Information and Software Technology 127 (2020) 106376

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

Cloud applications monitoring: An industrial study

Damian A. Tamburri[a,*], Marco Miglierina[b], Elisabetta Di Nitto[c]

[a] Jheronimus Academy of Data Science and the Eindhoven University of Technology
[b] Contentwise
[c] Politecnico di Milano

ARTICLE INFO

Keywords:
Cloud monitoring
Applications monitoring
Incident handling
Rapid response organizational structures
Online survey
Industrial study

ABSTRACT

Context: Modern software systems employ large IT infrastructures hosted in on-premise clouds or using "rented" cloud resources from specific vendors. The unifying force across any cloud strategy is incremental product and application improvement against conservation of those resources. This is where monitoring of cloud applications becomes a key asset

Objective: To shed light over the status of monitoring practices in industry, we study: (a) monitoring practices and tools adoption in industry; (b) size and complexity of industrial monitoring problems; (c) the role of software architecture and software process with respect to monitoring strategies.

Method: We conduct mixed-methods empirical research featuring interviews and a web survey featuring 140 + practitioners from over 70 different organizations.

Results: Even if the market makes available a significant set of monitoring tools, our results show a rather unappealing picture of industrial monitoring: (a) industrial decision-makers do not perceive monitoring as a key asset even though the downtime of their applications correlates heavily with the level of automation and responsiveness enabled by monitoring; (b) monitoring is done with crude technology, mostly MySQL querying or similar (e.g., Nagios); finally, (c) incidents are discovered by clients rather than application owners.

Conclusion: We conclude that the road toward the industrial adoption of cutting-edge monitoring technology is still one of the less travelled, presumably in connection to the considerable investment required. Furthermore, the lack of industrial cloud monitoring standards does not help in addressing the proliferation of multiple tool combinations, with varying effectiveness. Further research should be invested in looking into and addressing these major concerns.

1. Introduction

Cloud computing triggered huge technological advancements that drastically reduced entry costs and time to market in the software development scenario [1,2]. Hardware resources that were once manually provisioned and acquired after huge upfront investments, have become accessible via simple API calls, on-demand and paid per hour [3]. Modern software systems span across multiple services provided by different vendors with heterogeneous platforms (*i.e.*, IaaS, PaaS, and SaaS) [4].

In this scenario, effective cloud applications monitoring has become more important to support fast automation release cycles and to allow fast reaction to perceived issues or market opportunities. At the same time, it has become more difficult to achieve [5,6], due to reliance on multiple vendors with different pricing models and heterogeneous stacks and SLAs, which make modern distributed applications hard to be transparently *observed* [7] in an automated and tool-assisted fashion.

With the term *monitoring* in this context we indicate the explicit activity of gathering application front- or back-end measurement data to diagnose, prevent, or recover from cloud applications errors (e.g., runtime execution incidents which can include application performance slowdowns, application hangs, etc.) or failure [8], referred either to application-level (e.g., containers, production-code bundles, APIs, libraries, etc.) or infrastructure-level (e.g., virtual-machines, orchestration node-type scripts [9], network etc.) components.

While a considerable number of monitoring tools, both commercial and open-source ones, proliferated in the last few years [10,11], no clear established solution has yet arisen. Big corporations with high expertise such as Google, Facebook or Netflix are able to develop the appropriate solutions for their scales. Most of the other companies either do not monitor, implement custom solutions or use some custom composition of monitoring tools among the thousands of existing ones [10].

* Corresponding author.
E-mail addresses: d.a.tamburri@tue.nl (D.A. Tamburri), marco.miglierina@contentwise.com (M. Miglierina), elisabetta.dinitto@polimi.it (E.D. Nitto).

https://doi.org/10.1016/j.infsof.2020.106376
Received 27 February 2019; Received in revised form 6 March 2020; Accepted 29 June 2020
Available online 3 July 2020

To identify the main challenges and open problems that practitioners are facing while monitoring their cloud stacks, we conducted a mixed-methods empirical study. First, using structured interviews, we enquired 12 practitioners from 7 different organizations. Then, using this initial feedback we structured a multi-source, opt-in online survey.

The online questionnaire featured 38 closed questions prepared according to cross-examination guidelines from Kvale [12] and Pettigrew [13] so as to cover all topics, themes, and keywords emerged in our pilot interview study and its analysis. We received responses from 141 practitioners in about half as many distinct organizations. Responses were attained from 5 distinct sources used (i.e., Reddit, yCombinator, StackOverflow, SourceForge News, and Paystack) over a total period of 10 months. The relatively strong participation to the survey is itself a first result - the topic is perceived by practitioners as extremely timely and highly relevant. Analysing our results further, we identified common tool-adoption patterns as well as incident discovery and handling practices.

Three key findings emerge: (a) the most impactful challenge resulted to be the lack of standards in an overpacked market of monitoring tools - over 90% of our responders identified the lack of standardisation and the over-proliferation of monitoring solutions as clear problems leading to monitoring mishaps and misuse; (b) both SMEs and large organizations are mostly largely unaware of the potential behind analysing monitoring data for software evolution and modernisation - our data indicates that 1 out of 3 organizations do track historical data concerning their cloud application incidents but does not use such data for re-architecting or similar refactoring activities; (c) even the biggest of industrial players in our sample is unaware of its products' **observability**, that is, the ability to monitor and keep track of software functions via automated means, and how observability evolves with architectural complexity [14] - while we expected the observability to evolve jointly with architecture complexity, we noticed that there is an erratic if no correlation at all between these two key dimensions across our entire dataset.

The practical and academic implications of our research are manyfold: (a) a better understanding of the challenges perceived by industries; (b) a better focus on future directions for common strategies around monitoring, e.g., reducing entry-cost and learning-curves; (c) a reference yardstick for current monitoring research and practice against valuable industrial hard data; (d) an overview of the maturity of monitoring assets directly from their respective industries.

In conclusion, we argue that companies should carefully seek to prepare for the upcoming monitoring boom. At the same time, researchers should be prepared to address the technical and organisational challenges around the industrial issues we identified and highlighted in the scope of this article. While the technology to be applied for monitoring today exists, our evidence shows that the approaches, processes and skills needed to apply monitoring technology in concrete and diverse cases is still to be distilled from a large variety of industrial research endeavours in the area.

Paper Structure. The next section outlines the background necessary to properly grasp the contents of this article as well as related work in our area of study. Section 3 defines the research design behind our article. Section 4 outlines our results, Section 5 presents possible threats to validity and Section 6 discusses the results and offers an analysis of the insights we procured from practitioners in industry. Finally, Section 7 concludes the paper.

2. Background and related work

2.1. Monitoring: terms and definitions

Quoting from Netflix Inc., and in agreement with many other top-players, cloud software should be: “API-driven, self-service, and automatable” [15]. In this context, *monitoring* becomes a necessary prerequisite for self-service and automation and requires the adoption of explicit technologies, driven using explicitly-trained organisational structures, and accounted using appropriate frameworks.

Restricting the domain to the software engineering field, the subject being monitored is usually a system composed of components. Thus, monitoring can be redefined as “the action of observing and checking the behavior and outputs of a system and its components over time”[1]. Several other definitions of monitoring can be found, among which:

1. Bertolino [16] defines monitoring as observing “the spontaneous behavior of a system” and, given a specification of desired properties, checking “that such properties hold for the given execution”;
2. Fatema et al. [17] define monitoring as a “process that fully and precisely identifies the root cause of an event by capturing the correct information at the right time and at the lowest cost in order to determine the state of a system and to surface such state in a timely and meaningful manner”;
3. According to Cockroft[2], monitoring refers to both “problem detection and diagnosis” and “measuring business value”. Cockroft ultimately defines business value as “customer happiness, cost efficiency, safety and security, and compliance”;

From a technical perspective, a monitoring activity acquires information about some **metrics**, i.e., directly and atomically measurable properties of a phenomenon that can be quantitatively determined. Example: response time is a metric measuring the “elapsed time between the end of an inquiry or demand on a computer system and the beginning of a response” [18]. Each single measurement of a metric can be called **monitoring datum** (Example: the authentication service took 100 ms to respond). The subject of monitoring is a **resource**, for instance, a web server, a database, a virtual machine, a software container, an application component. The monitoring infrastructure can be generically seen as composed of **data collectors**, in charge of acquiring data by observing resources, **data analyzers**, in charge of processing (e.g., filtering or aggregating) monitoring data, and **monitoring dashboards**, in charge of support visualization of monitoring data by end users.

2.2. Monitoring: dimensions of analysis

According to the classification introduced by Barrat[3], a monitoring tool can perform one or more of the following actions on monitoring data: (i) collect, (ii) transport, (iii) process, (iv) store, (v) present. A data collector is usually either a daemon running on the monitored host or an agent scraping data from monitoring APIs exposed by the resource being monitored (e.g., JMX). Monitoring data can also be collected via code instrumentation, developers may use APIs to collect data. A tool transporting monitoring data is able to move data from a tool to another. Some tools may have transport capabilities implemented ad hoc, others may use existing general purpose solutions such as message brokers. Monitoring data per se is useless if it is only collected, it needs to be used for a purpose. Data can be processed, for example, to extract higher level knowledge from raw data or to verify conditions on it. Data can also be stored, for example in a time series database, or it can be presented to the user, for example via a dashboard, and let a human understand problems and patterns and take actions. Processing can be executed either in a distributed way or centrally on a single server. Distributed processing means analyzing data on several machines in parallel and eventually aggregate results on a single host. This solution can reduce network traffic and is more scalable, however it prevents more sophisticated processing algorithms which cannot be parallelized. Centralized processing requires data to be transported to a server where it is aggregated and analyzed.

1 https://blog.freshtracks.io/monitoring-is-dead-long-live-observability-235b62f4d1d1.

2 https://www.slideshare.net/adriancockcroft/monitoring-challenges-monitorama-2016-monitoringless.

3 http://serialized.net/2011/02/getting-more-signal-from-your-noise/.

Besides the above classification based on roles, other classifications identified in literature [19] mainly focus on how data is collected from resources. Collection can be either passive, also known as non-intrusive, or active, also known as intrusive. It is passive when there is no need to modify the resource to be monitored. Collecting data about the network activity through packet sniffing, for example, is considered to be passive. On the opposite, active monitoring is performed when the resource requires some modification for exposing data to the data collector (e.g., API). Another dimension of classification in collection techniques is push-mode versus pull-mode. Monitoring data can be either pushed by a monitoring agent to a monitoring server, or pulled from monitoring agents by a specific monitoring server or monitoring stack (e.g., the well known Elastic-Search/Logstash/Kibana monitoring stack, known as ELK[4]).

Furthermore, another dimension typically considered in monitoring approaches and systems is stateful vs. state-less log processing. On the one side, a monitoring tool can maintain information about the state of resources and model the inter-relationships among components. On the opposite, if no information is maintained across subsequents events, the tool is said to be stateless. Stateless tools are easier to scale.

A further classification concerning how monitoring data is processed is the expressive power of the configuration language, which actually tells what a user can do with monitoring data. A configuration language may allow to (i) report time series with a configurable granularity, (ii) offer statistical aggregation capabilities (e.g.: maximum, minimum, average), (iii) define thresholds, (iv) define alarms or actions to be taken under given circumstances, (v) provide filtering capabilities.

Another important technological distinction is between Application Performance Monitoring (APM) and server monitoring. The two type of tools are orthogonal. APM is responsible of monitoring user experience and how user traces behave across distributed systems, while the second is concerned with single nodes, their availability and behaviour, unrelated to the user interaction with it.

Lastly, from the business perspective, a tool in general can be classified in terms of its licensing model, i.e., open-source vs commercial, and its deployment model, i.e., self-hosted vs cloud.

2.3. Related surveys

While no rigorous industrial survey exists covering precisely the exact same scope as ours, several surveys and literature reviews exist, mainly in the grey literature, which are related to the target topic.

For instance, the SolarWinds survey[5] as well as the Netfort Study[6] focus on a specific aspect of monitoring, that is, network monitoring. Both surveys do not offer sufficient depth and methodological detail to establish conclusion and construct validity [6]. On the contrary, the intent of this paper is to develop a rigorous and systematic investigation of industrial and academic perspectives over cloud applications monitoring from a wider lens of analysis. Other works, such as [20] and [21] focus on a systematic analysis of monitoring (in particular, network monitoring) tools, but they do not provide hints on their adoption in practice.

On another front, from a more general and research perspective within the domain of DevOps software engineering, the work which is most closely related to ours in terms of objectives is represented by the effort of Aceto et al. [19] who perform a systematic study of cloud monitoring technology from a research literature perspective. Aceto et al., however, focus on providing an overview of the state of the art as opposed to a glimpse over the state of practice — the latter is what we offer in the scope of this manuscript; therefore, the two surveys can be seen as compounding reference material for the practitioner or academician who is entering the domain of cloud applications monitoring. From a specular perspective, Fatema et al. [17] provide a general overview over the existing tools and automated solutions currently used for monitoring cloud applications.

From a broader perspective, Jabbari et al. [22] address the research question of finding a distilled definition of DevOps as a framework for organisational, social, and technical lifecycle management. Their efforts are more poured into mapping the literature towards offering a precise definition from the state of the art. Similarly, M. Kersten regularly uses his column on IEEE Software to outline the evolution of DevOps, e.g., in terms of tools proliferation [23]. Further back in time, several surveys exist that address software continuous refactoring, software version merging, dependency management, most prominently by Mens [24,25] or even in the field of infrastructure management, infrastructure-as-code [26] or the well-known survey of cloud service providers by Prodan and Ostermann [27].

3. Research design

3.1. Research problem, goals, and questions

The major motivations behind this study stems from our industrial collaborations in the scope of the MODAClouds EU FP7 project [28]. In the context of the EU project, we observed that there is much disarray across industries when it comes to their monitoring assets, their structure, characteristics and general quality. From this general observation, in the scope of this article, we seek to assess the industrial awareness on custom-made monitoring infrastructures, their shortcomings, and limitations while, at the same time, assessing industrial understanding and visibility over the current tooling and incident handling methodologies.

From the research problem and goals outlined above, the following master research question emerges for the context of our study:

“What are the issues, tools, and procedures currently used for monitoring cloud applications and incident-handling in industry?”

Stemming from the above research question, we derive a set of 8 sub-research questions (see Table 1), namely:

1. *Is monitoring perceived as a fundamental asset?*
   Definitions and Rationale. Monitoring is, by definition, not an activity which generates direct and immediate Return-On-Investment. Therefore, it is itself a medium to long-term investment which can procure competitive advantage, i.e., an *asset* as part of an *asset management* strategy [29]. Consequently, such a strategy may reflect a clear and distinct organizational decision or an emerging tactic. We aim at establishing the current status over this issue, to increase practitioner awareness.
2. *Do all companies in our sample monitor their software systems? How?*
   Definitions and Rationale. Monitoring is defined as the act of continuously measure the parametric features of a software system, its visible and invisible qualities for the purpose of instrumenting corrective, preemptive, and proactive maintenance or improvement [30]. The literature identifies monitoring as a fundamental asset to elaborate corrective or preemptive actions over a running software system but the act of monitoring is considerable and, as systems scale up, this gets even more expensive. Our research goals also encompass establishing the current conditions of investment around monitoring in industry.
3. *What are the people/roles involved with monitoring?*
   Definitions and Rationale. Monitoring usually demands a more or less complex organisational structure [31] designed for but not limited to: (a) *preparedness* — i.e., the series of organisational protocols being enacted to avoid service discontinuity incidents; (b) *firefighting* — i.e., the organisational protocols being enacted to address and tentatively fix services after discontinuity incidents do manifest; (c) *recovery* — i.e., the organisational protocols enacted to recover from the discontinuity incidents. The above organisational scenarios

[4] https://www.elastic.co/elastic-stack.
[5] https://techcloudlink.com/wp-content/uploads/2019/09/A-Guide-to-Enterprise-Network-Monitoring.pdf.
[6] https://www.netfort.com/netfort-news/network-monitoring-survey/.

Table 1
Research questions, an overview.

RQ-id	Question
1	Is monitoring perceived as a fundamental asset?
2	Do all companies in our sample monitor their software systems? How?
3	What are the people/roles involved with monitoring?
4	How are incidents discovered and handled?
5	What is the Pandemic Ratio for incidents?
6	What are the most critical challenges perceived when trying to make a system observable?
7	Is there correlation between complexity of cloud architectures and the time of system unavailability?
8	Is there any relation between systems observability and architecture complexity?

usually require specific roles, that is, professionals whose organisational goal is to address each phase individually. Our research goal also encompasses establishing the current conditions of investment around such organisational structures, if any.

4. *How are incidents discovered and handled?*
   Definitions and Rationale. As a sub-research question to the previous one, we aim to zoom into the practices that address fire-fighting and incident recovery, since the practices involved in these two phases are aimed at reducing as much as possible the service discontinuity times. Consequently, we are interested in reporting and studying more deeply such practices and their consequences over service discontinuity.
5. *What is the Pandemic Ratio for incidents?*
   Definitions and Rationale. We define the Pandemic Ratio as the sum of people who are involved on average for crysis resolution, when incidents do in fact manifest. With the above five sub-research questions we aim at understanding whether there exists a recurrent organisational structure specifically designed to monitor systems and address operational incidents; for example, a pattern for the pandemic ratio may reveal vital descriptive quantities and features to be used as a best-practice for cloud monitoring.
6. *What are the most critical challenges perceived when trying to make a system observable?*
   Definitions and Rationale. As previously introduced, we define observability as the degree to which software architecture elements and properties in a system can be monitored with available off-the-shelf or ad-hoc technology and without specifically instrumenting the code during operations. We seek to understand the amount of work being invested in making software architectures monitored by-design.
7. *Is there correlation between complexity of cloud architectures and the times of system unavailability?*
   Definitions and Rationale. We define a simplistic measure of software architecture complexity SA_C for cloud architectures as the number of components C in the architecture multiplied by the number S of software architecture styles—as extracted from software architecture reference textbooks such as Bass et al. [32]—reported for that architecture normalized by the maximum number of architecture elements reported in the sample by our respondents. The reason for this simplistic and approximate notion of architecture complexity by multiplication, is grounded on the need of allowing the two involved quantities—different style, and number of components respectively—to account equally and linearly within the metric.
   In the scope of the aforementioned definition, we seek to understand whether more complex architectures reflect more incidents and/or whether such increased complexity corresponds to specific challenges, practices, fallacies, or pitfalls.
8. *Is there any relation between systems observability and architecture complexity?*
   Definitions and Rationale. We aim to understand whether there is a software style extracted from literature [33,34] or recurrent level of architecture complexity across our sample which is most consistent with a more observable and maintainable architecture in the cloud.

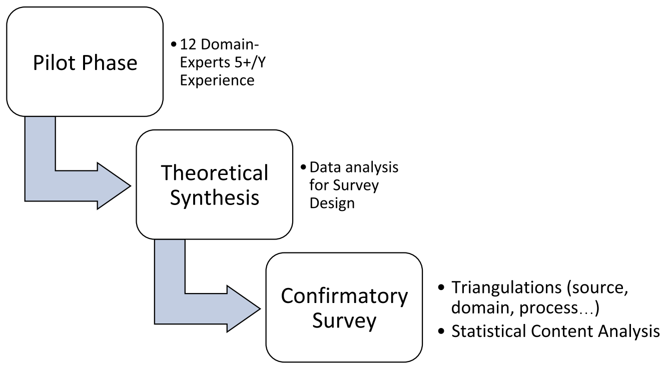
Pilot Phase
• 12 Domain-Experts 5+/Y Experience
Theoretical Synthesis
• Data analysis for Survey Design
Confirmatory Survey
• Triangulations (source, domain, process…)
• Statistical Content Analysis

Fig. 1. Research design, an overview.

3.2. Research methods

As previously introduced, our research design features a mixed-methods approach whereby we aimed at having a complete and comprehensive overview of the industrial scenario regarding the practical use of monitoring. With this goal in mind our intention was to form a large-scale online survey using solely the insights that can be gathered from highly-expert practitioners themselves.

In so doing, our study was designed to feature three phases, outlined in Fig. 1:

1. **Pilot phase.** In the scope of a pilot study with domain experts, we enquired experienced (5+ years of field experience) industrial practitioners from our own networks using generic and open-ended interviews to elicit the general understanding and concepts in cloud monitoring;
2. **Theoretical Synthesis.** Using pilot data, we distilled the concepts, definitions, and variables in the domain of cloud systems monitoring and used the outcome theory to structure our online survey;
3. **Confirmatory Survey.** We finally employed an online survey in all venues suggested by practitioners in phase (1);

The rest of this section outlines the 3 phases in detail, following the exact phases in which the study was executed. In total, the process lasted about 10 months, six of which have been dedicated to the pilot phase and about two to the confirmatory survey.

3.2.1. Pilot phase

Pilot-study interviews Starting from our sub-research questions (see Section 3.1), a set of 19 open questions were prepared to be used as starting point for a structured discussion[7]. The structured discussion guided

[7] Questions are available online for replication purposes: https://tinyurl.com/yahtmmaj.

by a questionnaire fulfils two research design objectives: (a) to guide the discussion towards refining our understanding of the interviewees' company characteristics, core business, the role of the interviewee within that company and the general characteristics of their maintained software products; (b) help the interviewer to elicit the practices, tools and people involved in the operations, maintenance and incident handling in the interviewee's company. Finally, the questionnaire was structured to include an *implicit* guide to have the interviewees expose problems, limitations, and open points regarding their own and other available monitoring tools and practices.

Pilot-study protocol The interview was conducted by one researcher while materials and data-gathering was conducted with enfield notes and observations by a direct observer during the interview itself. Rather than selecting candidates, thus introducing a bias in our study, we decided to share questions directly with our industrial contacts, together with an opt-in open invitation to participate to the study. The invitation featured an element of snowball sampling [35] whereby our contacts were encouraged to better identify the most qualified person for the interview and/or to gather such necessary information in advance, if needed. Face-to-face interviews were scheduled weekly, over a period of 3 months, and took about 1 h each. When face-to-face meetings were not possible for time or spatial constraints, conference calls were organised instead using either Microsoft Skype, UberConference or the WebEx video-conferencing systems.

Pilot-study sampling In the scope of Phase 1, we aimed at ensuring that our results were attained considering an appropriately-sampled population of companies developing and maintaining software systems covering most diverse architectural styles. Therefore, the following architectural styles were covered equally in our initial opt-in practitioner invitations:

- *Microservices architectures* - microservices were considered since they are an increasingly popular development approach and monitoring is challenging because of their intrinsic distributed nature;
- *Multi-tenant hosting* - companies offering multi-tenant solutions need to take special care of monitoring in order to guarantee the SLA agreed with the different users when resources are shared among them;
- *Sensor networks* - sensor-network companies have particular requirements such as energy efficiency constraints and unreliable infrastructures, requiring special monitoring attention;

Moreover, companies in our sample were chosen to cover for areas orthogonal to the previous classification, as follows:

- *Web applications* - we are interested in accounting for applications where monitoring is fundamental to provide required quality of service to users, which can cope with often unpredictable traffic spikes;
- *Real-time services* - we are looking for cloud software where timing constraints need to be monitored and addressed since they procure heavy impacts on returns and revenue (*e.g.*, gaming or online trading);
- *Technology providers* - these companies offer software solutions, installed on customers machines, and provide support for monitoring on heterogeneous hosting solutions;

Sampling our network of industrial contacts with the above control factors, we gathered a total of 20+ hours worth of interviews from 12 practitioners belonging to 7 different organizations over a 6-month period. Table 2 summarises the population we were able to analyse with in-person interviews. The resulting sample covers 80% of our expected population (see above), with the exception of "sensor-network" and "real-time" service architectures, which are only partially covered by company ID = "I6" (see Table 2).

Interview analysis

Interviews were analysed by means of taxonomy analysis [36] and thematic coding [37]. Our objective was to obtain: (a) confirmation over the validity of the questions, concepts, notations, and issues selected to be addressed in the subsequent online survey phase; (b) answer-sets to the online survey questions. This analysis allowed us to refine a set of closed survey questions, responses as well as hypotheses to be verified during phase (3).

The themes and taxonomy generated from the afore mentioned analysis are reported here below mapped to the research questions in this study (in round brackets) and hence the subsequent online survey questionnaire preparation.

First, the interest in monitoring exceeded our expectation (Q1). All interviewees considered monitoring a fundamental practice. It is performed in different ways, but all of them are doing it (Q2) and, most of all, no one asserted that monitoring is useless or that is not worth investing on it.

Small companies usually prefer cloud hosting solutions and tend to experiment with more recent technologies (e.g., Docker containers) and patterns (e.g., microservices). There is usually one team taking care of both development and operations (Q4). They do not invest much in monitoring, rather they prefer to use solutions that are available at small cost and with- out implying excessive effort (Q2). Incident handling is mainly diagnosed by means of manual log analysis (Q3). The most important metrics are availability and their product-specific business metrics.

In medium companies developers and operators are usually separated teams which have different responsibilities. Usually only operators receive alerts from the monitoring system (Q4). Operators are skilled enough to setup a monitoring platform using one or more open source monitoring tools for both system and application metrics and metrics can be shown on dashboards, usually accessed by operators only. Logs are collected and accessed using the ELK stack (Q2).

Large companies usually manage more complex systems and therefore tend to invest in more sophisticated monitoring solutions. Also, companies where incidents can cause huge losses prefer to pay for monitoring solutions and related support. These expensive solutions allow them to use advanced machine learning and big data techniques for analyzing monitoring data (Q2).

In all companies, alerts are mainly sent via email or via SMS in case of critical issues (Q3). If there is any automatic remediation based on some monitoring check, it involved only restarting a service or scaling up and down hosts (Q3). Issue detection is mainly based on threshold, only companies with strong IT departments or those investing a lot for third party monitoring tools have more sophisticated machine learning tools for detecting problems (Q2, Q3). In most of the cases custom code is used and monitoring data is analyzed using tools that are not monitoring-specific like relational databases or spreadsheets (Q2).

During the interview I3 a limitation of today's monitoring tools is identified, which is the lack of standards and integration. The company use different data collecting tools on the same machines with different scopes, with data flowing through different paths according to different protocols. Some metrics even overlap between each other.

This is for example a challenge addressed by our own internal solution for monitoring, namely, the Tower4Clouds toolsuite[8] (Q6) since it enforces the usage of the same protocol for sending data to one single data analyzer which is then able to route processed data to different destinations according to user-defined monitoring rules.

Another challenge that was identified during this first phase is the high market demand and competitiveness. Startups (interview I7) or emerging companies (interview I1) that are trying to address new market opportunities, require a speed to implement new features that delays the application of quality assurance techniques unless they require very small setup efforts (interview I6).

3.2.2. *Theoretical synthesis*

The survey form featuring closed questions was perfected iteratively with external reviewers from Academia, in order to improve clarity and

[8] https://github.com/mmiglier/tower4clouds.

Table 2
Interviewees population - when a company (ID in column 1) had multiple ongoing projects the interview was directed towards the major project, asset, or product; IT headcount is the number of IT people working on that selected product (or product line) against the total across the company.

ID	Domain	Role	Total/IT headcount	Product	Architecture	Software stack	Monitoring stack
I1	Travel	CTO	100/15	B2B and B2C web services	Microservices	Scala/Java, Akka, Docker containers, Amazon EC2	Amazon Cloud Watch, custom availability checks
I2	Software	Research Software Engineer in Innovation Lab	100/25	Stream processing tool for news validation (prototype)	Event driven, Microservices	Java, MongoDB	Custom code instrumentation
I3	Software	System Architect	200/150	SaaS services for enterprises business management	3-tier, multi-tenant DB	Java, Tomcat, C+, SQL Server, HAProxy, Linux/Windows, third party IaaS	Focused on DB metrics, Graphite, Grafana, Nagios, Icinga, ELK stack
I4	Software	Chairman, CTO	5/3	SaaS recommending system for business management	monolith with Multi-tenant DB	Java, Google App Engine, Google SQL	Focused on DB and business metrics, Google Analytics, Google App Engine Dashboard
I5	Software / Hardware	Distinguished Engineer	~1.2M/400K	Enterprise infrastructure system	Mainframe solution	Proprietary Embedded Solutions	Proprietary monitoring tools, analysis and alerting tools for all IaaS and PaaS services they provide
I6	Software / Hardware	Solution architect	~1M/100K	Full-Stack Product Presence	All	All	Services offered to financial corporations include expensive and sophisticate on-premise monitoring tools, e.g., Dynatrace; others prefer monitoring-as-a-service with little setup efforts. Most companies use our custom solutions featuring well-known tools (relational DBs, and spreadsheets for data-analysis)
I7	Finance	CTO	40/10	e-payment platform	Microservices	Java, third party services (*e.g.*, Google API), relational database, Amazon S3, Amazon EC2, Amazon SNS	Amazon Cloud Watch, manual log checking (soon moving to ELK stack), custom library pushing metrics to Cloud Watch and/or to report table

avoid ambiguities. First, the survey was shared among 3 colleagues of our research group - these colleagues were expert on incident management and were asked to review the survey form by trying to fill it and provide feedback. Then, the form was shared with 3 colleagues outside of our research group for an identical but external review process. Finally, the survey was sent to all practitioners who were initially involved in our interview phase (1) - our goal was collecting a baseline set of answers with which to check the proposed answer-set for closed questions and process any provided comment or question. The agreement between our expectations and the feedback received by the reviewers was evaluated using the well-known Krippendorff α coefficient [38] according to which the coded data was analysed by an independent third-party and a confirmation rate (i.e., an agreement cipher, “1” in our case) was left if the coding was agreed upon — at this point, K_α is measured as the ratio of agreement. The iterative review exercise was repeated until K_α was evaluated to $> > 0.800$, which is a standard reference value for K_α-based content Inter-Rater Reliability assessment [38]. Questions were added if disagreement existed between our desired information content and the interview data from either the reviewers or our industrial contacts. The process of adding questions was iterated until an acceptable agreement was achieved, as measured by the aforementioned K_α procedure — in the scope of this process, a total set of four rounds were invested in this endeavour until $K_\alpha = 0.84$.

The final form distilled through this process consisted of 38 questions and consisted of 3 Sections. Section 1 addressed questions about the respondent role and the company. Section 2 focused on understanding the software system architecture, its complexity, its development and organisational history as well as its lifecycle and releasing process. Finally, Section 3 focused on monitoring procedures and tools as well as how software incidents are solved. The form was fed online using Google Forms[9].

3.2.3. *Confirmatory survey*

A total of 141 responses were collected employing random, stratified sampling [39]. More in particular, our sample stratification strategy employed venues suggested by interviewees in Phase (1) of our study while a randomised sampling strategy reflected sharing the submission link openly with a brief introduction to topic and contents of the survey. The following sources were considered:

- timed posts on three social media sites (Facebook, Twitter, and LinkedIn);

[9] The form is still available for reference: http://tinyurl.com/lxegx6m.

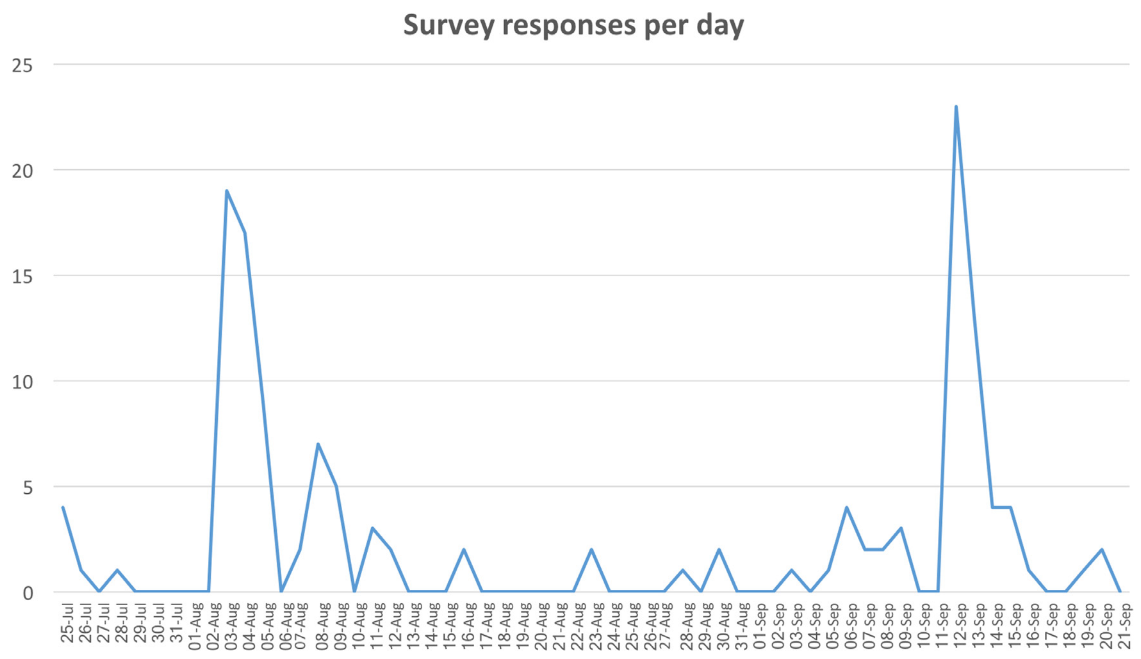

Fig. 2. Survey responses rate over time — the entire data-sampling phase lasted for 54 days.

- timed posts on all MeetUps[10] focusing on IT monitoring and cloud infrastructures — we asked MeetUp leaders to share the survey among MeetUp participants;
- timed posts on Reddit, under `/r/sysadmin` and `/r/devops` subreddits;

The form was left open for answers for 54 days, following the principles of non-invasiveness [40]. During this period of data acquisition the responses trend was monitored in order to understand the most impactful communication techniques - the resulting trend is shown in Fig. 2.

The trend clearly highlights three spikes that are related to two different kinds of advertising actions. The first spike, in the beginning of August, when the message was broadcast to around 250 industrial contacts in our own network - we requested this network to share the research survey to any acquaintance working in IT who may be able to address the survey goals. The second spike appears a few days later when the same message was broadcast to an additional set of 100 industrial contacts. The third and final spike is registered in the middle of September when the research was advertised on Reddit under the `/r/sysadmin` subreddit, which counted around 150.000 readers and revealed to be the most impactful channel.

For data analysis we: (a) computed descriptive statistics that offer an overview of our entire survey sources and resulting data - our goal was to offer a precise overview of the *status-facts* distilled in our survey; (b) queried the dataset using our research sub-questions (see Section 3.2.1) as queries for the Google Answers query engine - our goal was to elicit direct responses to address each question individually. Finally, with respect to our SRQs, we evaluated Pearson or Spearman correlations (when no assumption of linearity was possible) among the quantities in our dataset, making sure to adopt standard p-value measurements of $p < < 0.05$, for confidence of correlations to instrument proper hypothesis testing.

4. Analysis of the results from the confirmatory survey

This section analyses our survey results. More specifically, Section 4.1 outlines descriptive statistics for our survey results while Section 4.2 analyses the results in the light of the sub-research questions defined in Section 3.1.

4.1. Descriptive statistics

Our dataset appears quite heterogeneous in terms of market segments, as shown in Fig. 3: 43.7% of respondents belong to technological industries; the rest of the dataset is distributed across several areas: financial services, telecommunications, health care, consumer care and others.

Fig. 4 shows that the sample is well distributed with reference to the size of the companies. 25% are under 20 employees, 21% are between 21 and 100, 22% between 101 and 500, 22% between 501 and 10.000 and the remaining 10% over 10.000.

In terms of IT department size (Fig. 5) 30% of the companies has less than 5 IT employees, 26% between 6 and 20, 18% between 21 and 100, the remaining 26% has an IT bigger than 100 employees.

Respondents were usually covering multiple roles, however, 96% of them were either developers, system administrators or DevOps engineers. Among these, 39% covered exclusively a development role, 14% exclusively a system administration role, 12% exclusively a DevOps role.

Software systems are distributed as depicted in Fig. 6, more than a quarter of the companies develop, maintain, and commercialise enterprise software in several sectors and featuring several software applications and architecture types (see Fig. 6).

Furthermore, the distribution of architectural styles existing across our dataset is depicted in Fig. 7. Not surprisingly, the figure shows a clear predominance of client-server and service-oriented architectures with similarity of occurrence between monolithic and microservice architecture styles, which are opposite by definition. Most software systems are composed of a small number of deployable components: 72% have less than 10 components, with 38% having less than 3 components. 20% have between 11 and 50 components, the remaining 8% over 50 components.

Fig. 8 depicts the hosting solutions adopted by the interviewees. 36.17% of them use the public cloud, 11.4% of which using it as the only hosting solution for their entire system.

Regarding the type of deployment solution there is a considerable percentage of companies that deploy on bare metal and on IaaS (Fig. 9).

[10] https://www.meetup.com/.

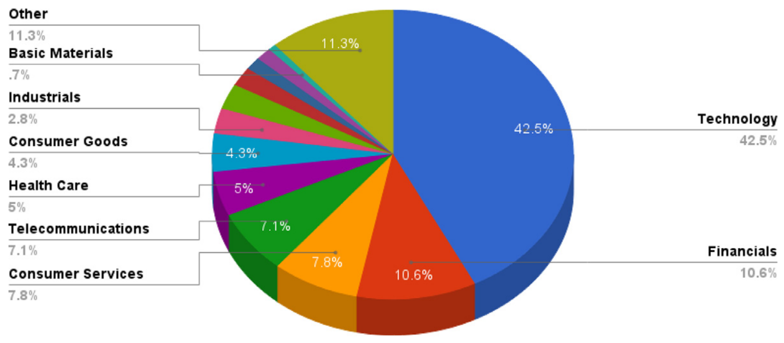

Fig. 3. Industry types distribution.

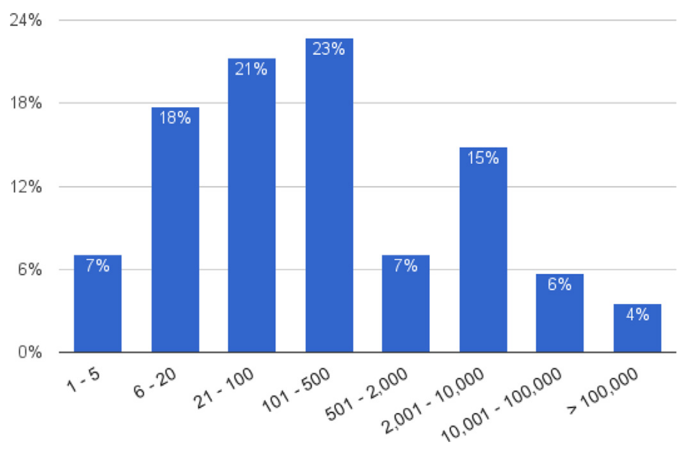

Fig. 4. Industry sizes distribution.

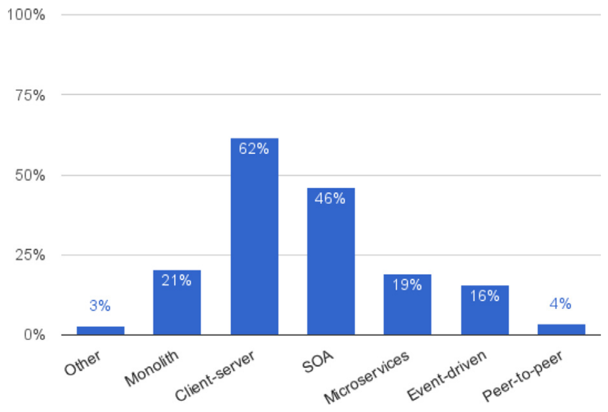

Fig. 7. Software architecture styles in our dataset.

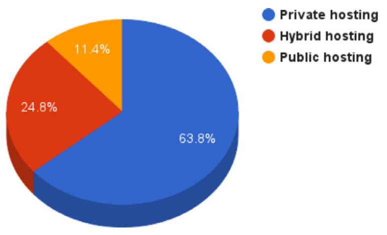

Fig. 8. Hosting solutions distribution.

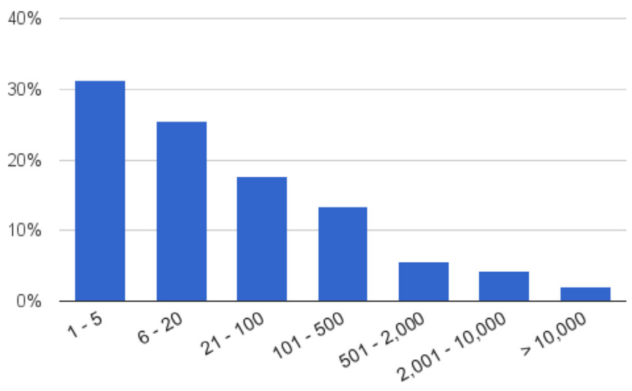

Fig. 5. IT sizes distribution.

However, a good part of our sample of companies are also using more ephemeral solutions or higher abstraction layers.

We also collected a subjective score of the automation level of the release cycle (Fig. 10), the release rate (Fig. 11) and, finally, the average workload their systems endure (Fig. 12). Only 4 companies release more than 10 times per day. These are large companies, 2 of them having more than 100.000 employees. They all use SOA architectures with 100+

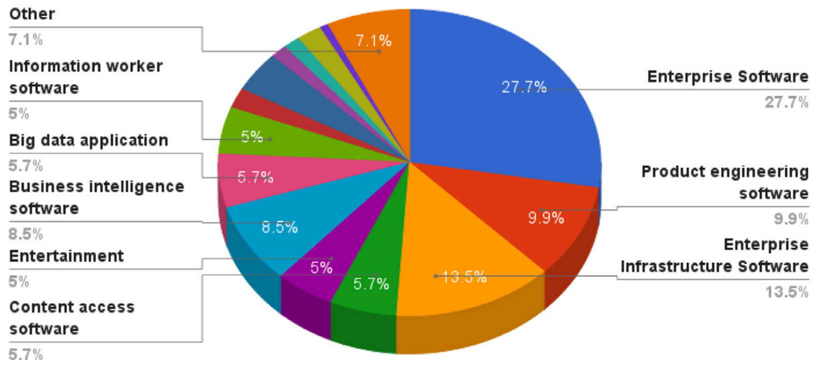

Fig. 6. Software systems distribution.

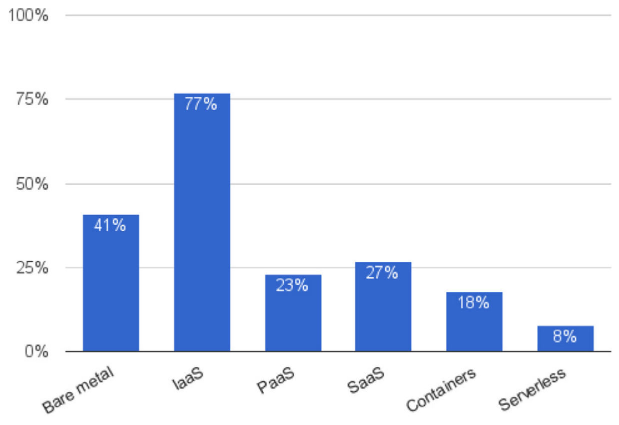

Fig. 9. Deploying model distribution.

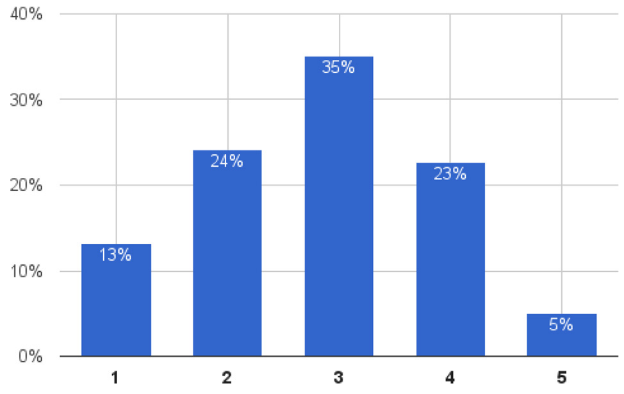

Fig. 10. Rates distribution for the level of automation in the release cycle. Average score is 2.83.

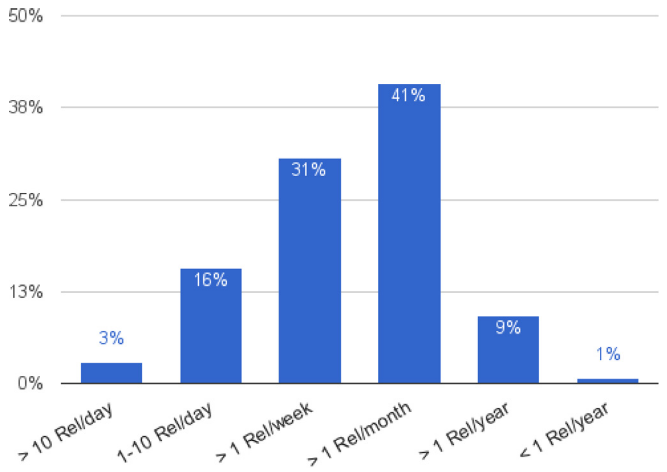

Fig. 11. Release rates distribution.

(micro-)services, serving 1000/s requests. Only 1 company releases less than once a year - it is a medium technological company with a small IT following a relatively big system composed of 20 to 50 components, privately hosted, serving hundreds of requests per second.

4.2. Analysis of results in the light of research sub-questions

Results are outlined addressing each of the research sub-questions listed in Section 3.1.

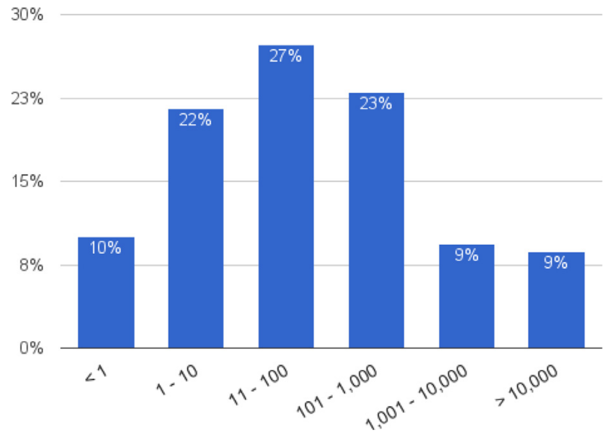

Fig. 12. Workload distribution in terms of users requests per second.

4.2.1. RQ1: Is monitoring perceived as a fundamental asset?

The large participation to the survey, even though the number of questions was high, is a first indicator of the interest that monitoring has across the industries. Nevertheless, when the respondents were asked whether they had planned to improve their monitoring asset, 12.8% of them answered *"No, because investing time and money on monitoring is not perceived as a profitable investment"*. According to our data, this perception can be found more concentrated in companies which are:

- managing a smaller number of components,
- hosted on private clouds,
- with smaller automation processes in action and
- releasing less often.

These companies, resulted to give the observability [7] of their system a lower score (mean of 2.59) than the average of the entire sample (3.1). From the answers they gave, these companies result to experience longer unavailability time on average, when incidents occur:

- more than 20 min in 84% of such companies, against 63% of the entire sample;
- more than 1 h in 39% of such companies, against 28% of the entire sample.

Also, the average time to diagnose the cause of an incident is negatively impacted in these set of companies which do not consider monitoring worth the investment: only 28% of them are able to diagnose a problem in less than 20 min, against the 44% of the entire sample.

Finding 1. Monitoring is perceived as a fundamental asset only for a specific profile of the industry that constitutes around 15% of the total.

4.2.2. RQ2: Do all companies in our sample monitor their software systems? How?

Fig. 13 shows the cumulative number of monitoring tools reported across our sample: 18% of the surveyed companies do not have any deployed system for monitoring, with pretty much the same percentage using at most one or two tools. These companies reportedly operate using manual checks via terminal commands such as *ping, ssh* or *grep* if there is any problem, as well as to troubleshoot. With respect to the rest of our data over these companies, it is clear that they handle usually small systems: 60% of them manage between 1 and 3 components, 32% between 4 and 10. They also have slow release rates compared to companies that practice continuous delivery. In fact, none of them releases more than once per day. Conversely, such companies are subject to small workloads and reportedly experience limited complexity of the system

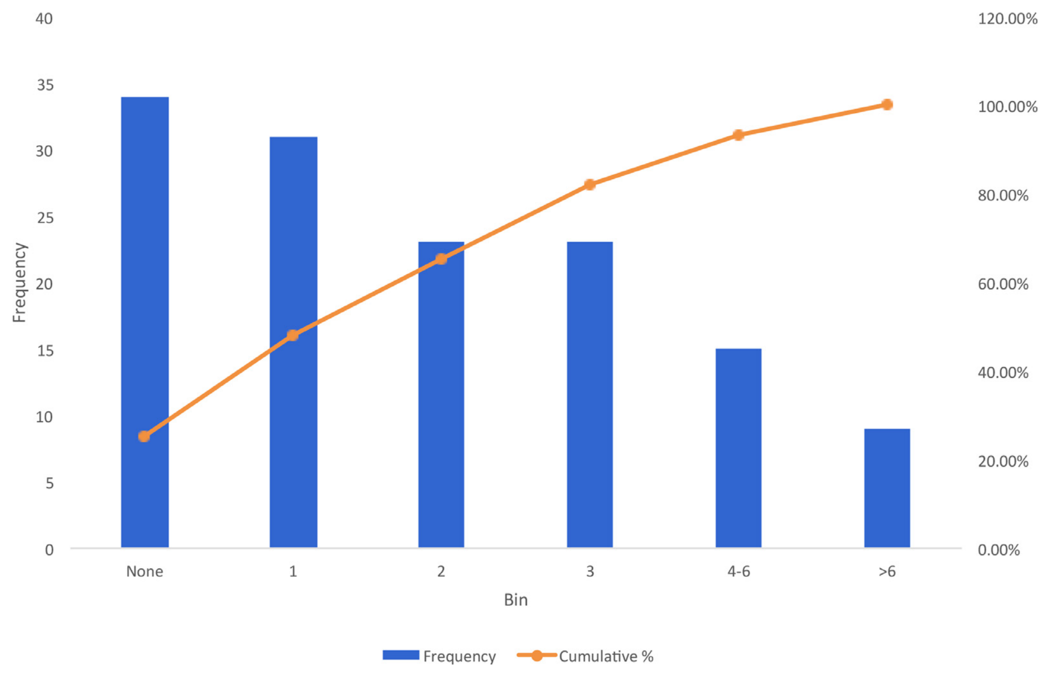

Fig. 13. Number of monitoring tools used across our sample.

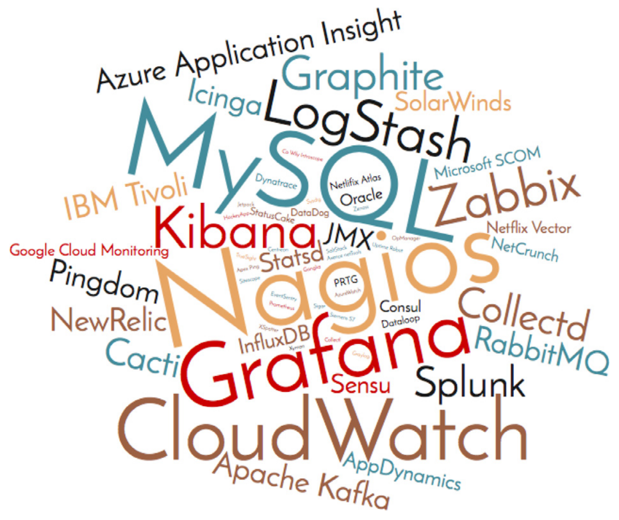

Fig. 14. Word cloud made with monitoring tools and their usage frequency. Google Analytics (i.e., the most frequent tool) was removed for picture clarity.

they manage, which leads to fewer incidents on average. Nonetheless, when incidents do occur, they experience longer unavailability times perceived by end users.

Moreover, 38% of the entire sample use no third party monitoring tools. They only use internally developed solutions, meaning that besides manual inspection, over a third of the companies in our dataset finds it more convenient to develop their own indoor custom monitoring and incident-tracking dashboard.

Among the companies using third party tools for monitoring purposes, the most used ones are: Google Analytics (18%), Nagios (16%), MySQL (14%), Grafana (12%), and Amazon CloudWatch (11%). In general, the monitoring tools offer is really fragmented among both commercial and open source solutions, with no clear winner, as can be seen from the word cloud generated according to the usage frequency in Fig. 14.

Finally, it is worth considering, that only 50% of all the surveyed companies release new features or new components always jointly with monitoring in mind, i.e., releasing new monitoring configurations or features specific for the new piece of the system.

Finding 2. Not all companies are used to monitor their software assets or have a strategy to do so; monitoring procedures and incident management organizational structures emerge organically often around custom-made ad-hoc monitoring solutions with little or no reference to the state of the art or practice.

4.2.3. RQ3 and RQ5: What are the people/roles involved with monitoring? What is the pandemic ratio for incidents?

Monitoring was once considered an operational endeavour [41]. However, our data shows that this trend is changed since in about 63% of the companies software developers access monitoring information and receive alerts. In about 16% of the cases also the business access monitoring information and, differently from our expectations, in 31.1% of the companies business directly receives alerts at the same time as operations and software development teams.

Whatsmore, we used the above figures and data to give an answer to RQ 5, namely, *What is the pandemic ratio for incidents? I.e., how many people are reached on average, if incidents do manifest?* To look for an answer, we correlated the average unavailability times with the sets of people and roles that access monitoring data.

Our data shows that the downtime perceived by end-users—that is, the people whose service is being interrupted—is inversely correlated with a Spearman correlation value of -0,37 ($p - value = 0.04$) to the number of roles that receive alerts immediately, and at the same time, meaning that downtime decreases the more roles are alerted at the same time. This confirms our conjectured existence of a **pandemic factor** that helps keeping downtimes low based on how many people are "involved" in incidents management and handling. However, we reported one single datapoint (or 0.6% of our dataset) where development, operations, and business personnel are all alerted by incidents - that same datapoint exhibits the lowest downtime perceived by end-users.

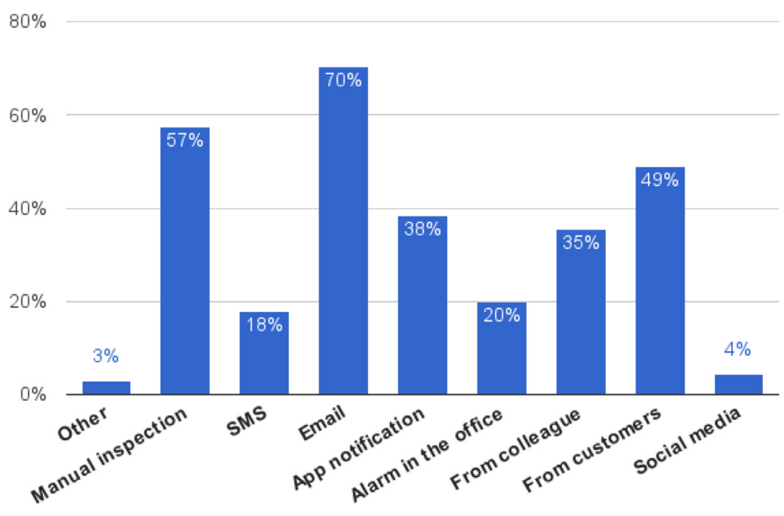

Fig. 15. Distribution of the alerting mechanism used by companies. Respondents could select multiple answers concerning incident reporting.

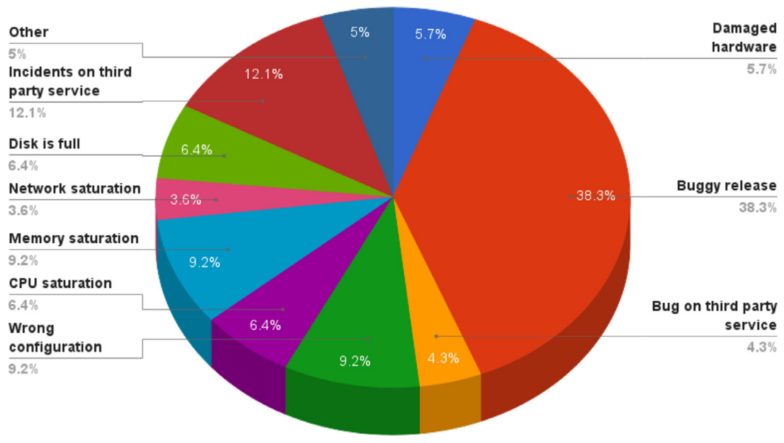

Fig. 16. Distribution of the most common cause for incidents according to interviewees.

Analyzing further in our data to elaborate more in the pandemic factor above, we observed that, according to our data, an average of 7 people receive 1 to 20 messages per day, with that average increasing linearly per size of the organization and with a maximum team-size of 14 people. Correlating the aforementioned average with team size, we observed that up to half the team can be notified and their activities disrupted whenever downtime is registered; this indicates a haphazard organizational structure around incident management. Also, our data reflects almost 20% of false-alarm rates - this leads to conclude that, on average, there exists an immature incident-management organisational structure.

Finding 3. application incidents reported through monitoring almost always involves only IT technical roles; however, when business roles are involved in the incident-management organizational structure, the incidents frequency and resolution times are hampered (*incident pandemic factor*).

4.2.4. RQ4: How are incidents discovered and handled?

In terms of how incidents are discovered and treated, 70% of surveyed companies use emails among the alerting techniques. 57.45% discover problems by actively inspecting logs or graphs. The third most frequent way of how employee are alerted is directly from end-users/customers. The complete distribution is depicted in Fig. 15. The release of buggy core components is considered the most common cause for an incident by 38.3% of the respondents. The distribution of the other causes is depicted in Fig. 16. The release of buggy core components requires manual intervention for 60.1% of the companies. For about half of the sample, manual intervention is also required for wrong configuration updates, filling of the disks, and hardware damage. Automatic remediation (or partial mitigation) is instead common for about 40% of the cases in the case of CPU or memory resource saturation.

Finding 4. 2 out of 3 incidents are found by direct manual log inspection and are reported via email. Heavy manual operations are involved in subsequent handling.

4.2.5. RQ6: What are the most critical challenges perceived when trying to make a system observable?

An explicit question in both our phase (1) interview campaign and phase (3) online survey asked participants information on the main obstacles they perceived as preventing the adoption of monitoring; as for other questions, the answer-set for this closed question was prepared using data from phase (1) of our study, but in the case of these questions we provided respondents with an open field to openly suggest an unforeseen answer. Fig. 17 shows the results. Almost 50% of the interviewees originally perceived the lack of standards as the main obstacle. This aspect, together with the proliferation of way too many tools for monitoring (34%), each one bringing new protocols and schemas, are undoubtedly the most critical challenges. Much in the same vein, challenges reported reflect learning curve and technical difficulty in usage (33%) - tools are hard to use - self-service and painless-setup solutions are preferred. Fur-

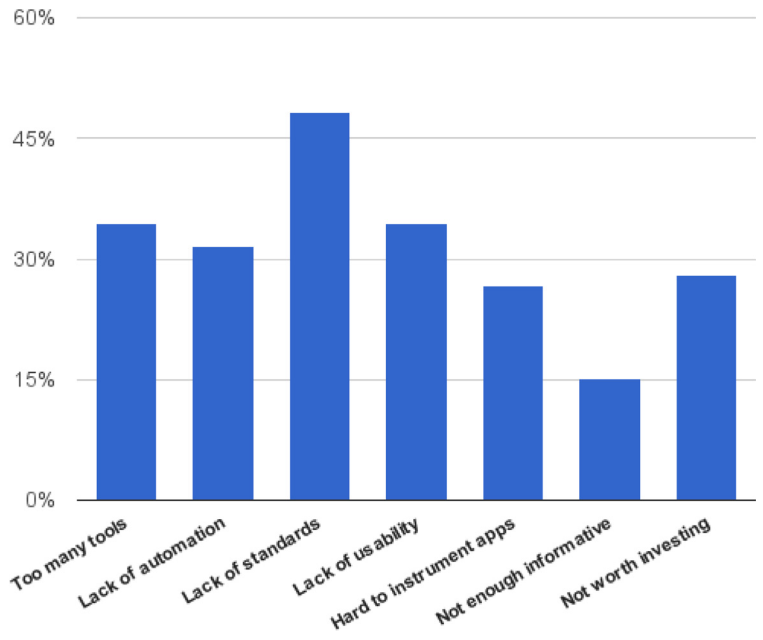

Fig. 17. Challenges perceived by interviewees in the adoption of monitoring.

thermore, organisational planning and IT strategies also play the role of the challenger - 29% of our respondents remark that monitoring is not currently perceived as a priority for internal infrastructure improvement investments. Lastly, a mere 2 respondents chose to suggest an alternative answer with the open-inquiry field.

Finding 5. Lack of (1) strategic monitoring business choices, (2) standardization, (3) faster learning-curves are perceived as the most critical challenges in systems monitoring and observability.

4.2.6. *RQ7: Is there correlation between complexity of cloud architectures and the time of unavailability reported for incidents?*

To address the above question we calculated the Spearman correlation coefficient between the measurement of cloud architecture complexity outlined in our research design (see Section 3.1 sub-question 7) with the average time of unavailability reported by our survey respondents. Our results show that the more complex and the more microservice-like the system is, then the less average unavailability is likely to be reported for that system. This finding seems to provide evidence of the benefits of microservice architectures [42] in the context of achieving large-scale complexity and availability at the same time. However, we report a little negative correlation around -0,28, which, even though statistically significant in terms of its p-value, may not reflect the commonly understood dimensions of software architecture complexity [43], mainly because of our particular definition of architecture complexity, devised specifically for the scope of this study.

Finding 6. complexity of cloud architectures, defined according to the SA_C metric (see Section 3), is (in a mild form) inversely proportional to unavailability times.

4.2.7. *RQ8: Is there any relation between systems observability and architecture complexity?*

To address the above question we calculated the Spearman correlation coefficient between our respondents' systems observability degree, as reported by respondents on a Likert-scale from no observability (equal to 0) to full observability (equal to 5), and SA_C outlined in our research design (see Section 3.1 sub-question 7). We report a mild, though still statistically significant, positive correlation of 0.16, meaning that for an increase in systems complexity an equal increase in systems observability also manifests in at least 15% of the times. On the one hand, the correlation indicates that people are aware of the “increasing architecture complexity ==> increased observability requirements” conundrum. On the other hand, the relatively low number could indicate the immaturity of architectures and systems observability [7] in this respect, e.g., in defining and operationalising systems observability as an architecture property and evaluate it in the context of cloud software architectures. Regardless of the previous tentative interpretation, the correlation does not warrant for any major observation about our target sub-research question, which, in the context of this study, remains un-answered.

Finding 7. there is limited to no perceivable relation between systems observability (as defined in literature [7]) and architecture complexity (as previously defined); the limited statistical significance and low correlation between the two scores, however, implies the need for further research in this field.

5. Threats to validity

Like any study of comparable magnitude and scale, this study is affected by several threats to validity [39]. In what follows we outline the major ones in our study design and execution.

5.1. *Internal and sampling validity*

Internal validity refers to the internal consistency and structural integrity of the empirical research design. More specifically it refers to how many confounding factors may have been overlooked. We attempted to address this threat with a sampling strategy where we controlled as many variables as possible to: (a) ensure a meaningful variety of our sample; (b) ensure that important variables for the objects of our study were controlled. For example, we controlled for organisational maturity, people age, gender, and more. Also, we controlled for process maturity, by selecting a heterogeneous sample according to a standard CMMI scale. Furthermore, there exist several angles to cloud applica-

Table 3
An industrial study of cloud applications monitoring — an overview of findings.

Finding	Description
1	Monitoring is perceived as a fundamental asset only for a specific profile of the industry and their subset constitutes around 15% of the total.
2	Not all companies are used to monitor their software assets or have a strategy to do so; monitoring procedures and incident management organizational structures emerge organically often around indoor ad-hoc monitoring solutions with little or no reference to the state of the art or practice.
3	Application incidents reported through monitoring almost always involve only IT technical roles; however, when business roles are involved in the incident-management organizational structure, the incidents frequency and resolution times are hampered (incident pandemic factor).
4	2 out of 3 incidents are found by direct manual log inspection and are reported via email. Heavy manual operations are involved in subsequent handling.
5	Lack of (1) strategic monitoring business choices, (2) standardization, (3) faster learning-curves are perceived as the most critical challenges in systems monitoring and observability.
6	Complexity of cloud architectures, in terms of the number of architecture components multiplied by the numbers of styles involved in such architectures, is (in some mild form) inversely proportional to unavailability times.
7	There is limited to no perceivable relation between systems observability (as defined in literature [7]) and architecture complexity (as previously defined); the limited statistical significance and low correlation between the two scores, however, implies the need for further research in this field.

tions monitoring that we could not consider, including for example the networked infrastructures monitoring angle in the scope of Software-Defined Networks (SDNs) domain area. From this perspective, we hope our study can become a reference work to drive further systematic analysis.

5.2. Construct and external validity

Although our measurements, observations, and findings are based on valid content (i.e., reported by practitioners who were directly involved with and witnesses to the reported subjects) and valid criteria, the external validity connected to the above-mentioned flaw may be compromised as well. For example, we used simple non-weighted and aggregate sums to evaluate the quantities involved in this study so we have no way of knowing whether the entity and *arity* of the involved quantities may yield different results. In this respect, we are planning further studies by automated quantitative means. Finally, our simplistic definition of architectural complexity was designed as a *proxy* for such complexity and is by no means of systematic inception and evaluation. We are aware that this definition would need further exploration and validation, but we feel this would fall out of the specific scope of this paper.

5.3. Conclusion validity

Conclusion validity represents the degree to which conclusions about the relationship among variables are reasonable. In the scope of the discussions of our results, we made sure to minimise possible interpretations, designing the study with reference to known hypotheses. Also, our conclusions were drawn from statistical analysis of our dataset. Finally, our conclusions, the dataset, the analyses were all disclosed for others to progress, replicate, or compound our results.

6. Discussion

Table 3 offers an overview of the findings we have identified in Section 4.2. From these, a rather obvious conclusion can be gathered: there is much untapped potential in the use of more precise and better instrumented monitoring practices for industrial cloud applications quality assurance.

6.1. Comparing our findings with the available literature: preliminary analysis

As a further step, we questioned whether there exist literary references, either grey or research literature, which address the points highlighted in the findings. In the following, we provide the preliminary results of our analysis.

First, with respect to findings 1 and 2, Gonzales et al. [44] highlight the exact same issues we reveal and offer an overview of *Continuous Monitoring*, a proposed methodology with supporting tools which reflects the adoption of a specific monitoring pipeline built as part of a DevOps pipeline and a specific strategy to use it during operations. The work provides an overview of the advantages, technical and business gains around the aforementioned adoption but also agree with several of the limitations highlighted in our finding 5 (e.g., lack of standardization) as the limiting factors hampering proper adoption of continuous monitoring in the field. Similar insights are provided by Chan et al. [45] who offer similar insights from an edge computing perspective.

Secondly, with respect to finding 3, while no research literature was found, several grey literature report on the urgency to devise more appropriate and effective ways to measure the quality of organizational structures involved in software engineering and cloud computing, especially around monitoring. Most prominently the article by Neray[11] offers several pointers for further research and practice in this direction.

Concerning finding 4, a white-paper from Instana[12], a software monitoring solutions vendor, highlights the risks and problems caused by manual monitoring in a continuous development/continuous integration context, where the presence of manual operations cause the whole pipeline to slow down.

Concerning finding 5, we were not able to find any literature to address either of the shortcomings emerged from our data analysis, other than whitepapers[13] or research literature [46] confirming the urgency and need to invest on the topic from both the academic and practical perspectives.

Finally, concerning finding 6 and 7, several recent works have touched upon cloud architecture complexity in terms of microservices proliferation as well as their governance and management. Most prominently by Toffetti et al [47] who offer experimental results on managing large-scale microservices solutions and Galletta et al. [48] who offer one such large-scale microservices management solution but in the very specific domain of oceanographic Big Data management and analysis. Furthermore, Tamburri et al. [7] offer an operationalisation of the concept of *Observability* and offer a refactoring exercise that shows how software maintainability improves and at which costs. Further research is

[11] https://searchitchannel.techtarget.com/tip/Cloud-demands-new-IT-organizational-structure-How-providers-can-help.

[12] https://www.instana.com/blog/continuous-deployment-and-continuous-monitoring-a-winning-pair/.

[13] https://www.infoq.com/articles/problem-with-cloud-computing-standardization.

still needed in this emerging research direction before the role of either microservices monitoring as well as observability becomes clear.

As mentioned above, our analysis is still preliminary and may have missed several other papers that explain or further elaborate on the findings we reported; a systematic literature review of methods and techniques for incident management would be needed and may reveal further insights into the organisational mechanisms and measures existing to sustain service continuity. This, however, is obviously beyond the scope of this paper and is the subject of our future work.

Summary of the literature preliminary analysis. From our preliminary analysis of the literature, the existing solutions are fragmentary and the field is rather immature; the present literature is not organic and there is no apparent continuity between the gray and research literatures. The works we reported upon confirm that monitoring platforms cannot be agnostic from the domain or from the business aspects around them and similarly there needs to be more research towards more effective and well-coordinated organizational structures around more suitable sets of tools. Finally, we report an evident barrier to the adoption of monitoring and incident management approaches and techniques that needs to be addressed. Our survey offers initial pointers in the direction, specifically, both towards a system that allows to define a better monitoring approach as well as towards standard monitoring systems and organizational structures [49].

6.2. Lessons learned

Beyond the literature-based discussion above, we report some lessons we learned from our findings.

Although the presence of standards is often frowned upon by researchers, academics, and practitioners themselves, *standards have a controlling power over the proliferation of differently-mature, diverse technologies*. In the scope of our findings, we revealed the presence of several technological “combo’s”, from the most basic ones (featuring MySQL for data storage and simple SQL querying mechanisms for monitoring data analysis) to de-facto standard patterns such as the ELK, ElasticSearch/Logstash/Kibana monitoring stack. In this proliferation, there exists a lack of harmonious effectiveness measurements, which leads software quality assurance researchers and practitioners to adopt an adhoc approach rather than standardised solutions to be improved over time with practice and experimentation. Perhaps the time has come for a standard to be proposed from industry and for industry, beyond previous proposals (e.g., the NIST proposed standardisation agenda over monitoring cloud applications[14]).

Beyond standards, *there exists no measurable quantity to evaluate to what degree cloud applications monitoring solutions are effective*, that is, there are no intuitive software architecture or code quality metrics that can be correlated with the number of incidents such that monitoring practice can be measurably improved. This also leads to a proliferation of rather simple-to-use and maintain solutions, without any definitive, disciplined approach emerging. Further software architecture, code quality, and maintainability research should look into this shortcoming for the benefit of both research and practice.

Finally, our results altogether show that *the human factor weighs heavily in monitoring effectiveness.* For example, we unwittingly uncovered a pandemic factor playing a role in reducing the amount of incidents reported with respect to architecture complexity, while investigating the impact of reported incidents across monitoring and quality management organisational structures. In the scope of this finding, we recon that very little research has been conducted so far in effective organisational structures for incident management, coverage, and disaster recovery. Further research in social software engineering should be invested in this direction, since the management practices we elicited from our results are basic and rather un-structured, deserving further investigation.

[14] https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.500-291r2.pdf.

7. Conclusion

The manuscript offers an industrial, mixed-methods study featuring interview, large-scale survey research. Our results shed light over several key research questions in the domain of cloud applications monitoring, spanning themes and topics such as (1) monitoring prominence in industry, (2) monitoring pandemics, (3) most critical challenges from a process, product, and lifecycle perspective.

Our results offer a glimpse of the untapped potential behind using more structured approaches for cloud applications monitoring and continuous quality improvement. In discussing our results, we conclude that the lack of technical and organisational standards were elicited as being the key limitations currently in the field.

In the future, we plan to refine a standards proposal to be addressed in the context of our participation within the OASIS Standardisation board such that industries at large may in fact benefit from our results and hopefully gain an active stance with respect to technological standardisation in the field.

Moreover, in the future we plan to further investigate the notion of effectiveness in cloud application monitoring organisational structures — this notion emerged as a novel contribution of our study, but we barely scratched its surface. Finally, we are planning to strengthen the validity of the contributions and results we outlined in this paper by executing a large-scale case-study campaign in industry, in hope of generalising our results by means of further observational, and exploratory research.

Declaration of Competing Interest

All authors declare not to have any financial or other cois of any kind.

CRediT authorship contribution statement

Damian A. Tamburri: Conceptualization, Data curation, Formal analysis, Funding acquisition, Investigation, Methodology, Project administration, Resources, Software, Supervision, Validation, Visualization, Writing - original draft, Writing - review & editing. **Marco Miglierina:** Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Software, Supervision, Validation, Visualization, Writing - original draft, Writing - review & editing. **Elisabetta Di Nitto:** Conceptualization, Funding acquisition, Methodology, Project administration, Resources, Software, Supervision, Validation, Writing - original draft, Writing - review & editing.

Acknowledgements

Some of the authors’ work is partially supported by the European Commission grant no. FP7-ICT-2011-8-318484 (MODAClouds), European Commission grant no. 0421 (Interreg ICT), Werkinzicht and the European Commissionrant no. 787061 (H2020), ANITA as well as the European Commissionrant no. 825040 (H2020), RADON and the European Commissionrant no. 825480 (H2020), SODALITE. We are very grateful to the domain experts who participated in our initial pilot analysis and to those who entusistically joined our survey and filled in our questionnaire.

References

[1] M. Conley, A. Vahdat, G. Porter, Achieving cost-efficient, data-intensive computing in the cloud., in: S. Ghandeharizadeh, S. Barahmand, M. Balazinska, M.J. Freedman (Eds.), SoCC, ACM, 2015, pp. 302–314.

[2] V. Prakash, Y. Wen, W. Shi, Tape cloud: scalable and cost efficient big data infrastructure for cloud computing., in: IEEE CLOUD, IEEE, 2013, pp. 541–548.
[3] L. Barolli, X. Chen, F. Xhafa, Advances on cloud services and cloud computing., Concurrency Comput. 27 (8) (2015) 1985–1987.
[4] L. Liu, Services computing: from cloud services, mobile services to internet of services., IEEE Trans. Serv. Comput. 9 (5) (2016) 661–663.
[5] P. Skvortsov, D. Hoppe, A. Tenschert, M. Gienger, Monitoring in the clouds: comparison of ECO2Clouds and excess monitoring approaches., CoRR (2016) abs/1601.07355.
[6] M. Zúñiga-Prieto, P. Cedillo, J. González-Huerta, E. Insfrán, S. Abrahão, Monitoring services quality in the cloud., ERCIM News 2014 (99) (2014).
[7] D.A. Tamburri, M.M. Bersani, R. Mirandola, G. Pea, Devops service observability by-design: experimenting with model-view-controller., in: K. Kritikos, P. Plebani, F.D. Paoli (Eds.), ESOCC, Lecture Notes in Computer Science, vol. 11116, Springer, 2018, pp. 49–64.
[8] J. Hernantes, G. Gallardo, N. Serrano, It infrastructure-monitoring tools., IEEE Softw. 32 (4) (2015) 88–93.
[9] P. Lipton, D. Palma, M. Rutkowski, D.A. Tamburri, Tosca solves big problems in the cloud and beyond!, IEEE Cloud Comput. 5 (2) (2018) 37–47.
[10] K. Alhamazani, R. Ranjan, K. Mitra, F.A. Rabhi, S.U. Khan, A. Guabtni, V. Bhatnagar, An overview of the commercial cloud monitoring tools: research dimensions, design issues, and state-of-the-art., CoRR (2013) abs/1312.6170.
[11] M.B. de Carvalho, R.P. Esteves, G. da Cunha Rodrigues, L.Z. Granville, L.M.R. Tarouco, A cloud monitoring framework for self-configured monitoring slices based on multiple tools., in: CNSM, IEEE Computer Society, 2013, pp. 180–184.
[12] S. Kvale, Doing Interviews (Sage Qualitative Research Kit), SAGE, 2008.
[13] A.M. Pettigrew, Longitudinal field research on change: theory and practice, Organ. Sci. Spec. Issue 1 (3) (1990) 267–292.
[14] H. Çam, Controllability and observability of risk and resilience in cyber-physical cloud systems., in: S. Jajodia, K. Kant, P. Samarati, A. Singhal, V. Swarup, C. Wang (Eds.), Secure Cloud Computing, Springer, 2014, pp. 325–343.
[15] T. Mauro, Adopting microservices at netflix: lessons for team and process design, 2015, ((accessed January 16, 2017) https://www.nginx.com/blog/adopting-microservices-at-netflix-lessons-for-team-and-process-design/).
[16] A. Bertolino, Software testing and/or software monitoring: differences and commonalities, Jornadas Sistedes (2014) 2093–2095.
[17] K. Fatema, V.C. Emeakaroha, P.D. Healy, J.P. Morrison, T. Lynn, A survey of cloud monitoring tools: taxonomy, capabilities and objectives, J. Parallel Distrib. Comput. 74 (10) (2014) 2918–2933, doi:10.1016/j.jpdc.2014.06.007.
[18] IBM, IBM Dictionary of Computing, 10th ed., McGraw-Hill, Inc., New York, NY, USA, 1993.
[19] G. Aceto, A. Botta, W. de Donato, A. Pescapè, Cloud monitoring: a survey., Comput. Netw. 57 (9) (2013) 2093–2115.
[20] C. So-in, C. Netflow, A survey of network traffic monitoring and analysis tools, (2020).
[21] I. Ghafir, V. Prenosil, J. Svoboda, M. Hammoudeh, A survey on network security monitoring systems, in: 2016 IEEE 4th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW), 2016, pp. 77–82, doi:10.1109/W-FiCloud.2016.30.
[22] R. Jabbari, N.B. Ali, K. Petersen, B. Tanveer, What is devops?: a systematic mapping study on definitions and practices, in: XP Workshops, ACM, 2016, p. 12.
[23] M. Kersten, A cambrian explosion of devops tools., IEEE Softw. 35 (2) (2018) 14–17.
[24] T. Mens, A state-of-the-art survey on software merging, IEEE Trans. Softw. Eng. 28 (5) (2002) 449–462.
[25] T. Mens, T. Tourwé, A survey of software refactoring, IEEE Trans. Softw. Eng. 30 (2) (2004) 126–162.
[26] J.S. Ward, A. Barker, A cloud computing survey: developments and future trends in infrastructure as a service computing., CoRR (2013) abs/1306.1394.
[27] R. Prodan, S. Ostermann, A survey and taxonomy of infrastructure as a service and web hosting cloud providers., in: GRID, IEEE Computer Society, 2009, pp. 17–25.
[28] , Model-Driven Development and Operation of Multi-Cloud Applications: The MODAClouds Approach, E. Di Nitto, P. Matthews, D. Petcu, A. Solberg (Eds.), Springer, Cham, 2017.
[29] R. Cimera, Asset Management, ADDitude, 2010.
[30] S.G. Linkman, Quantitative monitoring of software development by time-based and intercheckpoint monitoring., Softw. Eng. J. 5 (1) (1990) 43–49.
[31] D.A. Tamburri, P. Lago, H.v. Vliet, Organizational social structures for software engineering, ACM Comput. Surv. 46 (1) (2013) 3:1–3:35, doi:10.1145/2522968.2522971.
[32] L. Bass, P. Clements, R. Kazman, Software Architecture in Practice, SEI Series in Software Engineering, Addison-Wesley, 2003.
[33] L. Bass, P. Clements, R. Kazman, Software Architecture in Practice, Addison Wesley, 1998.
[34] P.C. Clements, R. Kazman, M. Klein, Evaluating Software Architectures, SEI Series in Software Engineering, Addison-Wesley, 2001.
[35] P. Biernacki, D. Waldorf, Snowball sampling. problems and techniques of chain referral sampling, Sociol. Methods Res. 10 (2) (1981) 141–163.
[36] C. Heath, J. Hindmarsh, P. Luff, Video Analysis and Qualitative Research, Sage Publications, London, 2010.
[37] S. Kvale, Doing Interviews (Sage Qualitative Research Kit), SAGE, 2008.
[38] K. Krippendorff, Content Analysis: An Introduction to Its Methodology (second edition), Sage Publications, 2004.
[39] C. Wohlin, P. Runeson, M. Höst, M.C. Ohlsson, B. Regnell, Experimentation in Software Engineering., Springer, 2012.
[40] E. Matouskova, K. Pavelka, Hyperspectral approach to a non-invasive survey of cultural heritage objects., in: WHISPERS, IEEE, 2013, pp. 1–4.
[41] K. Mahbub, Runtime Monitoring of Service Based Systems., City University London, UK, 2006 Ph.D. thesis. British Library, EThOS
[42] S. Newman, Building Microservices: Designing Fine-Grained Systems, O'Reilly, Beijing, 2015.
[43] R.S. Sangwan, L.-P. Lin, C.J. Neill, Structural complexity in architecture-centric software evolution., IEEE Comput. 41 (10) (2008) 96–99.
[44] G.C. Gonzalez, P.N. Sharma, D.F. Galletta, Factors influencing the planned adoption of continuous monitoring technology., J. Inf. Syst. 26 (2) (2012) 53–69.
[45] C. ling Chan, R. Fontugne, K. Cho, S. Goto, Monitoring TLS adoption using backbone and edge traffic, in: INFOCOM Workshops, IEEE, 2018, pp. 208–213.
[46] M. Miglierina, D.A. Tamburri, Towards omnia: a monitoring factory for quality-aware devops., in: W. Binder, V. Cortellessa, A. Koziolek, E. Smirni, M. Poess (Eds.), ICPE Companion, ACM, 2017, pp. 145–150.
[47] G. Toffetti, S. Brunner, M. Blöchlinger, F. Dudouet, A. Edmonds, An architecture for self-managing microservices., in: V.I. Munteanu, T.-F. Fortis (Eds.), AIMC@EuroSys, ACM, 2015, pp. 19–24.
[48] A. Galletta, L. Carnevale, A. Buzachis, A. Celesti, M. Villari, A microservices-based platform for efficiently managing oceanographic data., in: Innovate-Data, IEEE, 2018, pp. 25–29.
[49] M. Miglierina, Monitoring Modern Distributed Software Applications: Challenges And Soloutions, Politecnico di Milano, 2017 Ph.D. thesis. Under revision