Architectural Run-Time Models for Operator-in-the-Loop Adaptation of Cloud Applications

Robert Heinrich[1], Reiner Jung[2], Eric Schmieders[3], Andreas Metzger[3],
Wilhelm Hasselbring[2], Ralf Reussner[1], and Klaus Pohl[3]

[1]Karlsruhe Institute of Technology, Germany, {heinrich, reussner}@kit.edu
[2]Kiel University, Germany, {reiner.jung, hasselbring}@email.uni-kiel.de
[3]Paluno, University Duisburg-Essen, Germany, {eric.schmieders, andreas.metzger, klaus.pohl}@paluno.uni-due.de

***Abstract*—Building software systems by composing third-party cloud services promises many benefits. However, the increased complexity, heterogeneity, and limited observability of cloud services brings fully automatic adaption to its limits. We propose architectural run-time models as a means for combining automatic and operator-in-the-loop adaptations of cloud services.**

I. Introduction

Two major trends continue to significantly shape the way software engineers build and maintain future long-living software systems: (I) Software systems will be built by selecting, configuring, and composing third-party software-defined services, providing access to application software, cloud compute and storage facilities, internet-connected things, as well as data. Software-defined services separate ownership, maintenance and operation from usage of software. Service users do not need to deploy and run software on their own. They use software executed by third parties that can be remotely accessed through service interfaces [1]. (II) Software will be deployed on virtualized, remote compute infrastructures, such as cloud infrastructures, software-defined network devices, as well as Internet-of-Things devices. These deployment models result in software that will increasingly be deployed on hardware resources and on top of middleware that is owned and operated by third parties. The use of third party services and cloud infrastructures promises many benefits, such as flexibility, scalability, reusability and economic use of resources. Yet, the increased complexity and unprecedented heterogeneity of cloud services as well as their limited observability, will bring fully automatic adaptation to its limits. The distribution of cloud services across different legal systems, e.g., the EU and the USA, with different regulations for privacy and data proliferation, apply restrictions on the deployment and potential adaptations. The complexity of effects of external events lead to major challenges (e.g., see [2]). For instance, business decisions, such as sales campaigns, lead to an intensive application usage which cannot be anticipated by the system by monitoring data alone. Other effects, such as weather conditions and economic turmoil, may have an impact on customer behavior resulting in varying application utilization. Fully capturing and formalizing the knowledge required to take autonomic adaptation decisions may just become infeasible or may face prohibitive costs.

This work was partially supported by the DFG (German Research Foundation) under the Priority Programme SPP1593: Design For Future – Managed Software Evolution.

While, in recent years, the research focus for adaptation was on driving further its automation (e.g., see [3]), it becomes evident that we need to allow operators (humans) to engage in the adaptation process. An open question is how to facilitate such operator-in-the-loop adaptation. This paper takes a model-based stance by proposing architectural run-time models as a means for combining automatic and operator-in-the-loop adaptations of cloud services. A run-time model constitutes an abstraction of the software system of consideration (see [4]). Such models can be used as descriptive models, visualizing the currently executing system. In addition, they may also be used as prescriptive models, specifying the envisioned future state of the software system.

In this paper we present first results towards supporting the automatic and operator-in-the-loop adaptation of cloud-based systems by leveraging architectural run-time models for planning and executing adaptations, while taking into account multiple quality criteria. Our envisioned solution complements the iObserve approach by extending it to the planning and execution phases of the MAPE loop (iObserve so far focused on the monitoring and analysis phases [5], [6]). A set of adaptation candidates is computed and in case where selection among those candidates is not feasible automatically, alternatives are presented to the operator for decision making. This facilitates an operator to engage with the adaptation process in three distinct areas. Firstly, by allowing introducing external workload scenarios which are tailored to fit predictions regarding a specific external event. Secondly, by providing the means to observe, evaluate, assess and manipulate scenario candidates visually. Thirdly, by supporting the operator in translating adaptation candidates to actual adaptation tasks.

The remainder of the paper is structured as follows. Sec. II introduces iObserve as a basis for this paper's contributions. Sec. III discusses limitations of current adaptation mechanisms in the literature. Sec. IV sketches the main ideas and concepts for using architectural run-time models for operator-in-the-loop adaptation of cloud services. The paper concludes in Sec. V.

II. The iObserve Approach

iObserve [7] addresses the challenges of distributed cloud-based software systems by following the widely adopted MAPE control loop model for adaptation and extends it with shared models to ease the transition between adaptation and evolution. For adaptation, the state of the software system is determined by monitoring, analyzed to detect anomalies and predict deviations, used as input for planning the adaptation to mitigate anomalies, and finally to execute the adaptation. iObserve supports monitoring and analysis of cloud-based software systems through architectural run-time models [5] to facilitate the automated analysis, as well as the human inspection to detect, in particular, performance and privacy anomalies. In our work we focus on performance, privacy and costs as especially in the cloud context these aspects are closely interrelated. The application usage impacts on the application's performance. Continuously appraised elasticity rules trigger the migration and replication of cloud application's software components among geographically distributed data centers. Both, migration and replication, may increase performance yet lead to violation of privacy policies and increasing costs. The architectural run-time models are derived and updated or reverse engineered via dynamic analysis of monitoring data [8]. As an umbrella to integrate design-time models, code generation, monitoring, analysis, and run-time model update, iObserve introduced a concise megamodel [5] which is divided into four sections defined by two dimensions: one for design-time vs. run-time, and one for model vs. implementation level. At design-time the megamodel shows how an application model including an architectural model is combined with our model-driven monitoring approach [8] and used for code generation. On the run-time side, monitoring data which relates to implementation artifacts are associated with run-time application model elements based on a correspondence model [9]. The megamodel builds upon an aspect-oriented modeling approach [10]. Various analyses are performed on the monitoring data and used to update the application model. Further description of iObserve, background information and technical details are given in [5].

iObserve currently is focused to monitoring and analysis of cloud-based software systems. Addressing planing and execution phases of the MAPE control loop requires facing various challenges due to the complex and heterogeneous nature of cloud services. While different directions of further development are possible, we consider the following three pillars as next steps for evolving iObserve.

First, the distribution of cloud services across several legal systems with varying privacy regulations requires putting special emphasis on privacy-driven planning. Second, the complexity of cloud services and the complexity of effects of external events lead to major challenges in finding an appropriate system design. Therefore, we put special emphasis on design space exploration to address multiple quality criteria (esp. performance, privacy and cost) based on models to be explored for solutions. Third, restrictions in fully automatic adaptation need to allow human operators to engage in the adaptation process. Therefore, we put special emphasis on operator-in-the-loop adaptation to overcome limitations of automatic adaptation. For all three pillars architectural run-time models are key artifacts to address these challenges.

III. State of the Art

This section discusses existing work and its limitations in context of the three pillars of the envisioned approach.

Privacy-driven planning, as an adaptation activity during run-time, is an emerging field of research. It combines cloud characteristics such as elasticity and shared ownership. Existing research tackles the enforcement of privacy policy compliance from two different angles – privacy-by-design and privacy audits. Further, planning techniques have been developed for quality attributes such as privacy and reliability. Privacy-by-design employs two main concepts: access control and privacy-aware deployment and elasticity rules. Research on access control mechanisms (e.g. [11]) investigates how to equip components with mechanisms that permit or grant data access after matching the client characteristics with the data policies. However, access control mechanisms face difficulties in the cloud whenever the controlled component is migrated or replicated across data centers (e.g., a database). Elasticity and deployment rules are typically defined during design-time. However, rules that implement data geo-location policies have to be defined considering the data stored by a virtual machine as well as the data, which may be transferred to it. Yet, this information is not available during design-time, and thus rules defined during design-time may not accurately capture geo-location constraints at run-time.

Approaches for privacy audits of cloud services (e.g. [12]) use host geo-location and provable data possession to evaluate the compliance of privacy regulations. Host geo-location can be checked based on ping round trip times for a service interface utilizing different ping origins. This allows to decide whether a service interfaces reside at geo-locations excluded in privacy policies. However, the software components behind the service interfaces might be migrated or replicated, while the service interface remains at the same geo-location. Fundamentals for facilitating privacy-driven planning for cloud adaptations has been proposed in [13], [14]. Further, existing work focuses on the identification of privacy violations and does not cover planning or adaptation aspects.

Quality aware planning techniques employ the phases of the MAPE loop to adapt cloud applications as soon as, e.g., performance issues emerge. These techniques exploit the proportional relation between allocated hardware and performance. For instance, calibrated descriptive run-time models [15] are used in performance predictions to steer the planning activities and adjust the number of hardware nodes to reach the desired performance. However, resolving the root causes of privacy violations is inherently different from resolving performance issues. While adding additional hardware may increase system performance it will not, e.g., avoid personal data being moved to excluded geo-locations.

To summarize, existing work on privacy-driven-planning neglects two essential characteristics of cloud applications, (1) the dynamic migration and replication of cloud application components (cloud elasticity) across data centers [16]; (2) the limited control and visibility of cloud elasticity by cloud customers (shared ownership).

Design space exploration: Many cloud based applications serve thousands if not millions of users (e.g, 425 million users for gmail) whose behavior cannot be fully understood at design-time, because their behavior may change over time due to a wide variety of external and internal events, such as seasonal sales and campaigns. Furthermore, service offers and cloud products change over time in performance and cost. In combination with privacy concerns, this yields a complex problem space where multiple solutions can be computed forming a design space. Design space exploration requires several different input models describing various aspects of the design space, in particular user behavior, workload intensity, cloud profiles covering cost and resource descriptions, and privacy concerns. These models are then explored for solutions.

Workload characterization approaches used for adaptive systems and workload generation [17] focus on the construction of workload intensity pattern composed of statistical functions and recorded request sets. However, the intensity alone is not sufficient to understand the workload and predict future workloads. Therefore, a wide variety of methods, including clustering and Markov chain models of service utilization [18] are used to model user behavior and workloads [19]. These models can be derived from observation data [5] and must reflect the business impact on system usage [20]. However, the discrimination of different user behaviors and the aggregation of similar behaviors is complicated. The BEAR approach [21] addresses the construction and analysis of such behavior models. However, BEAR focuses on the engineer and the use of the analysis at design-time, e.g., in an evolution step. Furthermore, its user distinction is based on IP addresses which does not work well in the present condition of the Internet, especially for mobile devices, where many users share the same IP address based on network address translation.

Cloud profiles describe various properties of cloud services including resources and cost, like in the CloudMIG approach [22], [23]. While CloudMIG originates in the software migration domain, its cloud profiles can also be used in the adaptation context. Based on these models, design space exploration can be performed. An overview of design space exploration methods is given in [24]. Basically, research in this area can be distinguished into three different groups. Rule-based approaches can only change models according to predefined rules and are limited to single quality attributes. Specialized optimization approaches only consider predefined degrees of changes. Meta heuristic-based approaches use degrees of freedom in models but mostly do not enable trade-off decisions. CDOXplorer [25] employs genetic algorithms and simulation to assess the fitness of cloud development options based on cloud profiles.

To summarize, existing work on design space exploration neglects the differentiation of user behavior and trade-off decisions between different quality attributes.

Operator-in-the-loop adaptation: In the operation of software systems, the main goal was to automate adaptation and eliminating human involvement. To the best of our knowledge, the combination of automatic and operator-in-the-loop adaptation is not covered in related work. However, the increasing complexity of systems require to address this subject. Hence, we rely on other research areas as source for operator-in-the-loop approaches. Combined automatic and operator adaptation is addressed in so-called supervisory control and data acquisition (SCADA) systems [26] designed to control and operate large scale distributed industrial plants and infrastructure networks, such as gas, water, and power supply. In all these systems, operators are included in the control process, as complete automation may result in undesirable states.

In the software engineering domain, recommendation systems [27] are used for software development and evolution, as well as, other design-time tasks. They use source code, version history and issue tracking systems, as corpus to derive knowledge from. For example, source code acquired from other projects utilizing the same set of libraries and frameworks are analyzed to recommend methods calls [28] and even the most suitable method parameters [29]. Developers can be supported by recommendation systems, like Hipikat [30], in selecting the correct artifacts of a software project according to the assigned maintenance task. Recommendation systems are also used for architecture comprehension [31], software migration [22], and performance optimizations [32]. Especially in the cloud context, performance and pricing are important aspects to be considered when selecting specific cloud providers for a software system. Therefore, approaches, like CloudAdvisor [33] and CloudMIG [22], support multi-objective optimization. Supplemental, ExplorViz [34], [35] provides a 3D live visualization of the system architecture based on monitoring data to support the human operator to perform changes when adaptations exceed automated planing routines. KAMP [36] supports human operators by semi-automatically analyzing the change propagation in architecture models caused by a given change request, such as replace a database [37] or split an interface [38] for solving a performance bottleneck.

To summarize, while we could not find combinations of automatic and operator-in-the-loop adaptation, SCADA and recommendation systems provide foundations to involve humans in the control loop. Furthermore, there are approaches for architecture visualization and architecture-based change propagation analysis that support operators in evolving the system.

IV. Addressing the Challenges

Addressing aforementioned challenges requires building out a platform for supporting automatic and operator-in-the-loop adaptation processes as depicted in Fig. 1. The cloud application is instrumented with monitoring probes [8]. The *Monitor* phase exploits information from probes to keep the descriptive architectural run-time models causally connected

with the underlying cloud application and infrastructure [5]. Architectural run-time models are used during the *Analyze* phase to determine any problems or deviations and thus triggering the need for an adaptation. Problems may be potential privacy violations or performance bottlenecks. Further, root causes that led to the problem are identified.

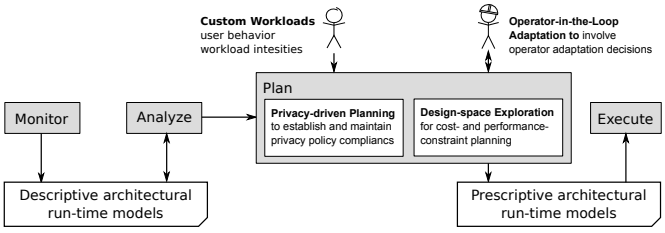

Fig. 1. Illustration of the planning and execution phases in context of the MAPE loop of our operator-in-the-loop adaptation approach.

If any privacy or performance issues are identified, adaptation candidates are generated and specified as candidate prescriptive run-time models during the *Plan* phase. Due to the different nature of privacy and performance, planning is performed by distinct techniques. The generated candidates aim for solving the root cause that led to the raised issues. As part of the design space exploration, candidate models are evaluated and ranked with respect to performance, usage, feasibility, and realization effort. Subsequently, these models are operationalized by deriving concrete tasks of an adaptation plan for the *Execute* phase [36]. When selecting one concrete model among the candidates or deriving an adaptation plan cannot be done fully automatically, the human operator (cf. Fig 1) choses among the presented adaptation alternatives. In cases where no candidate could be created, e.g., due to lack of information or criticality of decision, the operator will also be involved. Realizing this approach requires going beyond the state of the art in the following areas.

Privacy-driven planning requires algorithms and techniques for generating, evaluating, and determining degrees of freedom for adaptation that allow to resolve or mitigate privacy violations. They are defined around three key facets: First, identification and specification of appropriate cloud adaptation mechanisms which need to be included in the descriptive run-time models in order to empower the adaptation routines for carrying out privacy-aware adaptations during run-time. Second, constraint-based generation of adaptation alternatives under consideration of the available mechanisms. And third, impact analysis of privacy related adaptations on multiple application characteristics.

Design space exploration is based on multiple criteria representing different dimensions for architectural adaptations of cloud applications during design-time and exploit this specification for generating, evaluating, and implementing design alternatives during run-time.

First, to determine the different degrees of freedom in the design-space, several input models are required beside the descriptive architectural run-time model, including cloud profiles and the workload characterization models. While iObserve already observes application behavior and aggregates the information in usage models [9], these are insufficient to describe workloads in detail reducing its quality to predict future workloads. Therefore, we will extend the behavior model generation in iObserve to support behavior mixes based on different detected behavior patterns and combine this with distinct workload intensity profiles. In our approach, we will utilize data mining and data clustering tools (cf. [19]) in combination with graph based selectors to separate different user behaviors and relate workload intensities to these behaviors.

Second, to be able to perform adaptation, the adaptation plans are translated into detailed adaptation tasks realizing the transformation of the present architecture into the target architecture. For task identification we build upon KAMP [36].

Operator-in-the-loop adaptation can be split into two main activities. First, the introduction of information on external events, and second, the selection of adaptation plans from the design-space exploration. External events, such as sales campaigns, are determined by external parties and cannot be modeled into the architecture model at design-time. Therefore, the operator must be able to introduce new cloud profiles, data handling policies, and workload models to compose and inject knowledge at run-time. As workload models are complex, operators want to base them on observed behavior. Therefore, the approach will provide the ability to extract and modify intensities and behavior pattern.

The design-space exploration might return multiple adaptation candidates all representing a different element of a Pareto optimum. Human operators must assess and select adaptation plans which is supported by views and analyses. Therefore, we investigate the creation of views on the system that provide operators with up to date information about the actual cloud application derived from the architectural run-time models of iObserve [5], the different adaptation candidates, and the associated adaptation tasks.

V. Conclusion

We proposed extensions to the iObserve monitoring and analysis approach for combining automatic and operator-in-the-loop adaptation founded on architectural run-time models, workload characterization, privacy-driven planning, and design space exploration. Extensions are necessary to address challenges when including planning and execution activities.

In the future we will refine the proposed approach and extent the existing iObserve tooling for monitoring and analysis of cloud-based software systems by planning and execution capabilities. We will elaborate and evaluate the approach utilizing the community case studies CoCoME [39] and SPECjEnterprise[1]. We plan to perform simple user experiments and more complex controlled experiments. In the latter humans will have to execute specific prepared scenarios shaped around a set of external events which partly will occur without preparation for the operator. As base line, we will also perform an automatic selection and application of adaptation candidates.

[1]SPECjEnterprise is trademark of the Standard Performance Evaluation Corporation (SPEC), http://www.spec.org/jEnterprise2010/

REFERENCES

[1] E. Di Nitto, C. Ghezzi, A. Metzger, M. Papazoglou, and K. Pohl, "A journey to highly dynamic, self-adaptive service-based applications," *Automated Software Engineering*, 2008.

[2] A. Metzger (Ed.), "Software engineering: Key enabler for innovation," NESSI White Paper, July, 2014.

[3] B. H. Cheng, R. de Lemos, H. Giese, P. Inverardi, J. Magee, J. Andersson, B. Becker, N. Bencomo, Y. Brun, B. Cukic, G. Di Marzo Serugendo, S. Dustdar, A. Finkelstein, C. Gacek, K. Geihs, V. Grassi, G. Karsai, H. M. Kienle, J. Kramer, M. Litoiu, S. Malek, R. Mirandola, H. A. Müller, S. Park, M. Shaw, M. Tichy, M. Tivoli, D. Weyns, and J. Whittle, "Software engineering for self-adaptive systems: A research roadmap," in *Software Engineering for Self-Adaptive Systems*. Springer, 2009, pp. 1–26.

[4] B. Morin, O. Barais, J.-M. Jezequel, F. Fleurey, and A. Solberg, "Models@run.time to support dynamic adaptation," *IEEE Computer*, vol. 42, no. 10, pp. 44–51, 2009.

[5] R. Heinrich, E. Schmieders, R. Jung, W. Hasselbring, A. Metzger, K. Pohl, and R. Reussner, "Run-time architecture models for dynamic adaptation and evolution of cloud applications," Kiel University, Kiel, Germany, Technical Report 1503, April 2015. [Online]. Available: http://eprints.uni-kiel.de/28566/

[6] E. Schmieders, A. Metzger, and K. Pohl, "Runtime model-based privacy checks of big data cloud services," in *13th International Conference on Service Oriented Computing*. Springer, 2015.

[7] W. Hasselbring, R. Heinrich, R. Jung, A. Metzger, K. Pohl, R. Reussner, and E. Schmieders, "iObserve: integrated observation and modeling techniques to support adaptation and evolution of software systems," Kiel University, Kiel, Germany, Technical Report 1309, Oktober 2013. [Online]. Available: http://eprints.uni-kiel.de/22077/

[8] R. Jung, R. Heinrich, and E. Schmieders, "Model-driven instrumentation with Kieker and Palladio to forecast dynamic applications," in *Symposium on Software Performance*. CEUR Vol-1083, 2013, pp. 99–108.

[9] R. Heinrich, E. Schmieders, R. Jung, K. Rostami, A. Metzger, W. Hasselbring, R. Reussner, and K. Pohl, "Integrating run-time observations and design component models for cloud system analysis," in *9th International Workshop on Models@run.time*. CEUR Vol-1270, 2014, pp. 41–46.

[10] R. Jung, R. Heinrich, E. Schmieders, M. Strittmatter, and W. Hasselbring, "A method for aspect-oriented meta-model evolution," in *2nd Workshop on View-Based, Aspect-Oriented and Orthographic Software Modelling*. ACM, 2014, pp. 19–22.

[11] U. e Ghazia, R. Masood, and M. Shibli, "Comparative Analysis of Access Control Systems on Cloud," in *13th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel Distributed Computing*, 2012, pp. 41–46.

[12] M. Gondree and Z. N. Peterson, "Geolocation of data in the cloud," in *3rd ACM conference on Data and application security and privacy*. ACM, 2013, pp. 25–36.

[13] E. Schmieders, A. Metzger, and K. Pohl, "A runtime model approach for data geo-location checks of cloud services," in *12th International Conference on Service Oriented Computing*. Springer, 2014, pp. 306–320.

[14] ——, "Architectural runtime models for privacy checks of cloud applications," in *7th International Workshop on Principles of Engineering Service-Oriented and Cloud Systems*. ACM, 2015.

[15] N. Huber, A. van Hoorn, A. Koziolek, F. Brosig, and S. Kounev, "S/T/A: Meta-modeling Run-time Adaptation in Component-Based System Architectures," in *IEEE 9th International Conference on e-Business Engineering*. IEEE, 2012, pp. 70–77.

[16] A. van Hoorn, M. Rohr, I. A. Gul, and W. Hasselbring, "An adaptation framework enabling resource-efficient operation of software systems," in *Warm Up Workshop (WUP 2009) for ACM/IEEE ICSE 2010*. ACM, Apr. 2009, pp. 37–40.

[17] J. von Kistowski, N. R. Herbst, D. Zoller, S. Kounev, and A. Hotho, "Modeling and Extracting Load Intensity Profiles," in *10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, 2015.

[18] D. A. Menasce and V. Almeida, *Capacity Planning for Web Services: Metrics, Models, and Methods*. Prentice Hall PTR, 2001.

[19] A. van Hoorn, C. Vögele, E. Schulz, W. Hasselbring, and H. Krcmar, "Automatic extraction of probabilistic workload specifications for load testing session-based application systems," in *8th International Conference on Performance Evaluation Methodologies and Tools (ValueTools 2014)*. ICST, 2014, pp. 139–146.

[20] R. Heinrich, *Aligning Business Processes and Information Systems: New Approaches to Continuous Quality Engineering*. Springer, 2014.

[21] C. Ghezzi, M. Pezzè, M. Sama, and G. Tamburrelli, "Mining behavior models from user-intensive web applications," in *36th International Conference on Software Engineering*, 2014.

[22] S. Frey and W. Hasselbring, "Model-based migration of legacy software systems to scalable and resource-efficient cloud-based applications: The CloudMIG approach," in *1st International Conference on Cloud Computing, GRIDs, and Virtualization (Cloud Computing 2010)*, Lisbon, Portugal, 2010, pp. 155–158.

[23] ——, "The CloudMIG approach: Model-based migration of software systems to cloud-optimized applications," *International Journal on Advances in Software*, vol. 4, no. 3 and 4, pp. 342–353, Apr. 2011.

[24] A. Aleti, B. Buhnova, L. Grunske, A. Koziolek, and I. Meedeniya, "Software architecture optimization methods: A systematic literature review," *Software Engineering, IEEE Transactions on*, vol. 39, no. 5, pp. 658–683, 2013.

[25] S. Frey, F. Fittkau, and W. Hasselbring, "Search-based genetic optimization for deployment and reconfiguration of software in the cloud," in *35th International Conference on Software Engineering (ICSE 2013)*. IEEE Press, May 2013, pp. 512–521.

[26] S. A. Boyer, *Scada: Supervisory Control And Data Acquisition*. International Society of Automation, 2009.

[27] M.P. Robillard, "Recommendation systems for software engineering," *Software, IEEE*, vol. 27, no. 4, pp. 80–86, 2010.

[28] F. McCarey, M. Cinnide, and N. Kushmerick, "Rascal: A recommender agent for agile reuse," *Artificial Intelligence Review*, vol. 24, no. 3-4, pp. 253–276, 2005.

[29] C. Zhang, J. Yang, Y. Zhang, J. Fan, X. Zhang, J. Zhao, and P. Ou, "Automatic parameter recommendation for practical api usage," in *34th International Conference on Software Engineering*. IEEE, 2012, pp. 826–836.

[30] D. Cubranic, G. Murphy, J. Singer, and K. Booth, "Hipikat: a project memory for software development," *Software Engineering, IEEE Transactions on*, vol. 31, no. 6, pp. 446–465, 2005.

[31] S. Lee, S. Kang, S. Kim, and M. Staats, "The impact of view histories on edit recommendations," *Software Engineering, IEEE Transactions on*, vol. 41, no. 3, pp. 314–330, 2015.

[32] C. Chambers and C. Scaffidi, "Impact and utility of smell-driven performance tuning for end-user programmers," *Journal of Visual Languages & Computing*, vol. 28, no. 0, pp. 176 – 194, 2015.

[33] G. Jung, T. Mukherjee, S. Kunde, H. Kim, N. Sharma, and F. Goetz, "Cloudadvisor: A recommendation-as-a-service platform for cloud configuration and pricing," in *IEEE 9th World Congress on Services*, 2013, pp. 456–463.

[34] F. Fittkau, J. Waller, C. Wulf, and W. Hasselbring, "Live trace visualization for comprehending large software landscapes: The ExplorViz approach," in *1st IEEE International Working Conference on Software Visualization (VISSOFT 2013)*, Sep. 2013.

[35] F. Fittkau, S. Roth, and W. Hasselbring, "ExplorViz: Visual runtime behavior analysis of enterprise application landscapes," in *23rd European Conference on Information Systems (ECIS 2015 Completed Research Papers)*. AIS Electronic Library, May 2015.

[36] K. Rostami, J. Stammel, R. Heinrich, and R. Reussner, "Architecture-based assessment and planning of change requests," in *11th International ACM SIGSOFT Conference on Quality of Software Architectures*. ACM, 2015, pp. 21–30.

[37] R. Heinrich, K. Rostami, J. Stammel, T. Knapp, and R. Reussner, "Architecture-based analysis of changes in information system evolution," *17th Workshop Software-Reengineering & Evolution, Softwaretechnik-Trends*, vol. 35, 2015.

[38] C. Heger and R. Heinrich, "Deriving work plans for solving performance and scalability problems," in *Computer Performance Engineering*, ser. LNCS Vol. 8721. Springer, 2014, pp. 104–118.

[39] R. Heinrich, S. Gärtner, T.-M. Hesse, T. Ruhroth, R. Reussner, K. Schneider, B. Paech, and J. Jürjens, "A platform for empirical research on information system evolution," in *27th International Conference on Software Engineering and Knowledge Engineering*. KSI Research Inc., 2015, pp. 415–420.