers by specifying an *application profile*. In the application profile, developers can specify significant

Listing 1. Excerpt from an example application profile for a fast-paced action game

```
<appProfile>
  <metric>UpdateRate</metric>
  <addResourceThreshold>25</addResourceThreshold>
  <removeResourceThreshold>100</removeResourceThreshold>
</appProfile>
```

monitoring values for their particular application, e.g., update rate in Hz, see Listing 1 for an example. For each monitoring value in the application profile thresholds have to be defined. If a monitoring value exceeds the `addResourceThreshold`, a new resource is added to the application processing; if monitoring values of any application server fall below the `removeResourceThreshold` dispensable resources are removed. Application developers can find suitable values for these thresholds by considering the runtime behaviour of their application on physical resources and calculating the virtualization overhead of Cloud resources, or by conducting benchmark experiments in a Cloud.

In order to choose between different adaptation strategies, RTF-RMS creates *reports* for each application server. A report describes the load of a particular server, e.g., current update rate in Hz. A *zone report* is created for each zone by collecting reports from all servers involved in the zone processing and calculating their average load.

Fig. 2 illustrates an example of choosing adaptation strategies using zone reports. We assigned Server A to the processing of Zone 1, and Server B and C were assigned to Zone 2. The zone report of Zone 1 identifies an average update rate of 20 Hz. Given an `addResourceThreshold` of 25 Hz, RTF-RMS decides to add a new resource to the processing of Zone 1. The zone report of Zone 2 identifies an average update rate of 60 Hz which is between `addResourceThreshold` and `removeResourceThreshold`, but Server B has an update rate of 20 Hz. Hence, RTF-RMS migrates users from Server B to Server C to distribute the workload equally on both servers.

Another challenge for the implementation of adaptation triggers on Clouds is the compensation of long startup time of Cloud resources which may take up to several minutes. In RTF-RMS, multiple requests for new Cloud resources are sent in parallel to the Cloud API in order to start multiple resources as quickly as possible. Moreover, RTF-RMS introduces a *resource buffer* to which a predefined number of Cloud resources are moved in advance, i.e. before they are demanded by the application. Resources in the resource buffer are immediately available. If any resource from the resource buffer is integrated in the application processing, RTF-RMS checks whether new Cloud resources must be started in order to keep a certain number of resources in the resource buffer. A detailed description of how to choose the resource buffer size in order to minimize the cost-overhead generated by leasing resources in advance is provided in [16].

4.2 Adaptation Strategies for ROIA on Clouds

The adaptation triggers described in the previous section identify the need for adaptation which is implemented by different adaptation strategies. In order to react to changes and avoid inefficiencies, it is necessary to identify the most suitable adaptation strategy that is able to align the application behaviour with the context and system requirements.

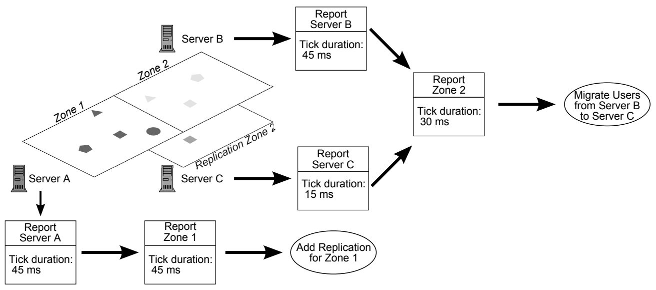

Fig. 2. Finding a suitable adaptation strategy using zone reports

Fig. 3 illustrates how RTF-RMS chooses between four adaptation strategies proposed in our previous work [18]:

1. *User migration*: Users are migrated from an overloaded server to an underutilized server which is replicating the same zone. For this purpose, user connections are switched from one server to another. RTF-RMS distributes users by default equally between the application servers for a particular zone. User migration is restricted to servers that are replicating the same zone because managing users located in different zones on the same server would considerably increase the inter-server communication for processing user interactions which are typically limited to nearby users. User migration is the preferred action if the load of an overloaded server can be compensated by running resources.
2. *Replication enactment*: New application servers are added in order to provide more computation power to the highly frequented zone. This strategy is called replication: each application server keeps a complete copy of the application state, but each server is responsible for computing a disjoint subset of entities. As soon as the new server has been added, RTF-RMS migrates a number of users to the new replica in order to balance the load. Replication implies an additional inter-server communication, so its scalability is limited. Since the replication overhead depends on the inter-server communication in a particular application, the maximum number of replicas per zone can be specified in the application profile. If the number of active replicas for a particular zone is below the maximum number of replicas specified by the application profile, then replication is used; otherwise the resource substitution strategy (described next) is preferred.
3. *Resource substitution*: An existing application server is substituted by a more powerful resource in order to increase the computation power for highly frequented zones. For this purpose, RTF-RMS replicates the targeted zone on the new resource and migrates all clients to the new server. The substituted server is shut down.

   If no better resources are available for substitution, the application reached a critical user density, i.e., large numbers of users in the same geographical area, which cannot be further improved by the generic adaptation strategies offered by RTF-RMS. In this case, the application requires redesign according to the design time activities of the Lifecycle Model.

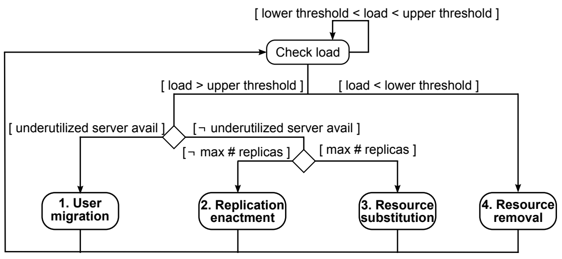
[lower threshold < load < upper threshold]
Check load
[load > upper threshold]
[load < lower threshold]
[underutilized server avail]
[¬ underutilized server avail]
[¬ max # replicas]
[max # replicas]
1. User migration
2. Replication enactment
3. Resource substitution
4. Resource removal

Fig. 3. RTF-RMS chooses between four different adaptation strategies

4. *Resource removal*: If the update rate of an underutilized application server falls below the `removeResourceThreshold`, RTF-RMS checks whether the zone that is managed by this server is replicated by other servers. If not, nothing happens since each zone must be assigned to at least one application server. If other application servers are replicating this zone, users are migrated equally to these servers, after which the underutilized server is shut down.

Another challenge for the cost-efficient adaptation of ROIA on Clouds is the consideration of leasing periods. In commercial Cloud systems, resources are typically leased and paid per hour or some longer leasing period. Since ROIA have dynamically changing user numbers, the time after which Cloud resources become dispensable is very variable. However, resources will not be used cost-efficiently if they are shut down before the end of their leasing period. In RTF-RMS, resources that have become dispensable are removed from the application processing and moved to the resource buffer. Cloud resources in the buffer are shut down at the end of their leasing period or they are integrated in the application processing again if required.

5 Experiments

In the following, we target the "Enact adaptation" phase of the Lifecycle Model and report experimental results of the replication enactment adaptation strategy using an example of a multi-player action game called RTFDemo [17]. In our experiments, we evaluate how RTF-RMS triggers adaptation if QoS changes as described in Section 4.1; from the adaptation triggers described in Section 4.1, we chose the update rate to trigger adaptation. In order to provide a seamless gaming experience, users should not receive less than 25 updates per second over a longer time period. Hence, we defined an adaptation trigger rule with 25 updates per second as the `addResourceThreshold`.

In our experiments, we use a private Cloud with the Eucalyptus framework (version 2.0.2) [19]; servers are Intel Core Duo PCs with 2.66 GHz and 2 GB of RAM.

We start a single RTFDemo server on a Cloud resource and connect 260 clients to it. Fig. 4 shows that the update rate of Server 1 initially drops with the growing number of connected clients. When the update rate of Server 1 falls below the threshold of 25 Hz, RTF-RMS requests a new Cloud resource from the resource buffer (for this experiment, the size of the buffer was configured as 1). Although the requested resource is already

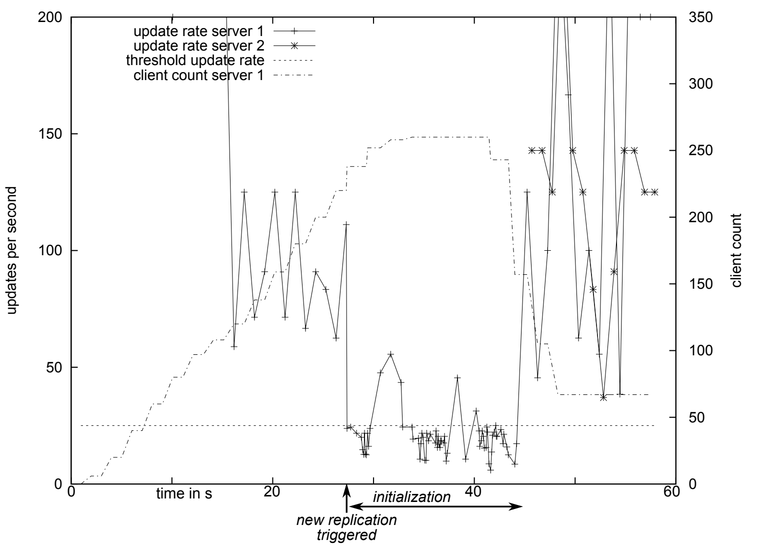

Fig. 4. Load balancing by replication enactment

started up (since it is in the buffer), we observe that a certain time period is still required to add the new server to the application processing and start user migration. This delay of approximately 15 seconds is caused by the initialization and inter-server communication that are required to integrate the new server in the application processing. After the migration is accomplished, the update rate of Server 1 has increased from 20 Hz to about 100 Hz. The update rate of server 1 and server 2 fluctuates between 50 and 200 Hz caused by the continuously changing number of interactions between the 260 clients. Note that if the resource were not in the buffer and would have been started up from scratch, the delay would be much longer, in the order of 130 seconds. Hence, using the resource buffer for starting Cloud resources in advance reduces startup times by a factor of approximately 9 which allows for faster adaptation enactment in contrast to solutions that start resources from scratch, e.g., Amazon Elastic Load Balancing [20].

6 Conclusion

This paper presents how the S-Cube Lifecycle Model can be utilized for developing adaptive ROIA on emerging Cloud systems. We showed how our resource management system RTF-RMS implements concrete mechanisms for ROIA development and execution on Clouds according to the Lifecycle. In extension of our previous work that proved the feasibility of applying the S-Cube Lifecycle on ROIA development on a static set of physical resources [2], this paper targets the specific challenges related to Clouds. In particular, we illustrated how the Cloud influences the definition of adaptation triggers and strategies and how RTF-RMS allows for a cost-effective leasing of Cloud resources on demand by buffering unused resources; thereby the startup times of Cloud resources are reduced. Our adaptation triggers are based on application-specific monitoring values provided by RTF-RMS, and, hence, go beyond the state-of-the-art adaptation mechanisms on common Cloud platforms that are based on generic system information. Our experimental results demonstrate how the replication enactment adaptation strategy implemented in RTF-RMS improves the performance of a multi-player, real-time online game and how the resource buffer decreases startup times of Cloud resources.

Acknowledgments. We are grateful to the anonymous reviewers for their helpful remarks on the preliminary version of the paper. Our research has received funding from the EC's 7th Framework Programme under grant agreement 215483 (S-Cube).

References

1. The S-Cube project (2011), `http://www.s-cube-network.eu`
2. Meiländer, D., Gorlatch, S., Cappiello, C., Mazza, V., Kazhamiakin, R., Bucchiarone, A.: Using a Lifecycle Model for Developing and Executing Adaptable Interactive Distributed Applications. In: Di Nitto, E., Yahyapour, R. (eds.) ServiceWave 2010. LNCS, vol. 6481, pp. 175–186. Springer, Heidelberg (2010)
3. Rational, Rational unified process - best practices for software development teams. Tech. Rep. TP026B (1998)
4. Papazoglou, M.P., van den Heuvel, W.: Service-oriented design and development methodology. Int. J. Web Eng. Technol. 2(4), 412–442 (2006)
5. Arsanjani, A., Ghosh, S., Allam, A., Abdollah, T., Gariapathy, S., Holley, K.: SOMA: a method for developing service-oriented solutions. IBM Syst. J. 47, 377–396 (2008)
6. Mittal, K.: Service oriented unified process (2009), `http://www.kunalmittal.com/html/soup.html`
7. Linner, D., Pfeffer, H., Radusch, I., Steglich, S.: Biology as Inspiration Towards a Novel Service Life-Cycle. In: Xiao, B., Yang, L.T., Ma, J., Muller-Schloer, C., Hua, Y. (eds.) ATC 2007. LNCS, vol. 4610, pp. 94–102. Springer, Heidelberg (2007)
8. Lane, S., Richardson, I.: Process models for service based applications: A systematic literature review. Information and Software Technology (2010)
9. Wautelet, Y., Achbany, Y., Lange, J.-C., Kolp, M.: A Process for Developing Adaptable and Open Service Systems: Application in Supply Chain Management. In: Filipe, J., Cordeiro, J. (eds.) ICEIS 2009. LNBIP, vol. 24, pp. 564–576. Springer, Heidelberg (2009)
10. Vale, S., Hammoudi, S.: Model driven development of context-aware service oriented architecture. In: The 11th IEEE International Conference on Computational Science and Engineering - Workshops, pp. 412–418 (2008)
11. Margaria, T., Steffen, B., Wirsing, M., et al.: SENSORIA Patterns: Augmenting Service Engineering with Formal Analysis, Transformation and Dynamicity. In: Margaria, T., Steffen, B. (eds.) ISoLA 2008. CCIS, vol. 17, pp. 170–190. Springer, Heidelberg (2008)
12. Spanoudakis, G., Zisman, A., Kozlenkov, A.: A service discovery framework for service centric systems. In: 2005 IEEE International Conference on Services Computing, vol. 1, pp. 251–259 (2005)
13. Verma, K., Gomadam, K., Sheth, A.P., Miller, J.A., Wu, Z.: The METEOR-S approach for configuring and executing dynamic web processes. Tech. rep. (2005)
14. Bucchiarone, A., Cappiello, C., Di Nitto, E., Kazhamiakin, R., Mazza, V., Pistore, M.: Design for Adaptation of Service-Based Applications: Main Issues and Requirements. In: Dan, A., Gittler, F., Toumani, F. (eds.) ICSOC/ServiceWave 2009. LNCS, vol. 6275, pp. 467–476. Springer, Heidelberg (2010)

15. Lane, S., Bucchiarone, A., Richardson, I.: SOAdapt: A Process Reference Model for Developing Adaptable Service-Based Applications. Information and Software Technology (2011)
16. Meiländer, D., Ploss, A., Glinka, F., Gorlatch, S.: A Dynamic Resource Management System for Real-Time Online Applications on Clouds. LNCS. Springer (2011) (to appear)
17. The Real-Time-Framework (RTF) (2011), `http://www.real-time-framework.com`
18. Glinka, F., Raed, A., Gorlatch, S., Ploss, A.: A Service-Oriented Interface for Highly Interactive Distributed Applications. In: Lin, H.-X., Alexander, M., Forsell, M., Knüpfer, A., Prodan, R., Sousa, L., Streit, A. (eds.) Euro-Par 2009. LNCS, vol. 6043, pp. 266–277. Springer, Heidelberg (2010)
19. Nurmi, D., Wolski, R., Grzegorczyk, C., et al.: The Eucalyptus Open-Source Cloud-Computing System. In: 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, pp. 124–131. IEEE Computer Society (2009)
20. Amazon Web Services (2011), `http://aws.amazon.com`