Self-Aware Software and Systems Engineering: A Vision and Research Roadmap

Samuel Kounev
Institute for Program Structures and Data Organization (IPD)
Karlsruhe Institute of Technology (KIT)
76131 Karlsruhe, Germany
kounev@kit.edu

September 8, 2011

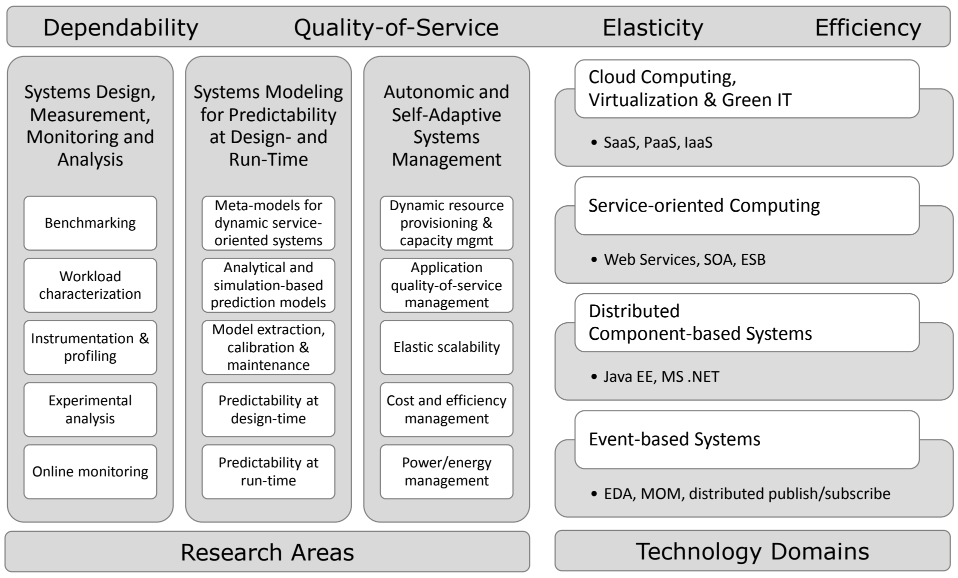
Abstract

With the increasing adoption of virtualization and the transition towards cloud computing platforms, modern IT systems and services are becoming increasingly complex and dynamic. The lack of direct control over the underlying physical hardware and the complex interactions between the applications sharing the physical infrastructure pose some major challenges in providing Quality-of-Service (QoS) guarantees. In this paper, we present a research roadmap and a long-term vision aiming to address these challenges. The presented research agenda is pursued by the Descartes Research Group at KIT. Our long-term goal is to develop a novel methodology for engineering of next generation *self-aware* IT systems and services. The latter will have built-in service architecture models enhanced to capture dynamic aspects of the system environment and maintained automatically during operation. The models will be exploited at run-time to adapt the system to changes in the environment ensuring that resources are utilized efficiently and that QoS requirements are continuously satisfied.

1 Introduction

Modern IT systems based on the Service-Oriented Architecture (SOA) paradigm have highly distributed and dynamic architectures composed of loosely-coupled services that operate and evolve independently. Managing system resources in such environments to ensure acceptable end-to-end application Quality-of-Service (QoS, e.g., availability, performance and reliability) while at the same time optimizing resource utilization and energy efficiency is a challenge. The adoption of virtualization and cloud computing technologies, such as Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS) and Infrastructure-as-a-Service (IaaS), comes at the cost of increased system complexity and dynamicity. The increased complexity is caused by the introduction of virtual resources and the resulting gap between logical and physical resource allocations. The increased dynamicity is caused by the complex interactions between the applications and workloads sharing the physical infrastructure. The inability to predict such interactions and adapt the system accordingly makes it hard to provide QoS guarantees in terms of availability and responsiveness, as well as resilience to attacks and operational failures [11]. Moreover, the consolidation of workloads translates into higher utilization of physical resources which makes the system much more vulnerable to threats resulting from unforeseen load fluctuations, hardware failures and network attacks.

Service providers are often faced with questions such as: What QoS would a new service deployed on the virtualized infrastructure exhibit and how much resources should be allocated to it? What would be the effect of migrating a service from one virtual machine (VM) to another? How should the system configuration be adapted to avoid QoS issues or inefficient resource usage arising from changing customer workloads? Answering such questions requires the ability to predict at *run-time* how the QoS of running services would be affected if the system configuration or the workload changes. We refer to this as *online QoS prediction*. Due to the inability to automatically keep track of dynamic changes in the system environment and predict their effect, modern service-oriented applications often exhibit poor QoS and resource efficiency, and have high operating costs. To accommodate load fluctuations, services are typically hosted on dedicated servers with over-provisioned capacity. Servers in data centers nowadays typically run at around 20% utilization [16] which corresponds to their lowest energy-efficiency region [1]. The growing number of under-utilized servers, often referred to as server sprawl, translates into increasing data center operating costs including power consumption costs, cooling infrastructure costs and system management costs. To counter this development, novel methods for online QoS prediction and autonomic resource management are needed.



Dependability
Quality-of-Service
Elasticity
Efficiency
Systems Design, Measurement, Monitoring and Analysis
Benchmarking
Workload characterization
Instrumentation & profiling
Experimental analysis
Online monitoring
Systems Modeling for Predictability at Design- and Run-Time
Meta-models for dynamic service-oriented systems
Analytical and simulation-based prediction models
Model extraction, calibration & maintenance
Predictability at design-time
Predictability at run-time
Autonomic and Self-Adaptive Systems Management
Dynamic resource provisioning & capacity mgmt
Application quality-of-service management
Elastic scalability
Cost and efficiency management
Power/energy management
Cloud Computing, Virtualization & Green IT
• SaaS, PaaS, IaaS
Service-oriented Computing
• Web Services, SOA, ESB
Distributed Component-based Systems
• Java EE, MS .NET
Event-based Systems
• EDA, MOM, distributed publish/subscribe
Research Areas
Technology Domains

Figure 1: Research Areas and Technology Domains

2 Research Agenda and Vision

The Descartes Research Group[1] at KIT, named after the French philosopher René Descartes, is working on novel approaches to software and systems engineering focusing on the research areas and technology domains depicted in Figure 1.

2.1 Vision

A major part of our research is focused on the development of novel methods, techniques and tools for the engineering of so-called *self-aware* IT systems and services [10]. The latter are designed with built-in online QoS prediction and self-adaptation capabilities used to enforce QoS requirements in a cost- and energy-efficient manner. The current focus is on performance, availability and efficiency aspects, however, long-term we are planning to consider further QoS properties such as reliability and fault-tolerance. Self-awareness, in this context, is defined by the combination of three properties that IT systems and services should possess:

1. *Self-reflective*: aware of their software architecture, execution environment and hardware infrastructure on which they are running, as well as of their operational goals (e.g., QoS requirements, cost- and energy-efficiency targets),
2. *Self-predictive*: able to predict the effect of dynamic changes (e.g., changing service workloads) as well as predict the effect of possible adaptation actions (e.g., changing service deployment and/or resource allocations),
3. *Self-adaptive*: proactively adapting as the environment evolves in order to ensure that their operational goals are continuously met.

[1]http://www.descartes-research.net

2.2 Approach

Our approach to the realization of the above vision is based on the use of *online* QoS models integrated into the system components and capturing all service aspects relevant to managing application QoS and resource efficiency during operation [6, 15, 10]. In contrast to black-box models, the modeling techniques we are working on are designed to explicitly capture all relevant aspects (both static and dynamic) of the underlying software architecture, execution environment, hardware infrastructure, and service usage profiles. In parallel to this, we are working on *self-aware* service platforms designed to automatically maintain models during operation to reflect the evolving system environment. The online models are meant to serve as a "mind" to the running services controlling their behavior, i.e., deployment configurations, resource allocations and scheduling decisions. To facilitate the initial model construction and continuous maintenance during operation, we are working on techniques for automatic model extraction based on monitoring data collected at run-time [4, 8, 7].

Current architecture-level[2] performance models for component-based architectures, surveyed in [12], (e.g., PCM [2], CBML [17], CB-SPE [3]) are used as a basis. The latter are being extended to capture the performance influences of the platforms used at each layer of the system execution environment focusing on the virtualization and middleware layers. Resource allo-

[2]We distinguish between *descriptive* architecture-level QoS models and *predictive* QoS models. The former describe QoS-relevant aspects of software architectures and execution environments (e.g., UML models augmented with QoS-related annotations). The latter capture the temporal system behavior and can be used for QoS prediction by means of analytical or simulation techniques (e.g., Markov chains, layered queueing networks or stochastic Petri nets).

cations at the various layers are modeled explicitly and benchmark results are exploited to quantify the relative performance of different execution platforms. This makes it possible to predict how the QoS of a running service will be affected if resource allocations are modified or if the service is migrated from one VM to another possibly running on a different platform.

Unlike conventional architecture-level QoS models, the developed *online* QoS models are *dynamic* in the sense that they are maintained automatically at runtime to reflect the evolving system environment. To realize this, execution platforms are enhanced to automatically extract and maintain models during operation. Depending on the type of system considered and the availability of monitoring and instrumentation frameworks, the degree of automation of the initial model extraction will vary. For example, for a newly developed system, the extraction could potentially be completely automated, whereas for legacy systems some manual steps might be necessary.

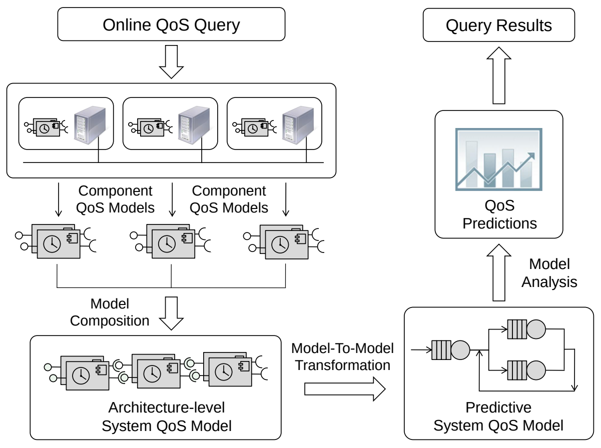


Figure 2: Online QoS Prediction Process

The main purpose of the developed online QoS models is to make it possible to answer QoS-related queries during operation such as: What would be the effect on the QoS of running applications and on the resource consumption of the infrastructure if a new service is deployed in the virtualized environment or an existing service is migrated from one server to another? How much resources need to be allocated to a newly deployed application to ensure that Service Level Agreements (SLAs) are satisfied while maximizing energy efficiency? What QoS would the system exhibit after a period of time if the workload continues to develop according to the current trends? How should the system configuration be adapted to avoid QoS problems or inefficient resource usage arising from changing customer workloads? What operating costs does a service hosted on the infrastructure incur and how does the service workload and usage profile impact the costs? We refer to such queries as *online QoS queries*.

Each time an online QoS query is executed, it is processed by means of the online QoS models which are composed dynamically after determining which specific parts of the system are relevant to answering the query. Figure 2 illustrates the process that is followed in order to provide an answer to a query. First, the online QoS models of all involved system components are retrieved and combined by means of model composition techniques into a tailored architecture-level system QoS model encapsulating all information relevant to answering the QoS query. Given the wide range of possible contexts in which the online QoS models can be used, automatic model-to-model transformation techniques are used to generate tailored predictive models on-the-fly depending on the required accuracy and the time available for the analysis. Existing model-to-model transformations for static architecture-level performance models are used as a basis, e.g., [14, 2, 13]. The target predictive model type and level of abstraction as well as the solution technique are determined on-the-fly based on the required accuracy and the time available for the analysis. Multiple predictive model types (e.g., queueing networks, stochastic Petri nets, stochastic process algebras and general-purpose simulation models) and model solution techniques (e.g., exact analytical techniques, numerical approximation techniques, simulation and bounding techniques) are employed here in order to provide flexibility in trading-off between prediction accuracy and analysis overhead.

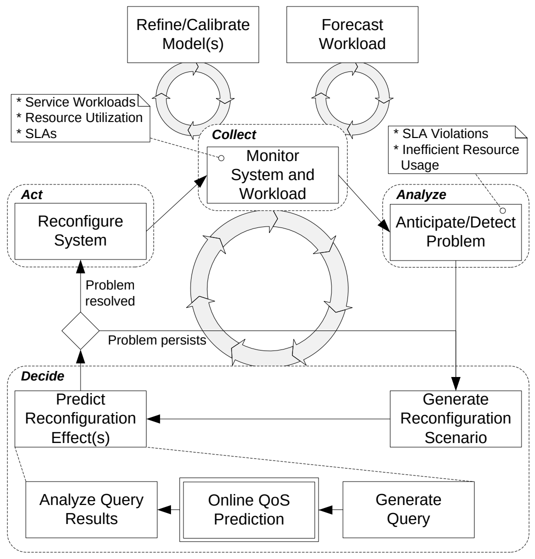


Figure 3: Online Reconfiguration Process

The ability to answer online QoS queries during operation provides the basis for implementing novel techniques for self-aware QoS management [10, 6, 15]. Such techniques are triggered automatically during operation in response to observed or forecast changes in the environment (e.g., varying service workloads). The goal is to *proactively* adapt the system to such changes in order to avoid anticipated QoS problems, inefficient resource usage and/or high system operating costs. The adaptation is performed in an autonomic fashion by considering a set of possible system reconfiguration scenarios (e.g, changing VM placement and/or resource allocations) and exploiting the online QoS query mechanism to predict the effect of such reconfigurations before making a decision [6]. Figure 3 depicts the online reconfiguration process and the self-adaptation control loop which is based on the generic model of a control loop from [5] extended to integrate the use of the online QoS query mechanism. In addition to the main control loop, two additional loops are running in the background, one for continuously refining and calibrating online models and one for forecasting the workload evolution.

2.3 Preliminary Results

As an initial step towards the described vision, we conducted several preliminary case studies. In [4], we studied a complex Java EE application showing how detailed architecture-level performance models can be extracted and maintained automatically at run-time based on online monitoring data. The extracted performance models provided performance predictions with less than 20% deviation from measurements on the real system. In [8], we studied an enterprise data fabric and developed a simulation-based tool for automated performance prediction and capacity planning. The tool, called Jewel, automates resource demand estimation, performance model generation, performance model analysis, and results processing. A detailed experimental evaluation of the tool demonstrated its effectiveness and practical applicability. In [15], we studied a SOA application running in a service-oriented Grid computing environment and showed how online performance models can be used at run-time for autonomic QoS-aware resource allocation. Finally, in our most recent case study [6], we showed how online *architecture-level* performance models can be exploited for self-adaptive resource allocation in virtualized environments. We explored the use of such models to predict the effects of changes in user workloads, as well as to predict the effects of respective reconfiguration actions, undertaken during operation to avoid SLA violations or inefficient resource usage. The case study demonstrated the feasibility of using architecture-level performance models at run-time and the benefits they provide in terms of cost savings.

3 Concluding Remarks

We presented a research roadmap and a long-term vision aiming to develop a novel methodology for the engineering of next generation *self-aware* IT systems and services. Such systems and services are characterized by the combination of three properties: self-reflective, self-predictive and self-adaptive. The described approach raises several big challenges that represent emerging hot topics in the software and systems engineering community and will be subject of long-term fundamental research in the years to come. *Self-Aware Software and Systems Engineering* [9] is a newly emerging research area at the intersection of several computer science disciplines including Software and Systems Engineering, Computer Systems Modeling, Autonomic Computing, Distributed Systems, Cluster and Grid Computing, and more recently, Cloud Computing and Green IT (see Figure 4). The realization of the described vision calls for an interdisciplinary approach considering not only technical but also business and economical challenges. The resolution of these challenges promises to reduce the costs of ICT and their environmental footprint while keeping the high growth rate of IT services.

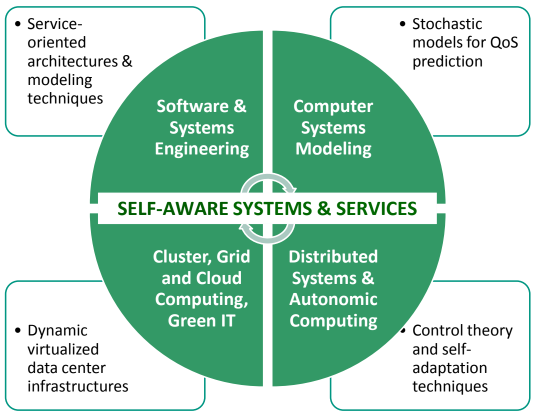

• Service-oriented architectures & modeling techniques
• Stochastic models for QoS prediction
Software & Systems Engineering
Computer Systems Modeling
SELF-AWARE SYSTEMS & SERVICES
Cluster, Grid and Cloud Computing, Green IT
Distributed Systems & Autonomic Computing
• Dynamic virtualized data center infrastructures
• Control theory and self-adaptation techniques

Figure 4: Self-Aware Software & Systems Engineering

References

[1] L. A. Barroso and U. Hölzle. The Case for Energy-Proportional Computing. *IEEE Computer*, 40(12):33–37, 2007.

[2] S. Becker, H. Koziolek, and R. Reussner. The Palladio component model for model-driven performance prediction. *Elsevier Journal of Systems and Software*, 82:3–22, 2009.

[3] A. Bertolino and R. Mirandola. Software Performance Engineering of Component-based Systems. In *Proc. of the 3rd International Workshop on Software and Performance (WOSP 2004)*. ACM Press, 2004.

[4] 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), November 6-11, Oread, Lawrence, Kansas*, 2011.

[5] B. H. C. Cheng, R. de Lemos, H. Giese, P. Inverardi, and J. Magee. Software Engineering for Self-Adaptive Systems: A Research Roadmap. In *Software Engineering for Self-Adaptive Systems, Springer LNCS 5525*, pages 1–26, 2009.

[6] N. Huber, F. Brosig, and S. Kounev. Model-based Self-Adaptive Resource Allocation in Virtualized Environments. In *SEAMS'11: 6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, May 23-24, Waikiki, Honolulu, Hawaii, USA*. ACM Press, 2011.

[7] N. Huber, M. von Quast, M. Hauck, and S. Kounev. Evaluating and Modeling Virtualization Performance Overhead for Cloud Environments. In *1st International Conference on Cloud Computing and Services Science (CLOSER 2011), May 7-9, Noordwijkerhout, The Netherlands*, 2011.

[8] S. Kounev, K. Bender, F. Brosig, N. Huber, and R. Okamoto. Automated Simulation-Based Capacity Planning for Enterprise Data Fabrics. In *4th International ICST Conference on Simulation Tools and Techniques (SIMUTools 2011), March 21-25, Barcelona, Spain*, 2011.

[9] S. Kounev, F. Brosig, and N. Huber. Self-Aware QoS Management in Virtualized Infrastructures (Poster Paper). In *8th International Conference on Autonomic Computing (ICAC 2011), June 14-18, Karlsruhe, Germany*, 2011.

[10] S. Kounev, F. Brosig, N. Huber, and R. Reussner. Towards self-aware performance and resource management in modern service-oriented systems. In *Proceedings of the 7th IEEE International Conference on Services Computing (SCC 2010), July 5-10, Miami, Florida, USA*. IEEE Computer Society, 2010.

[11] S. Kounev, P. Reinecke, K. Joshi, J. Bradley, F. Brosig, V. Babka, S. Gilmore, and A. Stefanek. Providing Dependability and Resilience in the Cloud: Challenges and Opportunities. In A. Avritzer, A. van Moorsel, K. Wolter, and M. Vieira, editors, *Resilience Assessment and Evaluation*, Dagstuhl Seminar 10292. Springer Verlag, 2011.

[12] H. Koziolek. Performance evaluation of component-based software systems: A survey. *Performance Evaluation*, 67(8):634–658, 2010.

[13] H. Koziolek and R. Reussner. A Model Transformation from the Palladio Component Model to Layered Queueing Networks. In *Proc. of the SPEC International Performance Evaluation Workshop (SIPEW 2008)*. Springer LNCS 5119, 2008.

[14] P. Meier, S. Kounev, and H. Koziolek. Automated Transformation of Palladio Component Models to Queueing Petri Nets. In *19th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS 2011), July 25-27, Singapore*, 2011.

[15] R. Nou, S. Kounev, F. Julia, and J. Torres. Autonomic QoS control in enterprise Grid environments using online simulation. *Journal of Systems and Software*, 82(3):486–502, Mar. 2009.

[16] K. Parent. Server Consolidation Improves IT's Capacity Utilization. Vol. 2006: Court Square Data Group, 2005.

[17] X. Wu and M. Woodside. Performance Modeling from Software Components. In *Proc. of the 3rd International Workshop on Software and Performance (WOSP 2004)*, pages 290–301. ACM Press, January 2004.