A Methodology and a Tool for QoS-Oriented Design of Multi-Cloud Applications

Giovanni Paolo Gibilisco

Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB)

Politecnico di Milano

Supervisor: Prof. Danilo Ardagna

Co-supervisor: PhD. Michele Ciavotta

Tutor: Prof. Carlo Ghezzi

A thesis submitted for the degree of

Doctor of Philosophy

05 Feb. 2015

Abstract

This work focuses on the support of the development of multi-cloud enabled applications with Quality of Service (QoS) guarantees. It embraces the model driven engineering principles and aims at providing development teams with methodologies and tools to assess the expected QoS of their applications early in the design stages. To do so we adopt and enrich different component based and UML-like modeling technologies like the Palladio Component Model and MODACloudML extending them in order to determine an optimized deployment in multi-cloud environments by introducing a new cloud specific meta-model. The integration of the new meta-model into state of the art modeling tools like Palladio Bench or Modelio allows software architects to use well known modeling approaches and specify a cloud specific deployment for their applications. In order to ease the portability of both the model and the application the meta-model uses three abstraction levels. The Cloud enabled Computation Independent Model (CCIM) allows to describe the application without any reference to specific cloud technologies or providers; the Cloud Provider Independent Model (CPIM) adds the specificity of some cloud technologies introducing concepts like Infrastructure and Platform as a Service (IaaS/PaaS) but still abstracts away the specificity of each particular provider; the Cloud Provider Specific Model (CPSM) adds all the details related to a particular cloud provider and the services offered allowing to automatize the deployment of the application and generate performance models that can be analyzed to assess the expected QoS of the application. High level architectural models of the application are then transformed into a Layered Queuing Network performance model that is analyzed with state of the art solvers like LQNS or LINE in order to derive performance metrics. The result of the evaluation can be used by

software architects to refine their design choices. Furthermore, the approach automates the exploration of deployment configurations in order to minimize operational costs of the cloud infrastructure and guarantee application QoS, in terms of availability and response time. In the IaaS context, as an example, the deployment choices analyzed by the tool are the size of instances (e.g. Amazon EC2 m3.xlarge) used to host each application tier and the number of replicas for each hour of the day. The problem of finding the optimal deployment configuration has been analyzed from a mathematical point of view and has been shown to be NP-hard. For this reason a heuristic approach has been proposed to effectively explore the space of possible deployment configurations. The heuristic approach uses a relaxed formulation based on M/G/1 queuing models to derive a promising initial solution that is then refined by means of a two level hybrid heuristic and validated against the LQN performance model. The proposed methodology has been validated by two industrial case study in the context of the MODAClouds project. A scalability and robustness analysis has also been performed and shows that our heuristic approach allows reductions in the cost of the solution ranging from 39% to 78% with respect to current best practice policies implemented by cloud vendors. The scalability analysis shows that the approach is applicable also to complex scenarios, with the optimized solution of the most complex instance analyzed being found in 16 minutes for a single cloud deployment and in 40 minutes for a multi-cloud scenario.

Contents

List of Figures

Chapter 1

Introduction

One of the most pervasive changes happened in recent years in the IT world is the appearance on the scene of *cloud* computing. The main feature of this new computing paradigm is its ability to offer IT resources and services in the same way fresh water or electric power is offered, as a utility. In a traditional environment in order to make use of a software system to address some business needs a company would need to acquire and manage the hardware infrastructure, different software stacks and, in many situations, develop their own software. Cloud computing changes this paradigm by offering all these elements as services that the user can acquire and release with high flexibility.

Cloud providers offer a variety of IT resources using the "*as a Service*" paradigm. Complex software systems that require multiple application stacks and different hardware resources can now be acquired, entirely or in parts, in matter of minutes. Hardware resources like computation nodes, storage space or network capacity are offered as *Infrastructure as a Service* (IaaS), software stacks that allows application developers to run their own code are offered as *Platform as a Service* (PaaS), finally, entire software system that can be directly used to provide some business value without the need of developing a new system are offered as *Software as a Service* (SaaS).

Since the early appearance of this technology in the market, many companies have decided to evolve their own infrastructure in order to embrace this new paradigm and the offering o cloud services, as well as the number of cloud providers, has grown quickly [38].

There are many advantages introduced by the adoption of the cloud technology,

among all one of the most important is the flexibility in the acquisition and decommission of software systems and the underlying infrastructure. This advantage is due to the ability of cloud providers to offer an almost unlimited amount of resources and a pricing policy that allow application developers to avoid long term commitments and pay only for resources they actually use. Another key advantage brought by the adoption of the cloud paradigm is the shift of responsibility in the management of the portion of the software system that is acquired from the cloud provider. If, for example, a company decides to decommission its own physical infrastructure in favor of a new infrastructure offered by a cloud provider all the maintenance operations required by the hardware and some software systems, like OS acquisition and update, are delegated to the cloud provider allowing the internal IT team of the company to focus on tasks that provide more value for the company.

Delegating management responsibility of part of the infrastructure to a third party, in this case a cloud provider, comes inevitably with a loss of control on the entire infrastructure. This change creates some new challenges for teams that were used to build entire software systems from the ground up. When faced with the selection of a cloud service the application developer has to take into consideration many new characteristics that he/she was probably not considering before. The wide variety of similar services, the lack of interoperability between APIs of services offered by different cloud providers and the lack of specific training for developers are just a few of the new challenges that the IT staff of a company has to face when considering a migration to a cloud infrastructure. Furthermore, the loss of control on the infrastructure exposes users to the variability of QoS offered by cloud providers. Usually providers address this issue by providing generic Service Level Agreements (SLAs) specifying their expected QoS level and providing discounts on future usage of their services in case the specified QoS is not met. Amazon EC2 SLA, for instances provides an availability of 99.95% of up-time in a month for VMs and in case this availability level is not met users are granted a discount on 10% on service cost. In many situations such a discount is not comparable to the possible loss generated by the downtime of the application. Many examples of cloud outages like the ones happened recently to Google Cloud [1] or Microsoft Azure [2] shows that availability concerns play a major factors in

[1] https://status.cloud.google.com/incident/compute/15045

[2] https://azure.microsoft.com/en-us/status/

moving a critical application to the cloud environment.

A solution to this problem comes from the wide variety of similar cloud services offered by other providers. If, as an example, the software architect thinks that the application under development is critical and requires an availability of 99.999% (also called 5-nines availability) he/she could replicate the application on two cloud providers, say Amazon AWS and Microsoft Azure, obtaining the required availability. Moreover, the use of multiple providers might allow the application developer also to redistribute the incoming workload in order to exploit differences in pricing in order to reduce the operational costs.

The contributions of this thesis are a methodology and a tool, to help software developers to build applications capable of exploiting the cloud infrastructure. In particular the work presented in this thesis tries to simplify the development process by providing software architects with a *meta-model* in order to describe possible deployments of their application over the cloud infrastructure. We then automate the QoS and cost analysis of such deployments in order to provide software architects with feedback on their deployment choices. Finally, we automate the generation of possible deployment configurations, possibly on multiple cloud providers, in order to minimize infrastructural costs and guarantee a desired QoS level.

Our work embraces the Model Driven Engineering (MDE) paradigm and makes use of models of the application specified at different levels of abstractions. Allowing the software architect to start by building simple component based models in a UML-like notation and then refine them adding information related to the desired cloud environment allows the development team to keep the focus on the functionality of the system their are building and delegate some of the architectural choices to the tool we have developed and integrated in the modeling environment.

Our approach targets the development team and in particular software architects, application developers and domain experts. The modeling paradigm that we embraced allow separation of concerns between these figures involved in the software development process. In the reminder of this thesis we will refer to this actors as *users* of our tool. In contrast the users of the cloud applications developed by using our approach and deployed on a cloud environment are mentioned as *final users* or *end users*. When the use of these terms might generate ambiguity we will speak directly of software architects or application developers.

The use of state of the art performance models and tools allows to provide to the development team estimation on the expected QoS of the system under development, in order to take informed decisions and adapt the design of the system early in the design stages avoiding complex and expensive re-factoring of the application.

The proposed approach allows designers to annotate theirs models with requirements related to the QoS of the application, like the expected response time of certain functionality or the minimum availability of the system and delegates to the tool the task of finding a deployment plan capable of fulfilling such constraints.

The deployment optimization strategy proposed in this thesis explores a huge and complex space of possible configurations by assigning to each component described in the model of the application a particular cloud service and analyze the behavior of the entire application in order to see if a particular choice of cloud services is capable of fulfilling the constraints. This search process takes also in to consideration the cost of the deployment solution and tries to minimize it.

The scientific literature shows some similar approaches that try to automate deployment decisions on component based systems but, to the best of our knowledge, this is the first approach that targets directly multi-cloud environments. In [49], Koziolek shows that due to the increasing size and complexity of software systems, architects have to choose from a combinatorially growing number of design and deployment alternatives. Different approaches have been developed to help architects explore this space with the help of automated tools like [10, 25] or [23, 49]. These approaches are presented in Chapter 2 help developers to analyze different design choices but do not address directly the cloud environment.

We argue that the problem becomes significantly more complex when considering the cloud services that can be employed for the execution of the application components. Traditionally the allocation problem has been considered independently from the design of the application but the possibility of exploiting different cloud services for different parts of the application has an impact on how the entire system works and makes the deployment problem even more relevant in the design phase.

If we consider even the simple example of a web application deployed on a single tier, we need to decide if we want to use a PaaS service to host our application code or to directly manage the platform used to run our code, say a Tomcat server. This choice directly affects the design and the development of the application. If we choose

to manage directly a Tomcat instance and deploy it on a Virtual Machine (VM) offered by Amazon, we still need to decide which type of VM we need to use and how many number of replicas of this machine according to the expected number of end user of our system. Using a high number of cheap VMs, like the m3.large in order to cope with a variable workload might seem a good strategy but the software stack needed to run our application might required more resources and, in this case using a smaller number of more powerful instances, like the c4.3xlarge, might be more convenient.

In a multi-cloud scenario this problem becomes even more complex because, beside making these decisions for both providers, we also have to define how the incoming workload is split among the providers. Since the performance and the price of resources offered by cloud providers might change with the time of the day, this problem is very dynamic and requires some automation to help the designer to generate possible configurations.

When we deal with a more realistic and complex application the development team is faced with deployment decisions and analyzing all the possible alternatives is a daunting tasks that calls for automation. The tool developed during this work, called SPACE4Cloud (System Performance and Cost Evaluation for Cloud), allows to automate the exploration of these design alternatives. SPACE4Cloud has been developed in the context of the MODAClouds FP7 IP [1] European project and constitutes one of the core tools of the MODAClouds IDE design time environment. Our approach takes into consideration QoS constraints that predicate on the response time, both on average and percentiles, constraints on the availability of the application and *service allocation constraints*. By service allocation constraints we mean those constraints that are related to the type of technology chosen to build the application and include minimum requirements on some characteristics of the cloud services required to host specific components, e.g., minimum amount of memory or cores, or limitations on the scalability of some service. Our approach differs from those already available in the literature, since it targets directly the cloud environment taking into consideration some peculiar features. Cloud environments are naturally shared by multiple users, the use of a shared infrastructure might lead to contention problems. To address this kind of behaviors we make use of a performance analysis tool called LINE that take into consideration variability in the characteristics of the processing resources by using

[1] www.modaclouds.eu

a statistical characterization (via Random Environment [29]). Web applications, like those developed in a cloud environment are also dynamic and the number of end users and the price of the cloud resources changes during the day. In many applications, the incoming workload shows a daily pattern, for this reason we introduce a time-dependent workload profile over a 24-hour time horizon, which leads to the solutions of 24 intertwined capacity allocation problems.

We first introduce the modeling paradigm proposed to apply the MDE principles in the context of cloud application development in Chapter 3. We also present an industrial case study that is used later on in the evaluation of the approach and throughout the thesis to clarify both modeling concepts and the optimization approach.

We then introduce the general design methodology and optimization strategy used to tackle the problem along with the architecture of the tool in Chapter 4. We then formalize the problem in Chapter 5 and we show it is equivalent to a class of NP-hard problems. These initial part of the work has been submitted for publication in IEEE Transactions on Software Engineering. In Chapter 6 we use a simplified performance model and a relaxed formulation of the problem in order to quickly derive a promising initial solution for the heuristic algorithm and in Chapter 7 we describe in details the main optimization algorithm used to explore the design space and derive the optimized deployment configuration. Details on the impact of the initial solution on the entire optimization procedure have been published in [17].

To validate our approach we have used two industrial case studies that show how software architects can benefit from using the early QoS analysis and deployment optimization provided by our work. We have also inspected how the complexity of the application under development affects the cost of the solutions obtained and the time required to execute the optimization with a scalability analysis, the results of this study is reported in Chapter 8. We compared our heuristic approach against common used threshold based heuristic that keep the utilization of the system below 60% or 80%. Using our heuristic we found optimized solutions with cost reductions ranging from 39% to 78%. The analysis also shows that the algorithm is both scalable and robust, the optimized solution for the most complex case was found in 36 minutes in a single cloud scenario and 42 minutes in multi-clouds. Robustness has been analyzed by repeating several times the optimization procedure during the scalability analysis and in the worst case the standard deviation of the time spent in the optimization is 18%

of the average execution time and the standard deviation of the cost of the solution is within 6% of the average solution cost. For what concerns the correctness of the QoS estimation with respect to the real model we relay on the extended literary work on performance prediction based on LQN starting from [36] that analyzes the accuracy of the QoS prediction for LQN models. With respect to the characterization of the parameters of the performance model used to evaluate application QoS we relay again on previous works like the one by Casale et al. [68] that presents several techniques to estimate application demands.

A discussion of the results achieved and an outline of future work are drawn in Chapter 9.

Chapter 9

Conclusions

"*That's all Folks!*"

Porky Pig

In this work we presented an approach that tries to simplify the process of migrating an application to the cloud by providing a methodology and a tool to support development teams in building new applications capable of running in a multi-cloud environment. We proposed a meta-model that describes cloud services and integrated it with well established modeling tools like Palladio and Modelio in order to allow application architects to specify configurations in cloud environments. We then automated the process of evaluating the QoS of the deployment configuration specified by the software architect allowing her/him to gain valuable insights on how the design reacts to different working conditions (e.g., variable incoming workload). This ability empowers the application architects to follow MDE principles and perform QoS and cost analyses early in the design stage allowing prompt modification of the architecture to tailor it better to the runtime environment. This ability has been shown by the first industrial case study by Softeam in which an early analysis of an initial architecture model revealed its inability to gracefully scale and support higher workloads. This discovery led to a re-design of part of the architecture leading to a system that could exploit better the scalability feature offered by cloud environments. A second industrial case study used the tool to evaluate different application deployments in a multi-cloud scenario.

We then focused on helping the application architect not only in the discovery of potential issues in the architecture or in a particular deployment configuration, but also in deriving an optimized deployment that minimize the cost of using cloud services and provide QoS guarantees at the same time. To derive this configuration we designed an optimization heuristic that

effectively explores a wide space of possible configurations. We first formalized the problem from a mathematical point of view and showed it to be NP-hard. We then used M/G/1 queuing network models to derive a closed form formulation of the application response time and used this model to solve a relaxation of the original problem and derive a promising initial solution. This solution is then modified by our hybrid heuristic that makes use of a more accurate performance model, i.d. LQN models, to evaluate the feasibility of the application against user defined constraints.

We evaluated the applicability of the proposed approach to complex models by means of a scalability analysis that showed how the solution derived by means of our heuristic algorithm outperform those derive by policies currently used by practitioners providing an average reduction in the cost of the deployment around 55%. The scalability analysis also showed that the approach can be effectively applied to complex problems, since the optimized solution was obtained in around 40 minutes in for the most complex model considered.

The main threat to the validity of our approach is the lack of accurate data on the performance of cloud services. The optimization approaches uses a Resource Database to update the performance model with the characteristics of the cloud resource under analysis, as show in Section 4.2. While some of the information stored in this database are provided publicly by cloud providers (e.g. the cost of using such a resource or the number of cores of a particular VM type), other parameters are unknown and have to be estimated by benchmarking such resources. Furthermore the service offer of main cloud providers change very frequently both in terms of performance upgrades or cost reductions. Since a complete benchmark campaign was not feasible, we decided to integrate in our approach the results of the ARTIST[1] European project which provide benchmarking information of many cloud resources.

From the optimization point of view, we allowed users to define constraint on the response time of the application. Current best practices use constraints on resource utilization since as long as the utilization is low the response time does not change significantly. Using directly the response time might be effective only if the constraint is so high that high utilization can be tolerated. To overcome this limitation we added the possibility to take into account resource utilization constraints as well so that if both type of constraints are defined, one will dominate the other. Finally the user can specify response time constraint on a functionality offered to the end user of the application and utilization constraints on part of the resources used to provide such a functionality. In such a scenario high utilization might be tolerated on some resourced involved in the processing of the user request while key components might be kept under a utilization constraint.

[1] http://www.artist-project.eu/collaboration

Our approach has been particularly tailored to support the analysis and optimization of common web based applications but the modeling approach and the optimization techniques presented in this thesis can be extended to address different deployment problem.

An extension that we have considered takes into account the use of a mixed cloud environment in which a private infrastructure is used to process most of the workload and, when the private infrastructure is not capable of accepting more users, a public cloud is used to replicate the entire system.

In our work we considered application deployment on multiple clouds in which the entire system has been replicated on all the available providers. Another possible research line can consider a more flexible deployment solution where application tiers are deployed on different cloud providers. The analysis of such a deployment configuration should take into consideration the peculiarity of the technology used to interconnect the application tiers, which we expect will be enabled by progress in the area of network design. The deployment configuration we considered in this thesis allows to exploit high performance networks available in cloud providers datacenter and avoid the increased complexity in the analysis.

Recent years showed the appearance of Big Data frameworks and applications. These applications perform complex analysis of high volumes of data in order to extract insights with business value. The complexity of the frameworks used to process this kind of data and the size of the infrastructure required to perform such analysis make the cloud environment the default choice for many companies that want to exploit this new technology. The approach proposed in this thesis could be adapted to take into consideration the peculiarity of these applications and frameworks by the use of a more tailored performance model.

If we consider current technology trends, the raise of container based hosting services like Amazon EC2 Container Service (ECS)[1] or Google Container Engine[2] eases the use of component based approaches in the development of cloud applications. In this context an optimization technique that derive container characteristics like the container size could greatly help architects to speed up the development and testing of their application architectures.

Finally, adding the link between the runtime application execution and the models built at design time by the use of automated tools and an appropriate monitoring technology will enable a full DevOps approach. The main idea is to use real monitoring data to refine application performance parameters (e.g., service demands, number of users at peak, switch probabilities in end user behaviour, etc.). In this vision development decisions and operational aspects are considered together and the barrier between design time and runtime can be dissolved.

[1] https://aws.amazon.com/it/blogs/aws/cloud-container-management/
[2] https://cloud.google.com/container-engine/

Bibliography

[1] Argouml - http://argouml.tigris.org/. 10

[2] Magicdraw - http://www.nomagic.com/products/magicdraw.html. 10

[3] Modeling software kitt - https://moskitt.gva.es. 10

[4] Omg model-driven architecture. 11

[5] Rational software architect - http://www-03.ibm.com/software/products/it/ratisoftarch. 10

[6] Staruml - http://staruml.io. 10

[7] Uml designer - http://www.umldesigner.org/. 10

[8] Upupa - http://www.modelexecution.org/. 10

[9] Bernardetta Addis, Danilo Ardagna, Barbara Panicucci, Mark S. Squillante, and Li Zhang. A hierarchical approach for the resource management of very large cloud platforms. *IEEE Trans. Dependable Sec. Comput.*, 10(5):253–272, 2013. 123

[10] A. Aleti, B. Buhnova, L. Grunske, A. Koziolek, and I. Meedeniya. Software architecture optimization methods: A systematic literature review. *Software Engineering, IEEE Transactions on*, PP(99):1–1, 2013. 4, 11

[11] Aldeida Aleti, Stefan Stefan Björnander, Lars Grunske, and Indika Meedeniya. Archeopterix: An extendable tool for architecture optimization of aadl models. In *Proc. of Workshop MOMPES 2009*, 2009. 15

[12] Amazon Inc. AWS Elastic Beanstalk. http://aws.amazon.com/elasticbeanstalk/. 71

[13] Nicolas Ferry Stepan Seycek Elisabetta Di Nitto Antonin Abherve, Marcos Almeida. Modacloudml ide final version. Public deliverable, 2015. 22, 25, 31

[14] D. Ardagna, B. Panicucci, M. Trubian, and L. Zhang. Energy-aware autonomic resource allocation in multitier virtualized environments. *Services Computing, IEEE Transactions on*, 5(1):2–19, 2012. 68, 123

[15] D. Ardagna and B. Pernici. Adaptive service composition in flexible processes. *Software Engineering, IEEE Transactions on*, 33(6):369–384, June 2007. 66, 124

[16] Danilo Ardagna, Sara Casolari, Michele Colajanni, and Barbara Panicucci. Dual time-scale distributed capacity allocation and load redirect algorithms for cloud systems. *Journal of Parallel and Distributed Computing*, 72(6):796 – 808, 2012. 137

[17] Danilo Ardagna, GiovanniPaolo Gibilisco, Michele Ciavotta, and Alexander Lavrentev. A multi-model optimization framework for the model driven design of cloud applications. In *SBSE 2014 Proc.* 2014. 6, 61, 75, 137

[18] Danilo Ardagna and Barbara Pernici. Adaptive service composition in flexible processes. *IEEE Trans. Software Eng.*, 33(6):369–384, 2007. 137

[19] George Kousiouris Danilo Ardagna Athanasia Evangelinou, Michele Ciavotta. A joint benchmark-analytic approach for design-time assessment of multi-cloud applications. In *Proceedings of the 1st International Conference on Cloud Computing, Information Technology, Big Data and Big Data Management*, 2015. 56

[20] S. Balsamo, A. Di Marco, P. Inverardi, and M. Simeoni. Model-based performance prediction in software development: A survey. *Software Engineering, IEEE Transactions on*, 30(5):295–310, 2004. 11

[21] Matthias Becker, Steffen Becker, and Joachim Meyer. SimuLizar: Design-Time Modelling and Performance Analysis of Self-Adaptive Systems. In *Proceedings of Software Engineering 2013 (SE2013), Aachen*, 2013. 11

[22] Matthias Becker, Markus Luckey, and Steffen Becker. Performance analysis of self-adaptive systems for requirements validation at design-time. In *Proceedings of the 9th International ACM Sigsoft Conference on Quality of Software Architectures*, QoSA '13, pages 43–52, New York, NY, USA, 2013. ACM. 11

[23] S. Becker, H. Koziolek, and R. Reussner. The palladio component model for model-driven performance prediction. *Journal of Systems and Software*, 82(1):3–22, 2009. 4, 11, 16, 22

[24] Christian Blum, Jakob Puchinger, Günther R Raidl, and Andrea Roli. Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing*, 11(6):4135–4151, 2011. 20

[25] F. Brosig, P. Meier, S. Becker, A. Koziolek, H. Koziolek, and S. Kounev. Quantitative evaluation of model-driven performance analysis and simulation of component-based architectures. *Software Engineering, IEEE Transactions on*, 41(2):157–175, Feb 2015. 4, 12

[26] Omid Bushehrian. The application of fsp models in automatic optimization of software deployment. In Khalid Al-Begain, Simonetta Balsamo, Dieter Fiems, and Andrea Marin, editors, *Analytical and Stochastic Modeling Techniques and Applications*, volume 6751 of *Lecture Notes in Computer Science*, pages 43–54. Springer Berlin Heidelberg, 2011. 18

[27] Claudia Canali and Riccardo Lancellotti. Exploiting ensemble techniques for automatic virtual machine clustering in cloud systems. *Automated Software Engineering*, 21(3):319–344, 2014. 71

[28] G. Canfora, M. Di Penta, R. Esposito, and M.L. Villani. An approach for qos-aware service composition based on genetic algorithms. In *Proceedings of the 2005 conference on Genetic and evolutionary computation*, pages 1069–1075. ACM, 2005. 17

[29] Giuliano Casale and Mirco Tribastone. Modelling exogenous variability in cloud deployments. *SIGMETRICS Perform. Eval. Rev.*, 40(4):73–82, April 2013. 6, 39, 62

[30] Benoît Colson, Patrice Marcotte, and Gilles Savard. An overview of bilevel optimization. *Annals of operations research*, 153(1):235–256, 2007. 74

[31] Vittorio Cortellessa, Antinisca Di Marco, Romina Eramo, Alfonso Pierantonio, and Catia Trubiani. Approaching the model-driven generation of feedback to remove software performance flaws. In *EUROMICRO-SEAA*, pages 162–169. IEEE Computer Society, 2009. 14

[32] MauroLuigi Drago, Carlo Ghezzi, and Raffaela Mirandola. Qvtr2: A rational and performance-aware extension to the relations language. In Juergen Dingel and Arnor Solberg, editors, *Models in Software Engineering*, volume 6627 of *Lecture Notes in Computer Science*, pages 328–328. Springer Berlin Heidelberg, 2011. 13

[33] B.K. Eames, S.K. Neema, and R. Saraswat. Desertfd: a finite-domain constraint based tool for design space exploration. *Design Automation for Embedded Systems*, 14(1):43–74, 2010. 19

[34] Nicolas Ferry, Alessandro Rossini, Franck Chauvel, Brice Morin, and Arnor Solberg. Towards model-driven provisioning, deployment, monitoring, and adaptation of multi-cloud systems. In Lisa O'Conner, editor, *IEEE CLOUD 2013 Proc.*, pages 887–894. IEEE Computer Society, 2013. 10

[35] G. Franks, T. Al-Omari, M. Woodside, O. Das, and S. Derisavi. Enhanced modeling and solution of layered queueing networks. *Software Engineering, IEEE Transactions on*, 35(2):148–161, March 2009. 22, 62

[36] Greg Franks and Murray Woodside. Performance of multi-level client-server systems with parallel service operations. In *Proceedings of the 1st International Workshop on Software and Performance*, WOSP '98, pages 120–130, New York, NY, USA, 1998. ACM. 7

[37] Sören Frey, Florian Fittkau, and Wilhelm Hasselbring. Search-based genetic optimization for deployment and reconfiguration of software in the cloud. In *Proceedings of the 2013 International Conference on Software Engineering*, ICSE '13, pages 512–521, Piscataway, NJ, USA, 2013. IEEE Press. 17

[38] Gartner Group. Hype Cycle for Cloud Computing, 2014. https://www.gartner.com/doc/2807621/hype-cycle-cloud-computing-, 2014. 1

[39] Fred Glover. Tabu search: part i. *ORSA Journal on computing*, 1(3):190–206, 1989. 60

[40] Fred Glover and Manuel Laguna. Tabu search. In *Handbook of Combinatorial Optimization*, pages 2093–2229. Springer, 1999. 84

[41] Daniel Gmach, Jerry Rolia, and Ludmila Cherkasova. Selling t-shirts and time shares in the cloud. In *Proceedings of the 2012 12th IEEE/ACM International Symposium on*

Cluster, Cloud and Grid Computing (Ccgrid 2012), CCGRID '12, pages 539–546, Washington, DC, USA, 2012. IEEE Computer Society. 71

[42] V. Grassi, R. Mirandola, and A. Sabetta. From design to analysis models: a kernel language for performance and reliability analysis of component-based systems. In *Proc. of the Workshop WOSP 2005*, 2005. 11

[43] A. Gunka, S. Seycek, and Kuhn H. Moving an application to the cloud an evolutionary approach. In *MultiCloud*, 2013. 116

[44] Ernest Friedman Hill. *Jess in Action: Java Rule-Based Systems*. Manning Publications Co., Greenwich, CT, USA, 2003. 13

[45] E.K. Jackson, E. Kang, M. Dahlweid, D. Seifert, and T. Santen. Components, platforms and possibilities: towards generic automation for mda. In *Proceedings of the tenth ACM international conference on Embedded software*, pages 39–48. ACM, 2010. 19

[46] Robert G Jeroslow. The polynomial hierarchy and a simple model for competitive analysis. *Mathematical programming*, 32(2):146–164, 1985. 74

[47] Anne Koziolek. *Automated Improvement of Software Architecture Models for Performance and Other Quality Attributes*. PhD thesis, Institut für Programmstrukturen und Datenorganisation (IPD), Karlsruher Institut für Technologie, Karlsruhe, Germany, July 2011. 16, 17

[48] Anne Koziolek, Danilo Ardagna, and Raffaela Mirandola. Hybrid multi-attribute qos optimization in component based software systems. *Journal of Systems and Software*, 86(10):2542 – 2558, 2013. 17, 61

[49] Anne Koziolek, Heiko Koziolek, and Ralf Reussner. PerOpteryx: Automated Application of Tactics in Multi-objective Software Architecture Optimization. In *QoSA 2011 Proc.*, QoSA-ISARCS '11, pages 33–42, New York, NY, USA, 2011. ACM. 4

[50] Anne Koziolek and Ralf Reussner. Towards a generic quality optimisation framework for component-based system models. In *Proceedings of the 14th International ACM Sigsoft Symposium on Component Based Software Engineering*, CBSE '11, pages 103–108, New York, NY, USA, 2011. ACM. 16

[51] Heiko Koziolek. Performance evaluation of component-based software systems: A survey. *Performance Evaluation*, 67(8):634–658, 2010. 11

[52] Dara Kusic, Jeffrey O. Kephart, James E. Hanson, Nagarajan Kandasamy, and Guofei Jiang. Power and performance management of virtualized computing environments via lookahead control. *Cluster Computing*, 12(1):1–15, 2009. 137

[53] Alexander Lavrentev. An optimization approach for cloud providers selection and capacity allocation for multi-iaas systems. Master's thesis, Politecnico di Milano, 2013. 79

[54] E.D. Lazowska, J. Zahorjan, G.S. Graham, and K.C. Sevcik. *Quantitative system performance: computer system analysis using queueing network models*. Prentice-Hall, Inc., 1984. 109

[55] Rui Li, Ramin Etemaadi, Michael T. M. Emmerich, and Michel R. V. Chaudron. An evolutionary multiobjective optimization approach to component-based software architecture design. In *Proc. of Congress, CEC 2011*, 2011. 15

[56] Adam Lipowski and Dorota Lipowska. Roulette-wheel selection via stochastic acceptance. *CoRR*, abs/1109.3627, 2011. 20

[57] Grzegorz Loniewski, Etienne Borde, and Emilio Insfran. Towards a model driven refinement process through architecture evaluation. In *Proceedings of the Fourth International Workshop on Nonfunctional System Properties in Domain Specific Modeling Languages*, NFPinDSML 2012, pages 4:1–4:6, New York, NY, USA, 2012. ACM. 14

[58] HelenaR. Loureno, OlivierC. Martin, and Thomas Sttzle. Iterated local search: Framework and applications. In Michel Gendreau and Jean-Yves Potvin, editors, *Handbook of Metaheuristics*, volume 146 of *International Series in Operations Research & Management Science*, pages 363–397. Springer US, 2010. 60

[59] Anne Martens, Heiko Koziolek, Steffen Becker, and Ralf Reussner. Automatically improve software architecture models for performance, reliability, and cost using evolutionary algorithms. In *Proc. of Conference WOSP/SIPEW 2010*, 2010. 12

[60] Raffaela Mirandola, Pasqualina Potena, and Patrizia Scandurra. Adaptation space exploration for service-oriented applications. *Science of Computer Programming*, 80, Part B:356 – 384, 2014. 18

[61] S. Neema, J. Sztipanovits, G. Karsai, and K. Butts. Constraint-based design-space exploration and model synthesis. In *Embedded Software*, pages 290–305, 2003. 19

[62] Qais Noorshams, Anne Martens, and Ralf Reussner. Using quality of service bounds for effective multi-objective software architecture optimization. In *Proceedings of the 2Nd International Workshop on the Quality of Service-Oriented Software Systems*, QUASOSS '10, pages 1:1–1:6, New York, NY, USA, 2010. ACM. 16

[63] OMG. *UML Profile for Schedulability, Performance, and Time Specification*, 2005. 9

[64] OMG. A uml profile for marte: Modeling and analysis of real-time embedded systems, 2008. 9

[65] Mohamed Ouzineb, Mustapha Nourelfath, and Michel Gendreau. Tabu search for the redundancy allocation problem of homogenous series-parallel multi-state systems. *Rel. Eng. & Sys. Safety*, 93(8):1257–1272, 2008. 17

[66] Trevor Parsons and John Murphy. Detecting performance antipatterns in component based enterprise systems. *Journal of Object Technology*, 7(3):55–91, 2008. 13

[67] J.F. Perez and G. Casale. Assessing sla compliance from palladio component models. In *Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), 2013 15th International Symposium on*, pages 409–416, Sept 2013. 22, 62

[68] Juan F. Pérez, Giuliano Casale, and Sergio Pacheco-Sanchez. Estimating computational requirements in multi-threaded applications. *IEEE Trans. Software Eng.*, 41(3):264–278, 2015. 7

[69] Marian Petre. Uml in practice. In *Proceedings of the 2013 International Conference on Software Engineering*, ICSE '13, pages 722–731, Piscataway, NJ, USA, 2013. IEEE Press. 10

[70] T. Saxena and G. Karsai. Mde-based approach for generalizing design space exploration. *Model Driven Engineering Languages and Systems*, pages 46–60, 2010. 19

[71] SOFTEAM. Modelio. The open source modeling environment. https://www.modelio.org, 2015. 9, 23

[72] T Stützle. Local search algorithms for combinatorial problems. *Darmstadt University of Technology PhD Thesis*, 1998. 20

[73] E-G Talbi. A taxonomy of hybrid metaheuristics. *Journal of heuristics*, 8(5):541–564, 2002. 20

[74] El-Ghazali Talbi. *Metaheuristics - From Design to Implementation*. Wiley, 2009. 12, 20, 84, 86, 101

[75] A.P.A. van Moorsel and K. Wolter. Analysis and algorithms for restart. In *Quantitative Evaluation of Systems, 2004. QEST 2004. Proceedings. First International Conference on the*, pages 195–204, Sept 2004. 84

[76] Andreas Wolke and Gerhard Meixner. Twospot: A cloud platform for scaling out web applications dynamically. In *ServiceWave 2010 Proc.*, 2010. 134

[77] M. Woodside, D.C. Petriu, D.B. Petriu, H. Shen, T. Israr, and J. Merseguer. Performance by unified model analysis (puma). In *Proc. of Workshop WOSP 2005*, 2005. 12

[78] Jing Xu. Rule-based automatic software performance diagnosis and improvement. In *Proc. of Workshop WOSP 2008*, 2008. 13

[79] Xiaoyun Zhu, Donald Young, Brian J. Watson, Zhikui Wang, Jerry Rolia, Sharad Singhal, Bret Mckee, Chris Hyser, Daniel Gmach, Robert Gardner, Tom Christian, and Ludmila Cherkasova. 1000 islands: An integrated approach to resource management for virtualized data centers. *Cluster Computing*, 12(1):45–57, March 2009. 134, 138