The Making of Cloud Applications: An Empirical Study on Software Development for the Cloud

Jürgen Cito, Philipp Leitner, Thomas Fritz, Harald C. Gall
University of Zurich, Switzerland
{lastname}@ifi.uzh.ch

ABSTRACT

Cloud computing is gaining more and more traction as a deployment and provisioning model for software. While a large body of research already covers how to optimally operate a cloud system, we still lack insights into how professional software engineers actually use clouds, and how the cloud impacts development practices. This paper reports on the first systematic study on how software developers build applications for the cloud. We conducted a mixed-method study, consisting of qualitative interviews of 25 professional developers and a quantitative survey with 294 responses. Our results show that adopting the cloud has a profound impact throughout the software development process, as well as on how developers utilize tools and data in their daily work. Among other things, we found that (1) developers need better means to anticipate runtime problems and rigorously define metrics for improved fault localization and (2) the cloud offers an abundance of operational data, however, developers still often rely on their experience and intuition rather than utilizing metrics. From our findings, we extracted a set of guidelines for cloud development and identified challenges for researchers and tool vendors.

Categories and Subject Descriptors

D.2.0 [**Software Engineering**]: General—*Standards*

General Terms

Cloud Software Engineering, Human Factors

Keywords

cloud computing, user study

1. INTRODUCTION

Since its emergence, the cloud has been a rapidly growing area of interest [1, 2]. Several cloud platforms, such as Amazon's EC2, Microsoft Azure, Google's App Engine, or IBM's Bluemix, are already gaining mainstream adoption. Developing applications on top of cloud services is becoming common practice. Due to the cloud's flexible provisioning of resources, and the ease of offering services online for anyone, the cloud also influences software development practices. For instance, cloud development is often associated with the concept of "DevOps", which promotes the convergence of the development and operation of applications [3].

There is currently significant research interest in how to efficiently manage cloud infrastructures, for instance in terms of energy efficiency [4] or maximized server utilization [5]. Another core area of interest in cloud computing research is its use for high-performance computing in lieu of an expensive computer grid [6]. However, so far, there is little systematic research on the consumer side of cloud computing, i.e., how software developers actually develop applications in and for the cloud. Only recently, Barker et al. voiced this concern in a position paper, stating that the academic community ought to conduct more "user-driven research" [7].

In this vein, this paper presents a systematic study on how professional software engineers develop applications on top of cloud infrastructures or platforms. We deliberately cover a broad scope, and analyze how applications are designed, built and deployed, as well as what technical tools are used for cloud development. We conducted a mixed-method study consisting of an initial interview study with 16 professional cloud developers, a quantitative survey with 294 respondents, and a second round of interviews with 9 additional professionals to dive deeper into some questions raised by the survey. All interview participants work at international companies of widely varying size (from small start-ups to large enterprises), and have diverse backgrounds with professional experience ranging from 3 to 23 years.

In particular, we address the following research questions:

***RQ 1**: How does the development and operation of applications change in a cloud environment?*

***RQ 2**: What kind of tools and data do developers utilize for building cloud software?*

Our research has important implications for cloud developers, researchers, and vendors of cloud-related tooling. Primarily, due to the volatility of cloud instances, developers need to accustom themselves to not being able to directly touch the running application any longer. That is, quick fixes of production configurations are equally impossible as logging into a server for debugging. As a research community, we need to investigate how to best support developers in this task, as well as analyze how code artefacts related to cloud instance management evolve. Finally, we have seen that more types of metrics get more and more

ESEC/FSE'15, August 30 – September 4, 2015, Bergamo, Italy
ACM. 978-1-4503-3675-8/15/08...$15.00
http://dx.doi.org/10.1145/2786805.2786826

important, but they are still not directly actionable for developers. Hence, we need to research better tooling that brings this data into the daily workflow of cloud developers.

The remainder of the paper is structured as follows. First, we provide some background on cloud computing terminology (Section 2), followed by a discussion of related work in Section 3. We present the study design in Section 4, followed by an in-depth summary of our findings (Section 5) and a discussion of the implications resulting from these findings (Section 6). We detail the major threats to validity of our research in Section 7, and conclude the paper in Section 8.

2. BACKGROUND

While the term "cloud computing" is commonly ill-defined[1], the research community has widely gravitated towards the NIST definition [8]. As illustrated in Figure 1, this definition considers three levels, each defined by the responsibilities of IT operations provided by the cloud vendor. In an *Infrastructure-as-a-Service (IaaS)* cloud, resources (e.g, computing power, storage, networking) are acquired and released dynamically, typically in the form of virtual machines. IaaS customers do not need to operate physical servers, but are still required to administer their virtual servers, and manage installed software. *Platform-as-a-Service (PaaS)* clouds represent a higher level of abstraction and provide entire application runtimes as a service. The PaaS provider manages the hosting environment and customers only submit their application bundles. They typically do not have access to the physical or virtual servers that the applications are running on. Finally, in *Software-as-a-Service (SaaS)*, complete applications are provided as cloud services to end customers. The provider handles the entire stack, including the application. The client is only a user of the service.

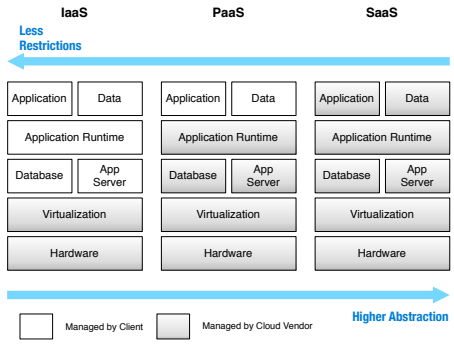

Figure 1: Basic models of cloud computing (following [8])

PaaS clouds are particularly interesting for software engineers, as they allow them to solely focus on developing applications. They typically relieve the developer from having to care about any operations tasks, and handle varying system load transparently via auto-scaling. This ability to adapt to workload changes is referred to as *elasticity*. However, in order to do so, these platforms impose severe restrictions. For instance, they typically only support rather narrowly defined application models (e.g., three-tier Web applications), and require the developer to program against provided APIs. This often also leads to vendor lock-in [9].

[1]Oracle's CEO Larry Ellison once noted jokingly that he cannot think of a single thing Oracle does that is not "cloud".

With IaaS, the idea of Infrastructure-as-Code (IaC) has also started to gain momentum. IaC allows users to define and provision operation environments in version-controlled source code. Essentially, in an IaC project, the entire runtime environment of the application (e.g., IaaS resources, required software packages, configuration) is defined using scripts, which can then be executed by tools such as Opscode Chef. These scripts allow entire test, staging or production environments to be started without manual interaction. The move towards IaC with its reproducible provisioning has become necessary since cloud applications often consist of a large number of machines that have to be configured automatically to scale horizontally.

Another concept commonly associated with cloud development is DevOps [3]. DevOps describes the convergence of the previously mostly separated tasks of developing an application, and its deployment and operation. In DevOps, software development and operation activities are often handled by the same team, or even by the same engineer. By aligning the goals of development and operations, DevOps aims at improving agility and cooperation.

3. RELATED WORK

There has been a multitude of empirical research on the development of general software applications. For instance, Singer et al. have recently researched how developers use Twitter [10]. Murphy-Hill et al. have looked at how bugs are fixed [11]. However, so far, very little empirical research has been conducted in the cloud computing domain, even though there are several calls for more research on software development for the cloud. Barker et al. [7] recently named "user-driven research" as one of the major opportunities for high-impact cloud research. Khajeh-Hosseini et al. [12] stated that the organizational and process-oriented changes implied by adopting the cloud is currently not sufficiently researched. While Mei et al. did not consider software engineering a major challenge for cloud computing in 2008 [13], they later on provided a whole list of software engineering issues to be tackled by research [14].

So far, research in cloud computing has mainly focused on provider-side issues (e.g., relating to server management [4, 5] or performance measurement[6]). Similarly, work by Bezemer et al. mainly focuses on performance problems on the server-side of SaaS applications, rather than development aspects of cloud software in general[15]. On the client side, some research has been conducted on concrete programming models. A large part of this research deals with data analysis, typically using the Map/Reduce paradigm (e.g., [16]). While interesting, these works do not cover the professional software development environment that we address with our study. Research on cloud programming models for non-scientific contexts is more limited. One example is the jCloudScale framework proposed in [17][18]. jCloudScale is a Java-based middleware that aims to simplify the development of IaaS applications. A similar goal also motivated the research presented in [19], which investigated an extension of Java RMI for simplifying the development of elastic, cloud-based applications.

One aspect that is already reasonably well-understood in literature is how and when companies choose to adopt cloud computing, and for which reasons[20][21]. Both of these studies are concerned primarily with SaaS adoption. That is, they target cloud adoption by end users more than by

professional software developers. This is not the case in a related industry study, dubbed the "DevOps Report" [22]. This survey garnered over 9200 respondents, praising the DevOps idea as a key enabler of profitable and agile companies. Given that the source of this report is also a major player in the DevOps business, independent scientific evaluation to support these results would be valuable.

None of the work discussed so far has empirically evaluated how cloud software is actually developed in practice. The only work we are aware of that goes into this direction is a (not peer-reviewed) white paper on enterprise software development in the cloud [23]. This report is based on a survey with 408 respondents. The report concludes that enterprise developers are largely not yet adopting the cloud, but if they do, they are able to improve time-to-market.

4. RESEARCH METHOD

To investigate how the cloud influences software development practices, we conducted a study based on techniques found in Grounded Theory [24]. Following the recommendations in [25], we used a mixed methodology consisting of three steps of data collection and iterative phases of data analysis. First, we defined a set of open-ended questions from our research questions and conducted *qualitative interviews* with 16 participants. Second, to further substantiate our findings, we ran a *quantitative survey* and gathered responses from 294 professional software developers. Using open coding, we identified 4 topics of high interest. To gain a better understanding and more details on these topics, we then conducted a second round of *qualitative deep-dive interviews* with 9 professional developers. Table 1 provides a more detailed breakdown of our participants. All interview and survey materials can be found on our web site[2].

Qualitative Interview Study (Interview$_1$).

Protocol. We conducted semi-structured interviews with software developers that had previously already deployed software in the cloud in a professional context. For these interviews, we defined a set of 23 questions based on our research questions. In the interviews, we covered all questions with each participant, but the concrete order of questions followed the "flow" of the interview. Interviews were conducted by the first author, either face-to-face on-site of the interviewee or via Skype. Interviews lasted between 30 and 60 minutes, were conducted in either German or English, and were audio-recorded.

Participants. Interview participants were recruited from industry partners and our personal network. To cover a broad range of cloud development experiences, we recruited participants from both, smaller companies (1 – 100 employees) and larger enterprises (> 100 employees). Participants had to either deploy on IaaS or PaaS as well as in public and private clouds. Furthermore, we also made sure to recruit some participants that delivered SaaS applications. Overall, we recruited 16 participants (P1 to P16), all male software developers with 3 to 23 years of professional development experience (average of 9 years ± 7 standard deviation), and from 4 different countries and two continents. Our 16 participants came from 5 different companies—12 participants worked in larger enterprises, 4 in smaller companies.

[2]http://www.ifi.uzh.ch/seal/people/cito/cloud-developer-study.html

Analysis. After the interviews, we transcribed all audio recordings. The first two authors then used an open coding technique to iteratively code and categorize participants' statements, resulting in a set of findings. All findings are supported by statements of multiple participants.

Quantitative Study (Survey).

Protocol. In the second step of our study, we designed a survey with 32 questions, most of which map to our initial findings. The questions were primarily formulated as statements, asking participants to state their agreement on a five point Likert scale (examples of these questions can be seen in Figure 3). To target developers with experience in cloud technologies, we gathered profiles of GitHub[3] users that "follow" a repository of a number of popular cloud platforms, including Amazon Web Services, Heroku, Google Cloud Platform, CloudFoundry, and OpenStack. We then discarded all users without a public email address in their profile, and contacted the remaining users with a description and link to our online survey. To motivate developers to participate in the survey, we held a raffle for all participants to win one of two 50 USD Amazon gift vouchers. The survey was in English, and took an average of 12.2 minutes to complete.

Participants. We emailed the survey invitation to 2000 GitHub users and gathered a total of 294 responses (response rate of 14.7%). Of all 294 participants, 192 were employed in smaller companies (between 1 and 100 employees) and 102 were employed in large companies. 71% stated their job role as software developer, 22% as team lead or product owner, 4% as operations engineer, and the remaining 3% listed software architect, researcher or chief technology officer (CTO). The average professional development experience per participant was 9 years (± 5).

Analysis. We analyzed the response distributions and present the results along with the findings from the interview study phase. Furthermore, we examined the responses of three free-text questions on overall differences in development in cloud versus non-cloud environments, restrictions in the cloud, and tooling. Based on these results, we were able to enhance our understanding of some of our findings.

Qualitative Deep-Dive Interviews (Interview$_2$).

Protocol. Through the examination of open-ended survey questions, we identified 4 topics of high interest: *(1) fault localization, (2) monitoring and performance troubleshooting, (3) cost of operation,* and *(4) design for scalability.* In order to get more profound insights into these topics, we defined an overall set of 27 questions for these 4 topics and conducted another round of semi-structured interviews. We followed the same protocol as in the first round of qualitative interviews. Interviews lasted between 30 and 45 minutes and were audio-recorded.

Participants. Interviewees were recruited through our personal network. Overall, we recruited 9 participants (D1 to D9), 8 male and 1 female software developer with an average professional development experience of 8 years (± 6) from 6 different countries and two continents. All participants were from different companies. 3 participants deployed on PaaS, 5 on IaaS and one on IaaS but also on PaaS.

Analysis. After the interviews and based on the topics and categories identified in the previous two steps, we used

[3]http://www.github.com

Table 1: Method and Participants

Study Type	# Questions	Participants		Platform		Company		Experience
		#	IDs	PaaS	IaaS	enterprise	small	Avg (± StdDev)
Interviews	**50**	**25**		**15**	**10**	**14**	**11**	**9 years (± 6.5)**
Interview_1	23	16	P1 - P16	13	3	12	4	9 years (± 7)
Interview_2	27	9	D1 - D9	3	6	2	7	8 years (± 6)
Survey	**23**	**294**		**103**	**191**	**102**	**192**	**9 years (± 5)**

open coding to categorize interview statements and gather more profound insights into the difference between cloud and non-cloud development.

5. FINDINGS

In the following, we present the findings of our study. We first give a high-level overview of our findings in Section 5.1, and then provide more detailed results relating to **RQ1** (Section 5.2) and **RQ2** (Section 5.3).

5.1 Overview

For our study, we were primarily interested in examining what makes software development for the cloud "unique", i.e., what differs in terms of processes, tools, and implementation choices, to other development projects. However, early on in our interviews, it became clear that "the" cloud does not exist for practitioners. Hence, we asked our survey respondents to list the main characteristics that define cloud computing for them and gathered answers from 160 developers (multiple answers were allowed). We categorized and quantified the most common themes, and present them in Figure 2. Note that the last three entries in the chart—automation, ease of infrastructure maintenance, and elasticity—are all strongly related, as are the first entries in the chart—focus on product, faster time-to-market.

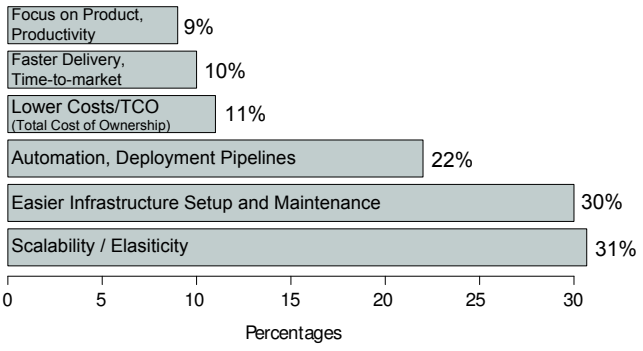

Figure 2: Main Differences in Cloud Development

The majority of developers thinks about the cloud mostly as a deployment and hosting technology, following either the IaaS or PaaS model. For these developers, the ability to easily scale applications and the ease of infrastructure maintenance is what makes the cloud unique. While productivity and faster time-to-market has been named as a distinguishing feature in our interviews, these were only relevant to about every tenth survey respondent. Finally, it is interesting to note that reduced Total Costs of Ownership (TCO) are also only a distinguishing factor for a minority of cloud developers.

Our interviews further revealed that there is a large mindset gap between developers who think of the cloud as either IaaS or PaaS, and those who think of it mostly in terms of SaaS. For developers where cloud is seen more as a delivery model (i.e., SaaS), how the application is actually hosted tends to be opaque in cloud and non-cloud environments alike and not much changed when adopting cloud computing.

This was, however, different for interview participants working on IaaS and PaaS services. These participants consistently commented on a range of differences. The analysis of our interview transcripts and the survey data shows that the changes when adopting cloud computing for IaaS and PaaS developers fall into the following broad categories: changes to how applications are provisioned and deployed (**Deployment & Automation**), changes in how applications are actually built (**Design & Implementation**), changes in how problems can be debugged (**Troubleshooting & Maintenance**), and cultural changes (**DevOps Communication**). Other areas that are part of software engineering, such as requirements engineering or security, are not addressed in this study as they did not emerge from our study data.

We were also interested in investigating the role of data and operational metrics, as it plays an important role in software development[26]. Indeed, the analysis of our study showed that it plays a major role in cloud development.

Concretely, the usage of cloud-based tooling has increased (**Cloud Tooling**), more and more types of data are available for teams (**Monitoring & Production Data**), but that developers currently struggle to fully utilize this additional data (**Usage of Runtime Data**). In the following, we will discuss our detailed findings based on these broad groups of changes.

In Figure 3, we summarize the quantitative results for the Likert-type scale questions from our survey that relate to our core research questions. We grouped the results in Figure 3 according to our subtopics. The insights from *Deployment & Automation* and *Troubleshooting & Maintenance* resulted primarily from our interviews, rather than the survey.

In general, we saw a high level of agreement with our interview findings. However, a few individual questions showed some disagreement as well, requiring more detailed study. These aspects are discussed in more detail in the remainder of the paper. In the following we present the main themes of our study. Where applicable, we provide quantitative results from our survey in the presentation of our findings.

5.2 Application Development and Operation

In this section, we report on how cloud computing has affected application development and operations, as well as on the main drivers for these changes, API-driven infrastructure-at-scale and cloud instance volatility.

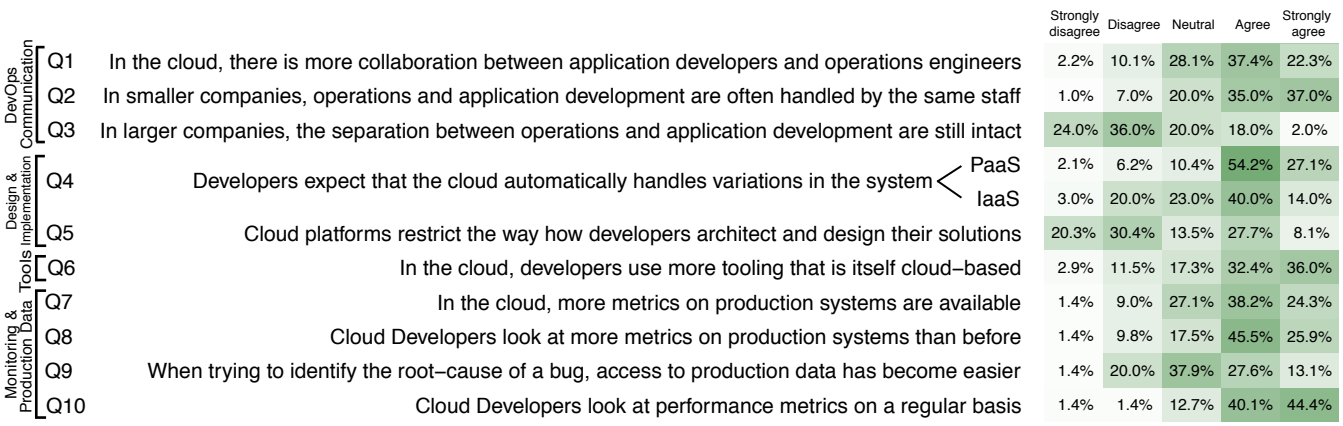

Figure 3: Results from our quantitative survey

5.2.1 Deployment & Automation

Our interviews have shown that elasticity, ease of infrastructure maintenance, and automation can be broken down into two fundamental aspects that drive most changes of software development in the cloud: (1) *API-driven infrastructure at scale* and (2) *volatility of cloud instances*. Both these aspects have ripple effects on almost every aspect of cloud development.

API-driven Infrastructure-at-Scale. Infrastructure-at-scale refers to the ability to quickly spawn up (and discard) many compute instances using an API. This ability requires more automation on many levels, including infrastructure, environment and test. In *IaaS clouds*, automation happens by defining your infrastructure as a set of software artifacts (see IaC in Section 2). This allows for automatic provisioning of newly created instances for different purposes (e.g., scaling up, setting up a test environment). In PaaS clouds, applications are required to be packaged in a way to be easily reproducible (e.g., containers, buildpacks) to manage this automation behind the scenes. All interviewees deploying on IaaS agreed that the use of IaC tools (e.g., Chef, Puppet) or other means of automation (e.g, shell scripts) for all automated provisioning of their infrastructure has become essential in the cloud.

Volatility of Cloud Instances. Cloud instances can be started up and shut down for various reasons. This volatility happens either through (1) the cloud provider shutting down instances, (2) the load balancer spawning or shutting down instances, and (3) the application itself shutting down a dynamically allocated instance that finished its work. One implication of instance volatility is that all infrastructure definition and server configuration has to be implemented in code. If provisioning and configuration is not automated, it is bound to be lost when instances are discarded. This means that every change in infrastructure results in a new deployment of the system. Four of the interviewees that deploy in IaaS referred to this practice as *immutable infrastructure*:

> "*We have now moved to strict Immutable Infrastructure in our deployment. We don't even put SSH keys into instances anymore, making changes to existing infrastructure impossible*" -P12$_{IaaS}$

In PaaS, the infrastructure is managed by the cloud provider. Hence, the infrastructure is immutable for developers by default.

Infrastructure Transparency. All interviewees deploying on IaaS mentioned the use of either IaC tools or shell scripts for all automated provisioning of their infrastructure. They argue that this brings them transparency regarding their infrastructure:

> "*What happens in our infrastructure is a lot more obvious. Everything we do on that level [infrastructure] is over code (...) So, I don't need ask my colleague what he did to get that process running - I just look at the code and maybe the commit history*" -P9$_{IaaS}$

Virtual Containers for Automation. Furthermore, three interviewees describe a push in virtualization from virtual machines towards virtual containers (e.g., LXC[4], Docker[5]) for their automation:

> "*Virtual machines are too slow in a large scale (...) Speed matters, also in integration testing. When I can make a build take 3 minutes instead of 20, that's a huge win*" -P13$_{IaaS}$

Traditional virtual machines impose a large performance overhead due to the additional virtualization layer. Containers allow for much faster start-up times and are, therefore, also increasingly used as a base for PaaS [27].

> Infrastructure provisioning and application deployment in the cloud are largely automated. Servers are not seen as durable entities. Hence, any changes to infrastructure need to be defined in code. This also leads to more transparency concerning infrastructure changes.

5.2.2 Design & Implementation

In this section, we report on how restrictions in the cloud influence how developers design and implement their applications.

Design Restrictions. As part of our survey, we asked whether the survey respondents face limitations in application architecture and design specific to the cloud. Results

[4] `https://linuxcontainers.org/`
[5] `https://www.docker.com/`

to this question were inconclusive, with 51% of respondents disagreeing and 36% agreeing (see Q5 in Figure 3). However, 119 respondents have still (in a free-form field in the survey) stated multiple restrictions, which we categorized and quantified in Figure 4.

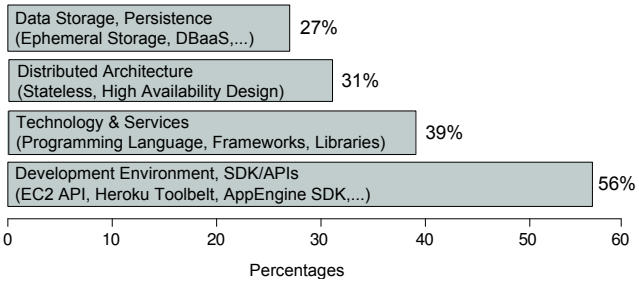

Figure 4: Most significant restrictions developers have encountered on cloud platforms

It is not surprising that there are technical restrictions regarding the supported SDKs, libraries, and frameworks. However, to our interview participants, these restrictions were not all negative. Some developers feel that technological restrictions allow them to focus more on delivering business value, instead of tinkering with low-level technology choices:

> "*But I don't wanna make those decisions [on technology] anyway. (...) The cool thing - from a product design point of view - is, that you know what works on AppEngine and what doesn't*" -P11$_{PaaS}$

More interesting is that close to a third of all respondents also consider the cloud to restrict them in the way they actually architect and design their applications. These restrictions in design and architecture are primarily caused by API-driven infrastructure-at-scale and volatility of cloud instances, as discussed in Section 5.2.1.

Design for Failure. Volatility of cloud instances naturally forces developers to build highly fault-tolerant applications, as IaaS providers reserve the right to shut down any resources at any time, on short or without any prior notice:

> "*One interesting thing that is very cloud specific and influenced our architecture is, that the cloud provider tells you, we can kill your machine any time we want.*" -P9$_{IaaS}$

Well-known cloud users have already adopted this mindset. Netflix, for instance, has stated that they use an application called Chaos Monkey[6] to randomly terminate cloud instances in production, to force application design that can tolerate such failures when they happen unintentionally.

Design for Scalability. Scalability is the most named difference in cloud development for survey participants (see Figure 2). In our deep-dive interviews we asked participants to explain how scalability considerations influenced them during design and implementation.

All interviewees stated that they always have scalability in mind, even when designing very simple cloud applications:

> "*Even if the customer needs only 1 server, we have an ELB (Elastic Load Balancer) anyway, because we expect everything to grow*" -D3$_{IaaS}$

The Twelve-Factor app[7] design has become the de-facto standard when it comes to best practices when building cloud applications. A few of the interview and survey participants referred particularly to this manifesto, while others often referenced similar practices for implementing scalability under a different name.

An alternative approach to implement fault-tolerant and scalable cloud applications that was mentioned are microservices [28]:

> "*We have divided our application into many services. For one specific service we kill off and start new instances all the time, also to have proper redundancy.*" -P9$_{IaaS}$

However, in our interviews, only five participants mentioned either currently using or having plans to move to a more service-oriented system design in the near future.

> The cloud imposes some restrictions on how applications can be built, in technological and software architectural terms. Specifically, applications need to be designed for scalability and fault tolerance. These restrictions are also seen as positive as they enforce the best practices and foster business value.

5.2.3 Troubleshooting & Maintenance

Activities that happen after the application has been deployed (i.e., troubleshooting and maintenance) have probably seen the biggest change in cloud development. Servers are not seen as durable entities anymore. Therefore, infrastructure maintenance has become an activity that now has to deal with adapting infrastructure code files rather than tweaking on live server instances.

Fault Localization. Fault localization, the activity of discovering the exact locations of program faults, cannot be done through ad-hoc inspection on, for instance, log files, memory dumps, or system metrics on live production servers anymore.

This means that every information that is of interest in the maintenance phase of the development life cycle must be specified before deployment and collected in a central repository, otherwise information runs the risk of being lost due to cloud instance volatility. These techniques are not necessarily bound to the cloud. However, in the cloud these best practices are seen as mandatory:

> "*In the cloud you are forced to use best practices, you are forced to use automation. You are forced on not relying to having root access to jump onto a machine and search logs by hand. You are forced to use better practices like aggregation.*" -D6$_{PaaS}$

Reproducing Issues. In terms of reproducibility of issues in a local environment for fault localization, our interviewees were somewhat divided. On the one hand, this task has become more difficult, as cloud applications are inherently distributed and reproducing a distributed environment locally is generally a difficult task. On the other

[6] `http://techblog.netflix.com/2012/07/chaos-monkey-released-into-wild.html`

[7] `http://12factor.net/`

hand, deployment automation makes it easier and faster to spin up a staging or testing environment in the cloud to reproduce issues. However, our interview partners also mentioned the additional cost involved of spinning up environments as a reason for first attempting to reproduce issues locally. Some interviewees also stated that proper end-to-end monitoring[29, 30] and request tracing needs to be in place to be able to reproduce issues correctly:

> "*If you missed some logs and the instance is already gone, have fun reproducing your environment*" -D8$_{PaaS}$

Especially with public cloud providers, hardware characteristics also need to be tracked, as you never know what specific hardware configuration will be served[31].

> Troubleshooting has changed, as problematic cloud instances are often not accessible or already discarded, rendering hot fixes or searching for logs in production impossible. Instead, relevant logs must be defined beforehand and collected in a central repository. These best practices are not exclusive to the cloud, but instance volatility makes their usage in cloud computing mandatory.

5.2.4 DevOps Communication

As already discussed in Section 2, it is often argued that cloud computing goes hand in hand with a DevOps style of communication, which leads to higher collaboration between software developers and operations engineers. In our interviews, this notion was not undisputed. While 12 of the interviewees have agreed with the DevOps vision, the remaining participants (mostly from enterprise companies) have argued that there are still dedicated development and operations teams that more or less work as silos. Even companies that self-identified as following a DevOps approach still seem to generally have a separation between engineers that solely implement functional features, and engineers that mostly develop infrastructure code:

> "*We have our server/DevOps guy. (...) He handles the whole monitoring and tools thing*" -P9$_{IaaS}$

In our survey, there was a general agreement with these observations (see Figure 5). The survey responses show that, especially in large companies, communication and interaction has increased between application developers and operations engineers (72%). Contrary to our interview study, the difference between the responses of developers working in smaller or larger companies in terms of collaboration between development and operations is not large. For smaller companies, 70-74% tend to agree that their operations and development are now handled by the same staff, versus 57-60% in larger companies. This data suggests that, even in larger companies, the gap between development and operations activities converges.

> Close to 60% of developers across companies of all sizes build applications following the DevOps notion of converging development and operations teams. However, even in a DevOps team, there are still dedicated engineers that are responsible for maintaining the infrastructure code.

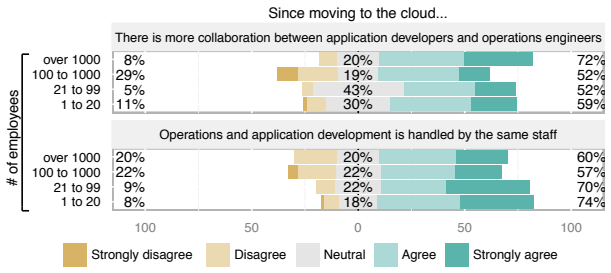

Figure 5: Results regarding Team Communication grouped by company size

5.3 Changes in Tools and Metrics Usage

In our research, we were particularly interested in how the usage of tools and production data has changed in cloud development projects. We report on (1) how tooling has evolved in the cloud, (2) what kind of metrics are available now, and (3) how these metrics are utilized.

5.3.1 Cloud Tooling

Tooling for the Cloud. In the survey, we asked what tools developers specifically use in development for the cloud, which they have not used before. 124 people responded to the question with multiple tools, which we categorized and quantified in Figure 6. We also asked whether cloud-based tool usage has generally increased. 67% of respondents agreed with this statement, while 15% disagreed (see Q6 in Figure 3).

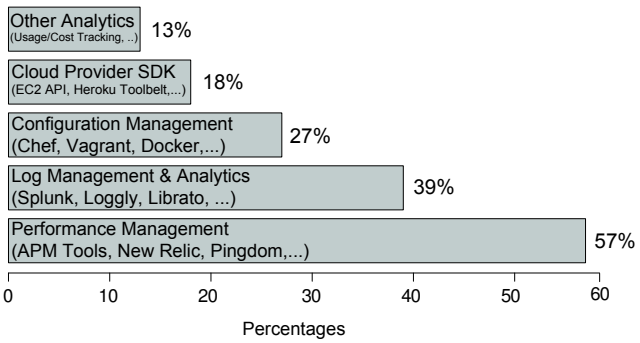

Figure 6: Tools used specifically for cloud development

Performance and Log management top the list of tools survey respondents use specifically for cloud development. This can again be attributed to the notions of API-driven infrastructure-at-scale and cloud instance volatility, which we have already discussed extensively in Section 5.2.

Tooling in the Cloud. We also observe an increase in the usage of tooling that is itself cloud-based. However, we cannot differentiate whether this rise has to do with development on cloud platforms, or with the fact that more and more tooling is moving to a SaaS model in general. Interviewees agreed that, when building cloud applications, many of the tools previously installed in the local infrastructure or on the developer machine are now also in the cloud. This includes tools for monitoring, analytics, and configuration management. Four of the interviewees even used an entirely Web-based IDE for most of their development tasks. All others moved at least part of their supporting tooling into the cloud. The interviewees developing entirely in the cloud

(i.e., through a cloud-based IDE) expressed the overall relief of not having to maintain or configure a local development environment:

> "*I don't need to take care of backups, or updates of the IDE. I can use the same setup everywhere. (...) Also there's no hassle about that missing plugin and that wrong configuration.*" -P1$_{PaaS}$

However, cloud-based IDEs were not mentioned by survey respondents at all as part of their tooling. Hence, we conclude that cloud-based IDEs have, unlike indicated by our interview study, not yet found mainstream adoption.

> Cloud-based tool usage has generally increased. Performance, log management and analytics tools are most used specifically for cloud development. Many of these tools are themselves cloud-based. However, despite some advantages, cloud-based IDEs have not yet found mainstream usage.

5.3.2 Monitoring & Production Data

Metric Awareness. All interview participants have mentioned using solutions for log aggregation and central operational metric collection for cloud development. The survey supports these claims with performance metric and log management tools being deemed the most important. What we have also observed is that the enforcement of these practices has led to more awareness for metrics:

> "*Software Developers have not - until recently - seen metrics as very important. They saw that stuff as an operations thing*" -D5$_{PaaS}$

In our survey, 72% of cloud developers agree that they look at more metrics on production systems than before (see Q8 in Figure 3).

Metric Availability. In general, interviewees expressed that they now have a much richer set of monitoring metrics on production systems available:

> "*Especially now in the cloud (with Heroku or Amazon) they provide so much data. Putting this data in a graphing tool and looking at it once in a while has become increasingly easier.*" -P12$_{IaaS}$

From the survey, we see that 62% of respondents agree that they actually have more operational metrics available than before (see Q7 in Figure 3). When asked whether access to production data has become easier, survey participants were divided: 21% disagreed, 38% were neutral and 41% agreed with the statement (see Q9 in Figure 3).

We investigated this claim further and found that metric utilization has increased both in quantity and dimension. In addition to technical system-level metrics (e.g., CPU utilization, cache hits), more teams now look also at business metrics (e.g., customer retention, number of logins) to guide their decisions. However, system performance metrics are deemed to be the most interesting by our survey participants. 84% state that they look at performance metrics on a regular basis (see Q10 in Figure 3).

> Cloud developers look at more production metrics. In addition to system-level metrics, business metrics are becoming more relevant. However, application performance is most often still measured via system-level metrics.

5.3.3 Usage of Runtime Data

An increase of tooling to acquire data, more metric awareness and availability spiked our interest in how developers use metrics in their work. Through analysis of our initial interviews and the open questions in our survey, we identified that *performance* and *cost* were the metrics of high interest to cloud developers. In the following we describe how our interview participants utilize this data in their regular development work.

Performance Troubleshooting. In our deep-dive interviews, we investigated how our study participants approach a particular performance problem by utilizing data from production systems to solve the problem.

A common theme is that developers follow a more reactive approach to metrics, i.e., they act on alerts or reports on the issue tracker provided by someone who is concerned with running the application in production (i.e., DevOps engineers, operations engineers, or system administrators). Metrics are typically provided in dashboards, accessible to everyone in the team, but mostly used by operations. However, when actively debugging a known performance issue, all interview participants would first go *"by intuition"* and reproduce the issue in their own development environment, before inspecting how metrics have evolved in production in the provided dashboards. Only if the local inspection does not yield any results, they would dig deeper into production data. Interviewees stated that they choose to ignore the data at first, because it is rather cumbersome to navigate performance dashboards, while at the same time navigating through code:

> "*I try to reproduce and solve the issue locally. Looking for the particular issue in the dashboard and jumping back and forth between the code is rather tedious*" -D2$_{IaaS}$

Costs of Deployment. To most of our interviewees, the costs of deployment in the cloud were generally deemed as important. However, developers seem to only in the abstract be aware that their design and implementation decisions have an influence on deployment costs. When confronted on how the costs of deployment (especially of specific code changes) are used in their daily work, it became clear that costs are not a tangible factor for developers. Some interviewees argued that having this information more wide-spread and accessible would be interesting to them, but does not fall into their responsibilities:

> "*I have no idea about the costs. I can just read it in the logs sometimes that in production we spawned 20-30 servers. It would be interesting to know, but it's not really important for application development*" -P5$_{PaaS}$

Not developers, but software architects or the CTO are concerned with the overall costs of operation. However, even for these roles, costs were considered in a post-design phase and currently do not influence their design decisions directly.

Currently, developers struggle to use the abundance of available runtime metrics. Rather, they often solve performance problems "by intuition" in a local environment. More detailed inspection of metrics is only used when this approach does not lead to a solution. Cloud costs are deemed as important, but are not tangible to developers.

6. DISCUSSION

We presented the results of the first systematic study on how professional software developers build applications on top of cloud infrastructures or platforms by addressing two research questions. We briefly revisit these research questions before discussing the ramifications of our results.

***RQ 1**: How does the development and operation of applications change in a cloud environment?*
In the cloud, servers are *volatile*. They are regularly terminated and re-created, often without direct influence of the cloud developer. Our study has shown that the concept of API-driven infrastructure-at-scale and the cloud instance volatility have ripple effects throughout the entire application development and operations process. They restrict the design of cloud applications and force developers to heavily rely on infrastructure automation, log management, and metrics centralization. While these concepts are also useful in non-cloud environments, they are mandatory for successful application development and operation in the cloud.

***RQ 2**: What kind of tools and data do developers utilize for building cloud software?*
Based on our research, more data, and more types of data, are utilized in the cloud, for instance business metrics (e.g., conversion rates) in addition to system-level data (e.g., CPU utilization). However, developers struggle to directly interpret and make use of this additional data, as current metrics are often not actionable for them. Similarly, cloud developers are in the abstract aware that their design and implementation decisions have monetary consequences, but in their daily work, they do not currently think much about the costs of operating their application in the cloud.

We now present, in a condensed form, the implications of our study results for practitioners, as well as the main challenges that cloud developers face. These form open problems that academic research and vendors of cloud-related tooling need to address to improve the experience of developers.

6.1 Implications for Practitioners

Our study has shown that there are a number of best-practices for building useful applications on top of IaaS and PaaS systems. These practices are not necessarily bound only to cloud development, but the nature of cloud infrastructures make these practices central for successfully deploying in a cloud. In the following we present a set of guidelines for software development in the cloud resulting from the findings of our study.

Cloud instances are *volatile*. Never assume that any instance will exist forever. In both, IaaS and PaaS cloud systems, backend instances come and go. Logs, configuration changes, or hot fixes stored only on a cloud instance are bound to be lost. As we have seen in Section 5.2.1, cloud developers should treat cloud instances as *immutable* black boxes, which they in general cannot fix, and in some cases cannot even log into. This requires a change in mindset for engineers used to having full control over their infrastructure.

***Anticipate runtime problems* and define relevant logs and operational metrics for fault localization before deployment. Use log management tools to centrally collect this data.** Related to the volatility of cloud instances, developers need to start thinking about how to localize and debug runtime faults already during development. This includes defining useful debugging statements, logs and operational metrics prior to deployment, as well as setting up and using tools to centrally *collect, persist, and analyse* these metrics outside of volatile instances. As discussed in Section 5.2.3, cloud instances are black boxes, drilling into unanticipated problems after deployment is often impossible. Hence, cloud developers should be aggressive in what they log and what operational metrics they trace. It is easier to filter out data that turns out to be unnecessary than to debug problems for which the relevant logs and metrics have not been collected.

Scalability and fault tolerance need to be *first-class citizens* in application design. While most distributed and Web-based applications have historically striven to be scalable and fault tolerant, at least to some degree, these concepts have become even more central in cloud projects. API-driven infrastructure-at-scale means that essentially any application or component can scale up dynamically (e.g., to react to increased load). Instance volatility means that runtime faults are bound to happen. As such, scalability and fault tolerance need to be first-class citizens when designing applications (see Section 5.2.2). *Best-practices for cloud application design* (e.g., the 12-Factor App) take this into account and should not be compromised by cloud developers. PaaS systems often enforce such application design through restrictions (e.g., in terms of statelessness). Developers should not aim to circumvent, but rather embrace those restrictions.

For IaaS, *automate* server provisioning and configuration, for instance using IaC tools. API-driven infrastructure-at-scale requires that the process of instance provisioning and configuration is fully automated. Besides being able to scale up quickly, this also has the additional advantage that knowledge about how servers are configured is explicitly documented in scripts or IaC code, and versioned in the project's version control system. This allows developers to *revert*, for instance, erroneous changes in the configuration of a cloud instance just like they would revert a broken application code change. It also makes infrastructure configuration and evolution explicit for other developers and DevOps engineers leading to more *transparency*, as discussed in Section 5.2.1. While some cloud developers currently use scripting (e.g., `bash`) for this purpose, the usage of dedicated IaC tools has additional advantages. Most importantly, IaC allows for reuse of existing open source provisioning and configuration code, and supports unit testing well.

Embrace the *tools and data* the cloud provides you. As elaborated in Section 5.3, we have seen that cloud developers have access to more tools and data. However, we have also seen that many developers still rather go *"by intuition"* when debugging problems rather than analyzing provided operational metrics. We argue that, besides better tooling (see Section 6.2), a change in mindset is required for cloud developers to fully embrace the additional options for de-

bugging and maintaining applications that the cloud environment provides them.

6.2 Challenges for Cloud Development

In addition to the best-practices and guidelines outlined in Section 6.1, which cloud developers can already implement today, we have also seen that there are a number of areas in which academic research and tool vendors should provide better techniques, approaches and tools.

Academic Research on Infrastructure Evolution. In IaaS clouds, software developers make use of scripting or IaC tools to formally specify and track infrastructure configuration. This means that infrastructure evolution is now tracked in version control systems the same way the evolution of regular code artifacts is. We argue that this provides *new opportunities for academic research* on software repository mining to investigate how infrastructure code evolves over the lifetime of a project, compared to the overall application code. This will allow us to, for instance, discover anti-patterns in infrastructure provisioning code.

Improved Log and Metrics Management. Cloud developers need to anticipate problems prior to deployment and define relevant logs and operational metrics. While this is largely already possible today, there is little support to guide developers *what* and *how* they should be logging exactly. Some study participants have reported that this results in rather excruciating trial-and-error. We argue that correct tracing and metric definition has to be introduced as part of the development workflow. Methods and tooling should be improved to support the process of defining and evolving useful logs and metrics for cloud applications.

Local Reproducibility through Containers. Cloud applications are distributed by default. Our study participants reported that trying to *reproduce faults locally* can be a tedious and time-consuming task. It involves knowing the exact state of when the fault occurred in production (i.e., infrastructure and data state), as well as the capability to replicate the environment and its state locally. We envision methods and tools that have the ability to recreate a distributed environment in a local development environment from a snapshot of when the fault occurred, utilizing containerization technology (e.g., using Docker).

Tools for Developer Targeted Analytics. In the cloud, more metrics are available, both in quantity and dimension. However, in our study we have observed that developers, before utilizing existing metrics, much rather rely on their experience and intuition to solve problems. Besides a required change in mindset, as discussed in Section 6.1, we attribute this to the fact that most monitoring tools are built to be used by operations teams, rather than software developers. Currently, the existing way of delivering metrics is difficult to leverage for developers as it is *not actionable in the development process*. Existing tools need to expose better APIs for data extraction and integration. Future research needs to address how the abundance of data in the cloud can become more actionable for developers and integrate it into their daily workflows[32]. Possible solutions could include integration of data in code views through IDE plugins or within issue tracking systems.

7. THREATS TO VALIDITY

While we have designed our research as a mixed-mode study to reduce threats to validity as far as possible, there are still a number of limitations inherent in our research design. Primarily, the question arises to what extent the 25 cloud developers we interviewed are representative (*external validity*). However, to mitigate this threat, we have made sure to recruit interview participants that are approximately evenly distributed between smaller companies and larger enterprises, as well as between IaaS and PaaS clouds. There still are a more interviewees that deploy on PaaS than on IaaS. We think to have mitigated this concern by having more IaaS participants in the survey, balancing the overall results. Further, our interview participants cover a broad range of experience levels, and work with various kinds of cloud systems on various kinds of applications. A similar threat also exists for the external validity of our survey. We recruited our participants almost exclusively via GitHub, meaning that we are likely to have attracted mostly software engineers who are actively interested in open source development, and who are also following the progress of at least one big open source cloud product. Further, responses were necessarily voluntary, hence we are likely to have attracted a crowd with higher-than-average interest in the topic of cloud computing.

In terms of *internal validity*, it is possible that we have biased our interview partners through the pre-selection of questions and topics, and that we missed entirely unanticipated differences and implications of software development in the cloud. Also, our themes focused on how software development in the cloud is different than in non-cloud environments. Thus, interview participants may have been inclined to overstate differences, leaving out similarities. However, given that no major undiscovered differences and implications were mentioned during the survey either, we judge this threat to be low.

8. CONCLUSIONS

We report on the first systematic study on how professional software developers build applications and utilize specialized tools and data on top of cloud infrastructures and platforms. The insights provided by our study help to better understand how cloud computing has made an impact throughout the software development life cycle. Our findings suggest, among others things, two major developments: (1) developers need better means to anticipate runtime problems in the cloud and rigorously define and collect metrics for better fault localization and (2) the cloud offers an abundance of operational data, however, developers still often rely on their experience and intuition rather than utilizing metrics to solve problems. Methods and tools for developers will, therefore, need to adapt to these required changes in the cloud. From our findings, we extracted a set of guidelines for cloud development and identified challenges for researchers and tool vendors to support developers in these efforts by developing new approaches for managing metrics and making operational data more actionable.

9. ACKNOWLEDGMENT

The authors would like to thank all interview and survey participants. The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 610802 (CloudWave).

10. REFERENCES

[1] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing As the 5th Utility," *Future Generation Computer Systems*, 2009.

[2] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica *et al.*, "A View of Cloud Computing," *Communications of the ACM*, 2010.

[3] M. Hüttermann, *DevOps for Developers*. Apress, 2012.

[4] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-Aware Resource Allocation Heuristics for Efficient Management of Data Centers for Cloud Computing," *Future Generation Computer Systems*, May 2012.

[5] P. Marshall, K. Keahey, and T. Freeman, "Improving Utilization of Infrastructure Clouds," in *Proceedings of the 2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID '11)*. Washington, DC, USA: IEEE Computer Society, 2011.

[6] A. Iosup, S. Ostermann, N. Yigitbasi, R. Prodan, T. Fahringer, and D. Epema, "Performance Analysis of Cloud Computing Services for Many-Tasks Scientific Computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 6, Jun. 2011.

[7] A. Barker, B. Varghese, J. S. Ward, and I. Sommerville, "Academic Cloud Computing Research: Five Pitfalls and Five Opportunities," in *Proceedings of the 6th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 14)*, Jun. 2014.

[8] P. Mell and T. Grance, "The NIST Definition of Cloud Computing," National Institute of Standards and Technology (NIST), Gaithersburg, MD, Tech. Rep. 800-145, September 2011.

[9] G. Lawton, "Developing Software Online With Platform-as-a-Service Technology," *Computer*, vol. 41, no. 6, Jun. 2008.

[10] L. Singer, F. Figueira Filho, and M.-A. Storey, "Software Engineering at the Speed of Light: How Developers Stay Current Using Twitter," in *Proceedings of the 36th International Conference on Software Engineering (ICSE 2014)*. New York, NY, USA: ACM, 2014.

[11] E. Murphy-Hill, T. Zimmermann, C. Bird, and N. Nagappan, "The Design of Bug Fixes," in *Proceedings of the 2013 International Conference on Software Engineering (ICSE'13)*, 2013.

[12] A. Khajeh-Hosseini, I. Sommerville, and I. Sriram, "Research Challenges for Enterprise Cloud Computing," *CoRR*, vol. abs/1001.3257, 2010.

[13] L. Mei, W. K. Chan, and T. H. Tse, "A Tale of Clouds: Paradigm Comparisons and Some Thoughts on Research Issues," in *Proceedings of the 2008 IEEE Asia-Pacific Services Computing Conference (APSCC '08)*. Washington, DC, USA: IEEE Computer Society, 2008.

[14] L. Mei, Z. Zhang, and W. K. Chan, "More Tales of Clouds: Software Engineering Research Issues from the Cloud Application Perspective," *Proceedings of the 2013 IEEE 37th Annual Computer Software and Applications Conference (COMPSAC)*, vol. 1, 2009.

[15] C. P. Bezemer and A. Zaidman, "Performance optimization of deployed software-as-a-service applications," *Journal of Systems and Software*, vol. 87, 2014.

[16] B. Palanisamy, A. Singh, L. Liu, and B. Jain, "Purlieus: Locality-Aware Resource Allocation for MapReduce in a Cloud," in *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis (SC'11)*. New York, NY, USA: ACM, 2011.

[17] R. Zabolotnyi, P. Leitner, W. Hummer, and S. Dustdar, "JCloudScale: Closing the Gap Between IaaS and PaaS," *ACM Transactions on Internet Technology (TOIT)*, 2015, to appear.

[18] P. Leitner, B. Satzger, W. Hummer, C. Inzinger, and S. Dustdar, "CloudScale: A Novel Middleware for Building Transparently Scaling Cloud Applications," in *Proceedings of the 27th Annual ACM Symposium on Applied Computing (SAC'12)*, 2012.

[19] K. Jayaram, "Elastic remote methods," in *Middleware 2013*, ser. Lecture Notes in Computer Science, D. Eyers and K. Schwan, Eds. Springer Berlin Heidelberg, 2013.

[20] B. Narasimhan and R. Nichols, "State of Cloud Applications and Platforms: The Cloud Adopters' View," *Computer*, vol. 44, no. 3, 2011.

[21] P. Gupta, A. Seetharaman, and J. R. Raj, "The Usage and Adoption of Cloud Computing by Small and Medium Businesses," *International Journal of Information Management*, vol. 33, no. 5, 2013.

[22] "2014 State of DevOps Report," Puppet Labs, IT Revolution Press, and ThoughtWorks, Tech. Rep., 2014. [Online]. Available: http://puppetlabs.com/2014-devops-report

[23] R. Shiver, "Survey: Enterprise Development in the Cloud," Gigaom Research, Tech. Rep., 2014.

[24] R. Hoda, J. Noble, and S. Marshall, "Developing a Grounded Theory to Explain the Practices of Self-Organizing Agile Teams," *Empirical Software Engineering*, 2012.

[25] L. Bratthall and M. Jorgensen, "Can you Trust a Single Data Source Exploratory Software Engineering Case Study?" *Empirical Software Engineering*, 2002.

[26] R. P. Buse and T. Zimmermann, "Information needs for software development analytics," in *Proceedings of the 34th International Conference on Software Engineering*.

[27] X. Tang, Z. Zhang, M. Wang, Y. Wang, Q. Feng, and J. Han, "Performance Evaluation of Light-Weighted Virtualization for PaaS in Clouds," in *Algorithms and Architectures for Parallel Processing*. Springer, 2014.

[28] S. Newman, *Building Microservices*. O'Reilly, 2015.

[29] J. Cito, D. Suljoti, P. Leitner, and S. Dustdar, "Identifying Root-Causes of Web Performance Degradation using Changepoint Analysis," in *Proceedings of the 14th International Conference on Web Engineering (ICWE)*. Springer Berlin Heidelberg, 2014.

[30] J. Cito, D. Gotowka, P. Leitner, R. Pelette, D. Suljoti, and S. Dustdar, "Identifying Web Performance Degradations through Synthetic and Real-User Monitoring," *J. Web Eng.*, vol. 14, no. 5-6, Jul. 2015.

[31] P. Leitner and J. Cito, "Patterns in the Chaos - a Study of Performance Variation and Predictability in Public IaaS Clouds," *ArXiv e-prints*, 2014.

[32] J. Cito, P. Leitner, H. C. Gall, A. Dadashi, A. Keller, and A. Roth, "Runtime Metric meets Developer - Building better Cloud Applications using Feedback," in *Proceedings of the 2015 ACM International Symposium on New Ideas, New Paradigms, and Reflections on Programming & Software (Onward! 2015)*, 2015.