Dynamic Modelling of Tactics Impact on the Stability of Self-aware Cloud Architectures

Salama, Maria; Shawish, Ahmed; Bahsoon, Rami

DOI:
10.1109/CLOUD.2016.0126

Dynamic Modelling of Tactics Impact on the Stability of Self-aware Cloud Architectures

Maria Salama
School of Computer Science
University of Birmingham
Birmingham, B15 2TT, UK
Email: m.salama@cs.bham.ac.uk

Ahmed Shawish
Faculty of Informatics & Computer Science
The British University in Egypt, Egypt
& Ain Shams University, Egypt
Email: ahmed.gawish@bue.edu.eg

Rami Bahsoon
School of Computer Science
University of Birmingham
Birmingham, B15 2TT, UK
Email: r.bahsoon@cs.bham.ac.uk

Abstract—**Given the elasticity, on-demand nature, and run-time dynamics of the cloud, a stable self-adaptive architecture should keep the fulfilment of Quality of Service objectives stable, while performing stable adaptations that converge towards these objectives. The dynamic management and selection of architectural tactics, as adaptation mechanisms, shall be in the heart of the adaptation process, as being essential for effective and stable adaptations. This calls for measuring the impact of tactics on the stability of inter-related quality attributes during run-time. In this paper, we introduce a Markovian-based analytical model for dynamically assessing the impact of tactics on the stability behaviour of self-adaptive cloud architectures. The model also employs self-awareness capabilities for better-informing the selection of optimal tactics configurations leading to stability. Experimental evaluations have shown the accuracy and efficiency of the model in measuring and predicting the impact of tactics on stabilising the Quality of Service provision and the adaptation process.**

I. Introduction

Given the elastic, dynamic and on-demand nature of the cloud, a cloud architecture is responsible for coping and responding to run-time workload in an elastic, scalable and dynamic manner. Self-adaptivity in cloud architectures is engineered to achieve that level of dynamicity and scalability, as well as to comply with the changes in components, fluctuations in workloads, and environmental conditions during run-time. Examples of adaptation strategies include architectural tactics, as mechanisms for better tuning, responding and achieving Quality of Service (QoS) attributes; such as response time, throughput, energy efficiency. Architectural tactics are inherently architectural decisions, with measurable response when supporting a quality attribute subject of interest.

With the cloud run-time dynamics, stability would be a desirable property for self-adaptive cloud architectures. In this context, stability could be seen in two-folds: (i) keeping the QoS provision stable, and (ii) performing stable adaptations. The former form of stability corresponds to the quality attributes desired to be fulfiled and how constantly they are fulfiled without SLA violations. The latter form is the degree in which the adaptation process converges towards QoS objectives [1]. Both aspects, mapped to each other, are used to evaluate the quality of adaptation [1], and hence the architecture efficiency in provisioning QoS objectives. An unstable architecture would result in not fulfilling required QoS and consequent SLA violations, as well as inefficient behaviour when performing unnecessary adaptations not converging towards quality objectives. Achieving stability for self-adaptive architectures during run-time calls for dynamic models to measure and predict: (i) the extent to which the architecture meets QoS objectives without violations, and (ii) whether the adaptation converges to the right objectives. Such models can be autonomously used by the self-adaptive architectures to measure and predict the impact of tactics, and hence, effectively perform stable adaptations.

Though architecture-level performance models provide a tool for performance prediction [2], current approaches for modelling the context of self-adaptive software are limited in their coverage for various quality attributes, with performance-being the commonly attribute of interest. They provide limited built-in run-time support for managing QoS, especially when dealing with stability of both QoS provision and adaptation. The Environment Domain Models and Impact Models [3] [4], designed to model the impact of adaptation, are declarative specification languages equipped with formal semantics to capture the run-time behaviour, but with no direct linkage to quality targets and their stability.

As tactics are mechanisms to achieve quality attributes, the dynamic management of these tactics shall be in the heart of the adaptation process, essential for an effective and stable adaptation. The run-time dynamic environment of cloud and the important role of tactics as a quality-response during run-time call for dynamic modelling is essential for measuring and predicting the impact of tactics on the stability of quality attributes, as well as the performance of tactics in converging towards the adaptation goals.

In this paper, we propose a Markovian-based analytical model for dynamically assessing the impact of architectural tactics on the stability of quality attributes during run-time. The model can assist in selecting the “optimal” tactic configuration, which can ensure the stability of QoS objectives and the convergence of adaptations towards these objectives. Given the uncertain fluctuating workload that may trigger adaptations, the Markov-based model is able to dynamically capture the states for processing the workload and the probabilistic transitions between these states. Several quality metrics; such as response time, throughput, processing cost, energy consumption;

can, then, be deduced from the model. The proposed model is assisted by queueing-theoretic concepts that discipline the analysis when handling run-time workload. The analysis aims for assessing and predicting the behaviour of the architecture during run-time.

Though Markov-based and queueing models were studied in the context of cloud performance modelling, the novelty of the proposed model relies in: (i) employing multiple dynamic parallel queues to deal with the heterogeneity of the environment, related to the different computational capacities and configurations of physical and virtual machines in the architecture, (ii) providing fine-grained control for the impact of tactics on the stability of quality attributes, as well as predicting their performance and their impact on inter-related attributes, and (iii) employing self-awareness principles to discipline the way workload is processed, in order to provide better informed adaptations relative to various scenarios. Employing self-awareness would provide more intelligent adaptation, and better trade-off management between different tactics. The benefit of the improved run-time adaptation is observed by achieving the stability of quality attributes subject of interest.

Our model is experimentally evaluated on the *RUBiS* benchmark [5] using the *World Cup 1998* workload trend [6]. The quality-driven self-aware cloud architecture [7] was simulated, extending CloudSim [8]. The comparison between the experimental results and the numerical results obtained by a MATLAB implementation of the analytical model shows the accuracy and correctness of our model. Comparing our self-aware model with non-self-aware, the former outperform in handling QoS and converging the architecture into stability following adaptations.

The rest of this paper is organised as follows. Section II introduces background about quality-driven self-aware architectures. Section III presents the proposed model, and section IV presents our experimental evaluation. Section V discusses related work. The paper is concluded in section VI.

II. Background

Self-awareness and self-expression have recently emerged in the design and operation of complex, heterogeneous and dynamic software [9] [10]. The self-aware architecture style draws on the principles of self-awareness to enrich self-adaptive architectures with self-awareness capabilities, and is defined based on a *self-aware node unit* [9].

Different levels of self-awareness, called capabilities, have been identified to better assist the self-adaptive process; such as stimulus-, goal-, interaction-, time-, and meta-self-awareness [9] [10] . For more details, the reader is referred to the latter references. In this work, we employ the goal-awareness capability, that is having knowledge of current goals, objectives, preferences and constraints, in such a way that it can reason about it.

Various self-aware architecture patterns have been introduced, where each contains different self-awareness capabilities [9] [10]. In our recent work [7], we have extended the emerging class of self-aware architectures with QoS self-management components and a catalogue of architectural tactics designated to fulfil different quality attributes. The architecture features a catalogue of architectural tactics; such as vertical scaling, horizontal scaling, virtual machines consolidation [7]. The purpose of adaptation is to satisfy the QoS requirements of multi-tenant users, by changing configuration and choosing optimal tactics.

Incorporating tactics, as adaptation mechanisms to meet the QoS objectives, aimed to improve and enrich the quality of self-adaptation. Meanwhile, achieving a stable QoS provision during run-time through stable adaptations requires continuous fine-grained evaluation of the tactics performance and their impact on the stability of inter-related quality attributes.

III. Dynamic Self-Aware Impact Model

Cloud architectures exhibit probabilistic behaviours during run-time, due to the uncertain fluctuation of workload at run-time, the constraints on available resources and changes in the environment. A self-adaptive architecture would dynamically respond to the run-time dynamics using adaptation mechanisms; such as architectural tactics; for better tuning, responding and achieving QoS. Architecture's behaviour can also be affected by prior adaptation actions.

Given the run-time cloud dynamics and the probabilistic behaviours of the self-adaptive architecture, a Markovian analytical modelling can provide a generic and scalable model for these probabilistic behaviours. Based on multiple parallel dynamic queues, the model can capture instance-related information at finer-grained level of tactics' configurations, given the environment heterogeneity. The model can, then, measure and predict quality attributes for a scenario of interest. Such measurements and predictions can assist in choosing the optimal tactics and their configurations, for stable adaptations achieving the stability of QoS provision.

We also employ self-awareness capabilities to provide a more intelligent way in handling workload and related adaptations. More specifically, we employ the goal-awareness capability in the model, in order to reach a stable adaptation converging towards the quality objectives. This allows taking better informed adaptation actions.

Nevertheless, the modelling is generic and can be extended, following the same fundamental principle of the modelling, to capture other adaptation properties similar to stability. Examples include short settling time - the time required for the adaptive architecture to reach the desired state - and small overshoot - the utilisation of computational resources during the adaptation process to achieve the adaptation goal [1]. In more details, the model can be further refined to capture more states and transitions between these states, where these refinements can enable finer and explicit analysis for other behavioural properties, which can influence adaptation properties.

In the below sub-sections, we present our assumptions, the dynamic queueing model, the quality model, and the tactics' impact modelling.

A. System Model

A self-adaptive cloud node is running m Physical Machines (PMs). A PM$_i$, where $i = \{1, ..., m\}$ runs n_i Virtual Machihes (VMs) sharing computational resources. The number of running VMs varies from one PM to another according to its computational capacity. Service requests (e.g., web service, SaaS request, computational request) are received and processed on the cloud infrastructure, where the workload tends to vary in number of incoming requests, length of each request, and quality requirements according to the client SLA.

We assume the total incoming workload λ will be divided among the m PMs resulting $\{\lambda_1, \lambda_2, \lambda_i, ..., \lambda_m\}$. Several algorithms have been proposed to manage the jobs placement in PMs and VMs. Though we follow a simple approach for requests placement, the same principle can apply to other placement mechanisms. The distribution of workload, in our case, is based on either the PM computational capacity in case PMs computational capacity are different, or equally on all PMs based on their availability.

Each PM, by its turn, will distribute its assigned workload on its n running VMs. Workload is distributed on VMs level either based on VM computational capacity in case the incoming request is constrained by certain computational requirements, or equally in case of no constraints. The workload is denoted by λ_{ij}, where i indicates the PM, j indicates the VM, and $j = \{1, ..., n_i\}$. For a VM$_{ij}$, an $m/m/1$ queue will be formed for the incoming requests to be processed, where the incoming rate of requests constitutes a poisson process of rate $\lambda_{i/n}$ (assuming equal workload distributed on all VMs), and the service process is markovian exponentially distributed, with parameter μ_{ij} and mean $1/\mu_{ij}$ that is handled by that VM. Thus, the total service handled by the self-aware node is $\sum_{i=1}^{m} \sum_{j=1}^{n} \mu_{ij}$. The handling of workload in a self-aware cloud node is illustrated in Figure 1.

Unlike most of the prior models that have employed only single queues, we employ multiple parallel dynamic queues, where the queuing can discipline the way we analyse the workload in relation to heterogeneous environments with varying PMs, VMs, and their computational capacities. The model also features scalability into the analysis, as well as helps in tracking and predicting the behaviour at a given time instance.

B. Quality Model

Considering the formed queue of incoming requests for VM$_{ij}$ as a continuous time Markov chain, the markovian analytical model allows to estimate the quality of service. Given the expected workload λ, the number of PMs m, VMs, and the capacity of both of them, the model can approximate different quality attributes; such as response time (R), mean queue (W), throughput (T), utilisation (ρ), by applying $m/m/1$ queue and Markov process properties, as well as Little's law.

Having performance metrics of each VM independently, all performance metrics for a given PM could be deduced, as well as for the self-adaptive cloud node. The mean response time for PM$_i$ is the mean response time for the n_i VMs running on

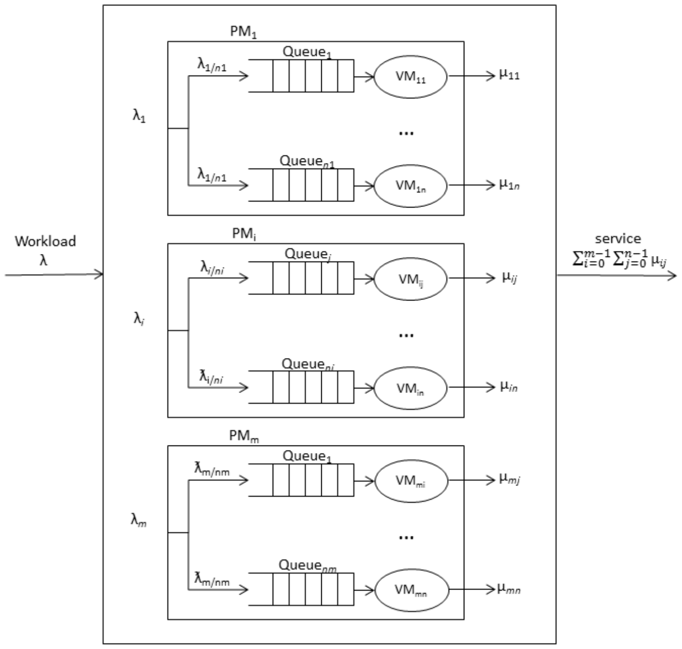

Fig. 1. Dynamic self-aware workload handling

that PM. Also, the mean utilisation and the throughput can be calculated as the sum of the related measures of the n_i VMs.

On the node level, same metrics could also be calculated as the sum of related metrics for the m PMs operating on the node. Operational cost could also be calculated among the node, that is the cost of processing the incoming workload: $C = \sum_{i=1}^{m} \sum_{j=1}^{n} Cost(CPU)_{ij} + Cost(memory)_{ij}$.

And, the total power consumption of all running PMs, given the varying number of VMs and their allocated CPU threads, would be: $C = \sum_{i=1}^{m} E_i$

C. Modelling the impact of adaptation tactics

As an architectural tactic represents codified knowledge about the relationship between architectural decisions and quality attributes, we describe hereunder how our analytical model can accommodate the impact of a diverse range of tactics on the stability of these quality attributes.

First, tactics related to PMs; such as horizontal scaling and consolidation; are reflected on our model by varying the value of m PMs. That is, scaling with a certain number of PMs will be reflected in our model when dividing the incoming workload λ on more PMs; i.e. $m + 1$. This would influence the stability of performance (response time) and greenability (energy consumption).

Second, tactics related to VMs; such as vertical scaling and VMs consolidation; are reflected in our model by increasing or decreasing the total value of n VMs. This influences the average latency of processing the incoming requests.

Third, tactics related to computational capacity; i.e., CPU threads of a specific VM$_{ij}$; are reflected in the increase or decrease of the corresponding service rate μ_{ij}, and hence influence the throughput. Also, the utilisation of VMs, determined by our model, allows consolidating the less utilised VMs (e.g. x VMs are less than 10% utilised) and re-checking the performance metrics given the new number of VMs ($n-x$).

Aiming to stabilise a specific quality attribute, the impact of related tactic could be predicted under different configurations of the tactic, in order to select the optimal configuration. Unlike prior related work, which considered a case of homogeneity, we consider the heterogeneity of environment in PMs, VMs, and their computational capacity. The proposed model is capable to model the sensitivity of quality parameters behaviour with different scenarios varying number of PMs, computational capacities of PMs, number of VMs, allocated CPU threads and requests constraints. Besides, our model allows measuring the cost and energy consumption of the self-adaptive cloud node under these different scenarios.

Given QoS objectives, the model can deduce the impact of adaptations on keeping the provision of these objectives stable. The model can also deduce to what extent an adaptation action could converge towards the stability of adaptation goals. The goal-awareness capability, when employed to discipline the way workload is processed dynamically, has better informed the adaptation process to select the tactic configuration that converges towards the adaptation goal.

IV. Experimental Evaluation

The main objective of our experiments is to evaluate the correctness and accuracy of our analytical model. We also examine the efficiency of our self-aware model compared with non-self-aware.

To conduct the experimental evaluation, the cloud self-aware architecture [7] was simulated, extending CloudSim [8]. The simulation was built using Java JDK 1.7, and was run on an Intel Core i5 3.40 GHz, 8 GB RAM computer. Our experiments used the *RUBiS* benchmark [5], an online auction application defining different services, categorised in two workload patterns: the browsing pattern assuming read-only services, and the bidding pattern simulating both read and write intensive services. Instead of using random workload trend to simulate the number of requests, the number of requests was varied proportionally according to the *World Cup 1998* trend [6]. We compressed the trend in the way that the fluctuation of one day in the trend corresponds to 200 seconds in our experiments. We simulated the cloud dynamics based on the benchmark workload patterns and run the entire World Cup 1998 trend separately for each workload pattern.

To verify the correctness and accuracy of our analytical model, we conducted experiments on the cloud architecture and implemented the equations derived from the analytical model using MATLAB. The model is applied at sampling interval of 200 seconds with the total of 87 intervals corresponding to the number of days in the World Cup 1998 trend. The predictions obtained from the analytical model are, then, compared with the results measured from the experiments, as shown in Figure 2. The comparative results show that both the analytical model and experiments are in agreement, with insignificant discrepancy, throughout the different time intervals of the varying workload trend.

To examine the efficiency of our model, we compared our model in cases of self-aware architecture and conventional self-adaptive architecture lacking self-awareness capabilities. More specifically, we run the entire workload for both service patterns under heterogeneous computational capacities and measured the quality attributes after performing necessary adaptations in both architectures. We report, here, on the average response time in the time intervals where the workload tends to highly fluctuate. The results are depicted in Figure 3. As shown in the figure, the self-aware queueing was able to result in better response time compared with the non-aware queueing. It is also worth to note that the model assisted with self-awareness showed better performance in cases of peak workload for the intensive workload pattern. This reflects the higher quality of adaptations and tactics selection.

To conclude, experimental evaluations have shown the correctness and accuracy of our analytical model, as well as the performance achieved when assisted with self-awareness capabilities. This reflects the model effectiveness in capturing fine-grained information to determine the impact of tactics on the stability of various quality attributes. Yet, it is worth to note some threats to validity: (i) the experiments were conducted in a controlled environment assuming poisson arrivals and servicing, which could be relaxed in real world dynamics, and (ii) experiments have not considered the real life scenario of switching between the browsing and bidding workload patterns. Though, we can consider that our experiments have given good enough indication and approximation of likely scenarios in a practical setting, given the use of the RUBis benchmark and the real world workload trend.

V. Related Work

In the context of modelling the impact of adaptations, Impact Models for architecture-based self-adaptation were proposed [4] to describe the impact of adaptations. Environment Domain Models were also used by adaptive systems to describe their run-time behaviour [3] [4]. The core contribution of both was declarative specification language equipped with formal semantics, but with no direct linkage to quality targets and their stability.

On the other hand, architecture-level performance models have been proposed in the performance engineering community. The works of [2] [11] developed run-time architecture-level performance models that explicitly consider the dynamic aspects of service-oriented software systems as part of their architectural model. Though these models provide performance prediction, they tend to be limited in covering various quality attributes, with performance being the commonly attribute of interest.

VI. Conclusion

In this paper, we proposed a dynamic analytical model for assessing the impact of architectural tactics on the stability of QoS provision and adaptations, given the run-time fluctuating

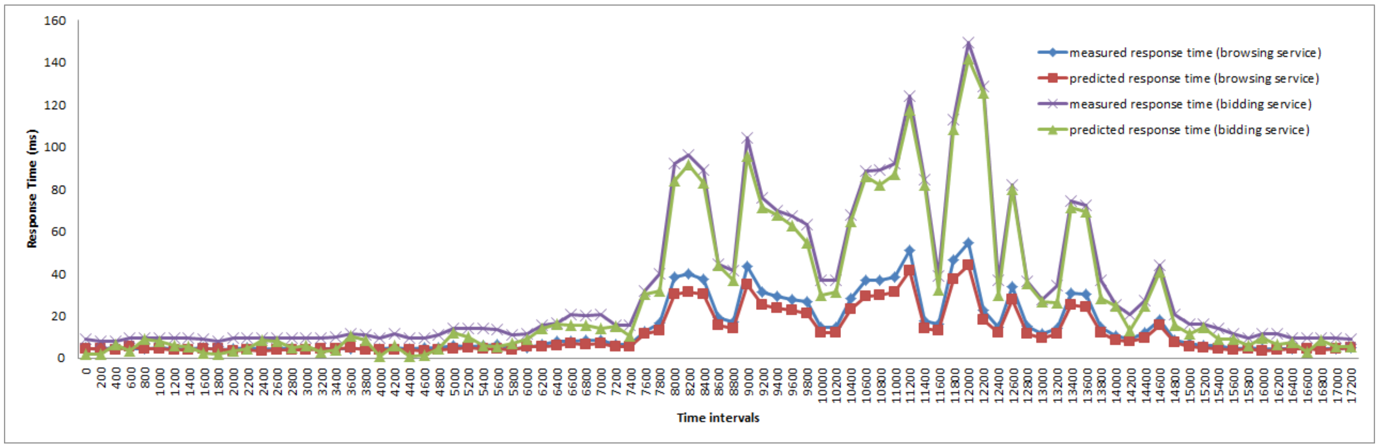

Fig. 2. Actual and predicted response time

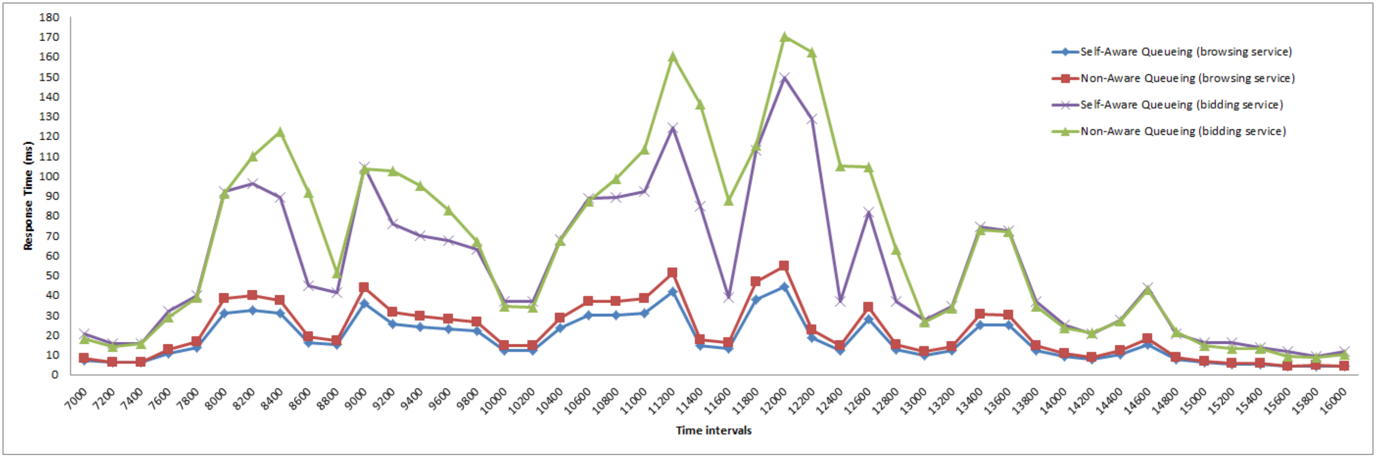

Fig. 3. Response time in cases of self-aware and non-aware queueing

workload. The model features a fine-grained control of the tactics' impact on quality attributes in self-adaptive cloud architectures. This allows dynamically determining the optimal tactic configuration, that will lead to a stable adaptation achieving the adaptation goal. Employing self-awareness capabilities have better informed the adaptation process, compared with non-self-aware. While we have looked at finer analysis of the queuing model to achieve the stability as an adaptation property, the model is scalable to capture other adaptation properties of interest following the same fundamental principle of the modelling.

Our future work will focus on employing historical information from the time-awareness level to devise learning about the performance of tactics under different workloads and environmental conditions.

References

[1] N. Villegas, H. Muller, G. Tamura, L. Duchien, and R. Casallas, "A framework for evaluating quality-driven self-adaptive software systems," in *Proc. 6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, 2011, pp. 80–89.

[2] F. Brosig, N. Huber, and S. Kounev, "Architecture-level software performance abstractions for online performance prediction," *Science of Computer Programming*, vol. 90, Part B, no. 0, pp. 71–92, 2014.

[3] R. Calinescu, C. Ghezzi, M. Kwiatkowska, and R. Mirandola, "Self-adaptive software needs quantitative verification at runtime," *Communications of the ACM*, vol. 55, no. 9, pp. 69–77, 2012.

[4] J. Camara, A. Lopes, D. Garlan, and B. Schmerl, "Impact models for architecture-based self-adaptive systems," in *Proc. 11th International Symposium on Formal Aspects of Component Software (FACS)*, 2014.

[5] "Rice university bidding system." [Online]. Available: www.rubis.ow2.org

[6] M. Arlitt and T. Jin, "A workload characterization study of the 1998 world cup web site," *IEEE Network*, vol. 14, no. 3, pp. 30–37, 2000.

[7] M. Salama and R. Bahsoon, "Quality-driven architectural patterns for self-aware cloud-based software," in *Proc. IEEE 8th International Conference on Cloud Computing (IEEE CLOUD)*, 2015, pp. 844–851.

[8] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. De Rose, and R. Buyya, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience*, vol. 41, no. 1, pp. 23–50, 2011.

[9] F. Faniyi, P. Lewis, R. Bahsoon, and Y. Xin, "Architecting self-aware software systems," in *Proc. 11th Working IEEE/IFIP Conference on Software Architecture (WICSA)*, 2014, pp. 91–94.

[10] T. Chen, F. Faniyi, R. Bahsoon, P. Lewis, X. Yao, L. Minku, and L. Esterle, "The handbook of engineering self-aware and self-expressive systems," University of Birmingham, Report, 2014.

[11] F. Brosig, N. Huber, and S. Kounev, "Automated extraction of architecture-level performance models of distributed component-based systems," in *26th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2011, pp. 183–192.