Using ChatOps to Achieve Continuous Certification of Cloud Services

Paul Ohagen
HECTOR School of Management and Engineering,
paul.ohagen@icloud.com

Sebastian Lins
Karlsruhe Institute of Technology,
lins@kit.edu

Scott Thiebes
Karlsruhe Institute of Technology,
thiebes@kit.edu

Ali Sunyaev
Karlsruhe Institute of Technology,
sunyaev@kit.edu

Abstract

Continuous service certification (CSC) recently emerged as a promising means to provide ongoing assurances and disrupt pertinent certification approaches. CSC involves the consistent gathering and assessing of certification-relevant data by certification authorities about service operation to validate ongoing adherence to certification criteria. While research on CSC is increasing, practitioners still struggle in transferring researchers' suggestions and guidelines into practice. This study provides a tentative design and a prototype of a monitoring-based service certification (MSC) system based on the novel ChatOps approach. Iterative evaluations support our propositions that ChatOps' three key elements, a chat platform, chatbots, and third-party integrations, support the achievement of CSC. We contribute to research and practice by proving the technical feasibility of an MSC system, guiding future research and practitioners on achieving monitoring-based CSC, and validate the applicability and usefulness of extant guidelines on monitoring-based CSC proposed by prior research.

1. Introduction

Consumers now have ample access to an unforeseen variety of digital services to manage office and business tasks, track their health and fitness, make payments, and listen to music, among others. Continuous service certification (CSC) recently emerged as a promising means to provide consumers with ongoing assurances of important service properties, such as security or data protection [1, 2, 3]. More importantly, CSC innovates pertinent certification approaches in highly dynamic service environments (e.g., cloud computing). In its basic sense, CSC involves the consistent gathering and assessing of certification-relevant data by certification authorities about service operation to validate ongoing adherence to certification criteria. CSC utilizes innovative monitoring and auditing techniques to continuously assess services' adherence, such as autonomous, intelligent agents, interceptors, and automated log inspection techniques [1]. Despite its recent emergence, CSC already gained high importance, as can been seen by the funding of further CSC research projects (e.g., MEDINA funded by the EU), and the incorporation of CSC in novel certifications (e.g., ENISA's novel cloud certification scheme as a response to the EU Cybersecurity act) and large industry projects (e.g., the European ecosystem GAIA-X requires continuous monitoring of offered services)–mostly pertaining to cloud services.

Notwithstanding CSC's bright prospect and the growing attention of researchers, service providers, and policymakers, it remains underexplored and has been test-marketed and evaluated only prototypically in research projects [4, 5]. Service providers and certification authorities still struggle with implementing CSC due to CSC systems' high complexity and the demanding interplay between service providers and certification authorities [5]. In particular, service providers face the unresolved issue of providing certification authorities with comprehensive data about service operations on an ongoing basis.

Research on CSC has progressed in past years, leading to the foundation of diverse research streams, such as research focusing on process models (e.g., [1, 6]), architectures (e.g., [1, 4]), and techniques (e.g., [2, 3]) for achieving CSC. One promising and highly discussed CSC approach uses existing service monitoring data to assess ongoing adherence to certification criteria. By applying this monitoring-based service certification (MSC) approach, service providers extract and synthesize monitoring data about services that is already routinely gathered by themselves and then internally aggregate and provide certification-relevant data to certification authorities to enable them to perform ongoing data analyses [7]. In contrast to related CSC approaches (i.e., test-based CSC [2, 3]), MSC enhances the flexibility of data gathering for providers,

URI: https://hdl.handle.net/10125/80254
978-0-9981331-5-7

reduces the costs for setting up CSC, and does not require invasive service access from certification authorities, thereby reducing security risks.

Because MSC is promising, recent literature started to guide how to design MSC systems that gather, integrate, and process certification-relevant data internally, which is then provided to and analyzed by certification authorities (e.g., [7]). However, we still lack implementations of MSC systems that fulfill proposed design guidelines because implementing MSC systems is complex and challenging. MSC systems must fulfill various requirements, such as integrating data sets across diverse monitoring technologies and providing data effectively to certification authorities. This lack of insights into how to implement MSC systems hinders the application of CSC, ultimately preventing providers from proving the ongoing compliance of their services. To address this gap, we ask: *How to implement MSC systems?*

To answer this research question, we apply a lightweight design science research approach (building on Peffers et al. [8]), derive a tentative design, develop a system prototype, and evaluate its usefulness and fulfillment of prevalent MSC design guidelines in the context of cloud services. We particularly rely on the novel ChatOps approach concerned with integrating development and operations tools, and processes into a collaboration platform to enable teams to communicate efficiently and manage their workflow easily [9].

The developed prototype and iterative evaluations support our propositions that a ChatOps approach is a suitable means in achieving CSC due to its three key elements: a chat platform, chatbots, and third-party integrations. This study proves the technical feasibility of an MSC system by developing and evaluating a prototype based on a ChatOps approach, thereby complementing monitoring-based CSC research and answering research calls (e.g., [5]). By deriving a tentative design and describing components used in our prototype, we guide future (design-oriented) research and practitioners on achieving monitoring-based CSC. Our prototype also provides first validation regarding the applicability and usefulness of extant guidelines on monitoring-based CSC proposed by prior research, which was lacking so far.

2. Theoretical background

2.1 Continuous service certification

The conventional certification of digital services is a static attestation, primarily conducted by humans that only retrospectively reflects the technical and organizational requirements satisfied at the time of the assessment. For example, certification authorities perform on-site visits, interviews, or document analyses to assess a service's compliance with certification criteria (e.g., security and data protection regulations). The resulting certificate traditionally has a fixed validity period (e.g., one to three years) that requires certain stability of the certification object to assume that the attestation results remain constant over the entire validity period. However, this is not necessarily the case for current services, such as cloud services, due to fast service technology lifecycles, agile development, and continuous integration practices, threatening the reliability of issued certificates.

Researchers recently started to examine how to innovate certification processes to enable certification authorities to attest services continually. A resulting promising concept is CSC, where certification-relevant data about a service is consistently collected, aggregated, and processed to enable certification authorities to validate services' compliance with certification criteria continuously. In general, CSC builds continuous monitoring and auditing and combines these approaches with additional mechanisms for the transparent provision of certification-relevant information.

To achieve CSC, researchers typically propose two distinct CSC approaches: test-based and MSC, which are complementary because they can be used simultaneously to collect diverse evidence about certification adherence [7]. Test-based certification is characterized by the direct, external access of the certification authority to the service providers' infrastructure to check service components and operations [2]. Test-based CSC, therefore, follows a pull model, where the certification authority itself gathers certification-relevant evidence. These test checks performed by the certification authority typically involve controlling some input to the service and evaluating the response [6]. However, this approach is controversial as service providers are reluctant to give the certification authority access due to technical (e.g., requiring extensive modifications to the infrastructure), organizational (e.g., resistance to integrating untrustworthy techniques of authorities), or legal reasons (e.g., data protection laws) [1].

The second approach is called MSC and is a promising approach to overcome these drawbacks. MSC differs because the certification authority does not need direct access to the service provider's infrastructure. The service provider monitors its service infrastructure, collects data, and then makes the certification-relevant data available to the certification authority [7]. MSC, therefore, follows a push model, where service providers solely gather certification-relevant evidence inside the trusted service operation environment and then push this evidence to the certification authority. MSC entails greater flexibility to respond to

Table 1. Related research on CSC

		CSC approaches	
		Test-based CSC	Monitoring-based CSC
Focus of study	Concepts and guidelines	A Typical research question: *"How can test-based CSC be achieved?"* Example studies: [1, 3, 6]	B Typical research question: *"How to design monitoring-based CSC systems?"* Example studies: [7]
	Implementation	C Typical research question: *"How to test security compliance continuously?"* Example studies: [2, 10]	D Typical research question: *"How to implement monitoring-based CSC?"* Example studies: [4], **This study**

ever-changing service infrastructures because providers can independently alter their service infrastructure while ensuring that they still transmit certification-relevant data to certification authorities.

To achieve MSC, the service provider has to establish a sophisticated monitoring system that collects and aggregates certification-relevant data scattered across implemented monitoring software and provides this data in a way and format that suits the certification authority, which we refer to as the *MSC system.* However, service providers and certification authorities struggle to implement a suitable MSC system due to the high complexity and challenging interactions between both sides [5].

2.2 Related research on CSC

Reviewing the literature on CSC reveals that we still require a deeper understanding of designing and implementing MSC systems. Related research can be separated based on their chosen CSC approach (test-based vs. monitoring-based CSC) and their study focus (providing concepts and design guidelines vs. implementing CSC), among others (refer to Table 1).

Based on this separation, quadrant A summarizes most of the related work that focuses on achieving test-based CSC by providing process models, architectures, and frameworks to enable certification authorities to assess services continuously (e.g., [1, 3, 6]). For example, Anisetti et al. [3] propose a test-based security certification process that dynamically generates BPMN-compliant compositions of services that hold a set of security properties.

Similar, research in quadrant B focuses on providing the foundations and design guidelines for achieving monitoring-based CSC. For example, Lins et al. [7] derived meta-requirements and design guidelines for MSC systems based on findings from expert interviews. These research efforts are augmented by industry and government innovations, such as NIST's Open Security Controls Assessment Language offering machine-readable representations of certification criteria and metrics, among others.

Research also has started to implement and evaluate test-based CSC approaches, as summarized in quadrat C (e.g., [2, 10]). For example, test-based techniques verify consumers' data integrity [10] or continuous service availability [2].

Monitoring-based CSC has been less implemented (quadrant D). An exception is the early works of Krotsiani et al. [4], who developed a prototypical monitoring-based CSC infrastructure (called "*CUMULUS*") to, for instance, verify database user identification to validate certification criteria. While providing the first proof in concept, it remains unclear whether this prototype is generalizable to other service types and whether it fulfills recent design requirements and guidelines proposed by research in quadrant B. To this end, we aim to implement an MSC system to understand better how to perform CSC. We ground our research on the novel ChatOps approach to design and implement a prototype.

2.3 ChatOps

ChatOps, composed of the words chat and operations, integrates operations and development tools and processes into a collaborative communication environment like a chat tool [9]. ChatOps enables service providers' teams to use a unified interface to communicate efficiently, view relevant information, and easily manage their workflow.

ChatOps belongs to the emerging practice DevOps (development and operations) [11]. The DevOps approach is concerned with the fast and continuous development and delivery of new quality software releases and focuses on improving the collaboration between development and operations [12]. ChatOps is thereby a practice that can help organizations blur the lines between the roles of development and operations personnel [13].

From a technological perspective, the ChatOps approach has three key elements [14]. First, ChatOps embeds a chat platform that offers an instant messaging system to increase collaboration among users by providing a complete set of services for chatting and conversation through the internet. Key features of a chat platform relevant for ChatOps are team rooms and message persistence.

The second key element of ChatOps is third-party integrations to connect to other (external) services or platforms [9, 14]. These integrations are predefined connections into the chat platform, which extend the users' reach and make it easy for them to interact with other services or platforms. The interaction is commonly enabled through short (text) commands via direct interfaces from the messaging system.

The third key element of ChatOps are chatbots which provide customizable automation [14]. A chatbot is generally a running application, script, or piece of software that automates tasks usually performed by a human and can interact with human users on a chat platform. In ChatOps, chatbots become necessary when a third-party integration to the desired service is not available or does not provide the required functionalities that the team needs [9]. Compared to prominent B2C chatbots, the ChatOps chatbots are used to help teams manage their day-to-day work instead of facilitating interaction with consumers.

The introduction of ChatOps is a journey, including adjustments of organizational processes and implementation of software systems [15]. From an organizational perspective, ChatOps adoption is characterized by teams within organizations trying to move communication from email to group chat platforms [15]. The chat platform is used for sending messages or sharing files like logs and configuration files within dedicated rooms or channels. From a technical perspective, tools and services are connected to the chat platform to increase automation. These tools can automatically send notifications and information to the chat platform to make users aware of certain events or facts. Users also can query data from integrated tools by using slash commands. Chatbots are commonly added to the chat platform, interacting with people and tools and automating common tasks. These chatbots can be enhanced with artificial intelligence, which enables them to, for example, recommend solutions or channels where similar discussions took place, turning the chat platform completely into the operating system of teams.

Different kinds of ChatOps use cases are possible [14]. For example, ChatOps is frequently applied to enhance incident management by leveraging chatbots and third-party integrations, enabling monitoring of the services and infrastructure with notifications that alert subject matter experts or teams in case of disruptions or outages [16]. After detecting an incident, custom chatbots can start so-called war rooms within a channel of the chat platform, invite all relevant individuals and bring incident details from other platforms or services into the channel [14, 16]. Then, experts can collaboratively analyze the incident and issue commands to other platforms to isolate the incident and identify an effective response.

Reflecting ChatOps' key elements, we believe it is a valuable approach to building an MSC system prototype. First, it provides means to integrate different technologies into a chat platform, which can integrate various certification-relevant data sources (e.g., monitoring tools). Second, the chat platform can be used for the (mainly) automated data exchange between the provider and the certification authority. Besides, ChatOps fosters the communication inside the company to gather additional certification-relevant evidence, which cannot be collected in an automated manner and then provide it via chat platform's data sharing capabilities to other stakeholders. Finally, using chatbots and related scripts empowers service providers and certification authorities to define automated processes, such as data aggregation and filtering, or initiate related workflows (e.g., automatically responding to authorities' manual evidence requests, etc.). We, therefore, next describe our research approach to examine whether ChatOps is suitable for performing MSC.

3. Research method

In this study, we align our prototype development with the design science research (DSR) paradigm. In essence, DSR involves creating new knowledge through the design and evaluation of novel (IT) artifacts, along with reflection and abstraction to improve and understand the behavior of the artifact [17]. Given the increasing interest in DSR, there has been a continuous (and controversial) scientific discourse on what DSR is (e.g., [17]) and is not (e.g., [18]), how to conduct DSR studies (e.g., [8, 19]), and on recommendations and criteria for rigor, utility, and aesthetic (e.g., [20, 21, 22]), among others. This excess of advice and expectations for carrying out DSR also challenges researchers, making it difficult and costly to carry out DSR projects and leading to less research that applies DSR [22]. Therefore, in this study, we decided to opt for a more lightweight DSR approach, aligning with the DSR methodology proposed by Peffers et al. [8] and thus match with the DSR genre '*DSR methodology*' [22] that emphasizes the design and construction of applicable IT artifacts.

To ease readers' understanding and conform to prevalent DSR canons (e.g., [8, 19]), we divided our study into five phases: (1) problem awareness: understanding requirements for monitoring-based CSC, (2) suggestion: proposing a generic design fulfilling the requirements, (3) implementation: developing a prototype system, (4) evaluation: demonstrating the use of

and evaluating the artifact, and (5) drawing conclusions. Each phase tackles a critical sub-question of the overall DSR study, and we will therefore report our research steps taken in the following sections in more detail. Note that we rather applied an iterative DSR approach [17]. We went back and forth between these phases and performed ongoing evaluations of our interim findings, following the design-evaluate-construct-evaluate pattern [20].

We choose cloud services as an example research context to align with prior research on CSC, which mainly focuses on cloud services and because cloud services are highly dynamic. Hence, cloud consumers will greatly benefit from means enabling continuous assurance. A prototype implementation was selected to check whether a ChatOps approach can verify that cloud services conform to a defined set of certification criteria and elaborate on the ease of use, efficiency, and implementation effort.

4. Problem awareness: Requirements for monitoring-based CSC

We first examined the problem domain and reviewed extant research on CSC to understand the design problem and define objectives that our prototype should fulfill (activity 1&2 [8]). Since service providers still struggle to implement a suitable system for CSC due to the high complexity and demanding interplay with certification authorities [5], this work focused on a ChatOps-based implementation of an MSC system as a design artifact. We, therefore, aim to design and develop an IT system that builds on the key elements of ChatOps (i.e., chat platform, third-party integrations, and chatbots) to gather and transmit certification-relevant data in an automated manner. In contrast to traditional certifications' manual processes, (semi-)automated collection, analysis and transmission of certification-relevant data enable certification authorities to actively detect and investigate critical defects as they occur, ultimately increasing the reliability of certifications.

Literature on MSC already provides rich guidelines and descriptions on how to design MSC systems. This study aligns with the meta-requirements (MRQs) for CSC monitoring systems determined by Lins et al. [7], clustered into five categories in line with the layered client-server architecture pattern used by traditional monitoring system architectures. MRQs specify a class of goals that a design artifact should fulfill. We selected a subset of MRQs, including at least one MRQ from each category.

We align with two MRQs from the data-gathering layer that focus on the gathering of all certification-relevant data (i.e., refer to DGL1 [7]) by leveraging the cloud service provider's existing monitoring technologies to enable CSC (DGL2). Further MRQs were selected from the application layer and are concerned with enabling aggregation (AL1) and filtering (AL3) of the gathered data so that certification authorities can focus on the necessary amount of information in a consolidated form. In addition, our prototype should archive both the collected and processed monitoring data for certain periods to identify criteria deviations or conduct trend analyses (DL1, data layer). In addition, our MSC system should enable the continuous provision and transmission of certification-relevant information to the certification authority (IL1, interface layer) while ensuring data security during the exchange (IL2). Finally, we also adopt two non-functional MRQs, requesting that MSC systems achieve a high degree of automation (NF1) and adaptability (NF2) to be efficient, cost-effective, and increase the transparency of the CSC process. Taken together, these MRQs become the objectives that we want to achieve when designing and implementing our prototype. We acknowledge that the remaining MRQs proposed by Lins et al. [7] are highly relevant but relate to ensuring data protection, integrity, and auditability, which should be addressed once the technical feasibility of using a ChatOps approach has been proven.

Next, the scope of the MSC system needs to be determined. MSC generally concerns the continuous verification that cloud services comply with a set of certification criteria. Thus, an MSC system must specify which criteria can be automatically validated by gathering and providing corresponding data. Several criteria catalogs can be used to certify cloud services, but only a few consider continuous attestations. One promising exception is the cloud security attestation *'Cloud Computing Compliance Criteria Catalogue (C5)'*, which was developed by the German Federal Office for Information Security and combined several security standards and related criteria catalogs (e.g., ISO/IEC 27001). Given its international recognition and compatibility to continuous attestations, we selected a subset of criteria from the C5 attestation that our design artifact should verify.

The selected subset is composed of four certification criteria of the area secure service operations (criteria ID OPS-02, OPS-13, OPS-17, OPS-21, refer to the C5 criteria catalog for more details) and two certification criteria of the area security incident management (SIM-02, SIM-03). The subset of criteria was selected because these relate to logging (OPS-13), monitoring (OPS-02, OPS-17), and communication with relevant stakeholders (OPS-21, SIM-03) as core functionalities which could also be used for criteria of other areas in an adapted form, ultimately allowing us to

conclude the general use of ChatOps for MSC. Furthermore, criteria relating to secure service operations area were selected because ensuring service availability is one of the most frequently required criteria in cloud service certification (OPS-02, OPS-17). We further added SIM-02 and SIM-03 relating to incident management because it is one of the common use cases for ChatOps.

5. Suggestion: Prototype architecture

In the suggestion phase (activity 3 [8]), we formulate a tentative design for our MSC prototype (Figure 1). On the one hand, there is the live cloud system, and on the other hand, there is the MSC system building on ChatOps' key elements, both operated inside the trusted cloud infrastructure. The live cloud system includes IT resources and applications, and services offered by the cloud service provider.

The MSC system should leverage monitoring software running on the live cloud system, including IT infrastructure monitoring systems, monitoring tools, and plugins to gather relevant data (fulfilling (ful.) DGL1 and DGL2). Building on ChatOps' key element of third-party integration, available monitoring software should be connected to the MSC system. Third-party integration can be achieved by directly accessing offered monitoring APIs or applying an agent-based architecture model that comprises teams of intelligent software agents distributed to each cloud live system and respective monitoring software. Using such an agent-based architecture enables efficient integration of additional monitoring software, increasing MSC systems' adaptability (ful. NF2). The MSC system includes a database to store the gathered data from the cloud system, using flexible and adaptive data storage technologies (ful. DL1 and NF2).

Furthermore, the MSC system comprises data analysis, service-focused aggregation (e.g., aggregating data to summarize the operation of one service; ful. AL1), filtering (ful. AL3), and visualization functionalities (ful. IL1). On top of these data processing functions, the MSC system includes an alerting function that can automatically alert based on defined thresholds when data analysis has revealed deviations from expected behavior.

A chat platform is used as the interface for the MSC system (ful. IL1). This chat platform includes several chatbots that can access the MSC system's functionalities to achieve a high degree of automation (ful. NF1). Finally, an access control component is built into the chat platform to control that users, who could be the internal team, the certification authority, or cloud service customers, can access only the relevant information (ful. IL2). Data transmission should be encrypted to increase security and prevent sensitive data leakage (ful. IL2).

To continuously verify cloud service's adherence to the six selected certification criteria, we designed 16 different functionalities. These functionalities include the gathering and analyzing capacity (e.g., CPU, RAM, disk utilization) and availability metrics of cloud services and their underlying IT resources and stakeholders' information about the availability and the exceedance of certain thresholds (ful. OPS-02). Apart from the cloud services themselves, metrics related to the individual components of the monitoring and logging system need to be gathered, analyzed, and alerted in case of unreachability (ful. OPS-17). In addition, logs regarding the services and their underlying IT resources need to be gathered and analyzed to determine deviations from expected behavior and inform the relevant stakeholder about the irregular events (ful. OPS-13). However, the stakeholder should not only be automatically informed about incidents that affect them but should also be kept up to date on the status of the incident and informed about its resolution with the actions taken (ful. OPS-21). The last derived functionalities concern security as the prototype should enable the automatic identification and processing of security incidents (ful. SIM-02). Furthermore, the prototype should also allow for documentation of the processing and resolution of security incidents with the subsequent provision of the documented resolution to the affected customer (ful. SIM-03).

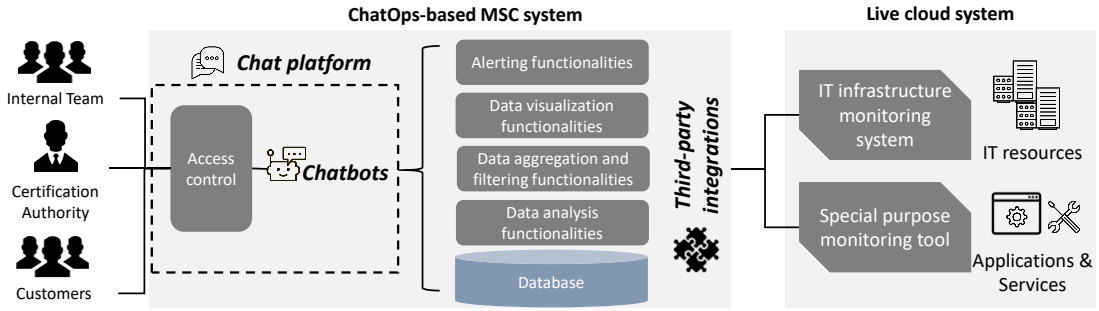

Figure 1. Abstract architecture for an MSC system with the ChatOps approach

6. Implementation: Prototype development

For developing the prototype, specific technologies for each component of the tentative design were required (activity 3&4 [8]; refer to Figure 2). To create a realistic scenario for the prototype, we decided to implement the prototype for a simulated cloud service provider that offers SaaS while using the infrastructure of another cloud service provider that offers PaaS. For the cloud service provider offering PaaS, Amazon Web Services (AWS) was chosen because AWS is one of the market leaders and is frequently used as underlying infrastructure. For the software services offered by the simulated cloud service provider, two different ways of deploying sample applications were used to verify that they are compatible with the ChatOps approach. These were Docker and Kubernetes, two of the most popular methods for deploying applications in cloud computing.

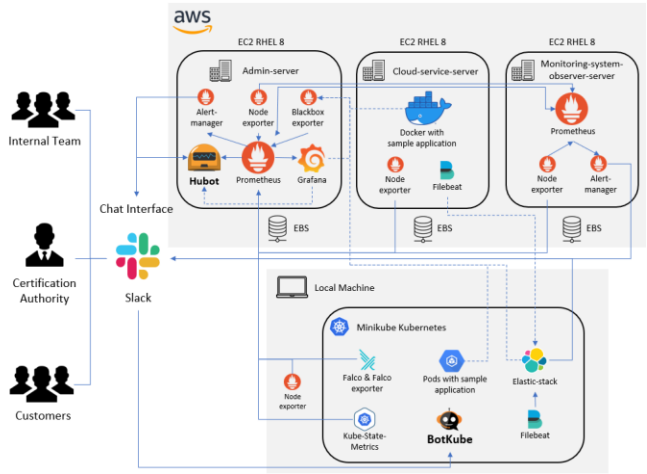

Figure 2. Prototype overview

Because the MRQs demand that existing monitoring technologies are leveraged to gather relevant data (i.e., DGL2), the selected monitoring tools for the prototype should be widely adopted to enhance generalizability. We, therefore, selected prominent and reputable monitoring systems, as reported by the end-user technology radar. We chose Prometheus as a monitoring and alerting tool, Grafana as a technology for creating observability dashboards, and the Elastic Stack, which deals with real-time analysis and visualization of log data.

For the chat platform used as the interface of the MSC system, Slack was chosen because it is one of the most popular chat platforms used for ChatOps [9]. Slack offers a variety of third-party integrations and hosted chatbots. For the prototypical implementation, the chatbots Hubot and BotKube were used. Hubot is currently one of the most well-known chatbots with the most extensive list of scripts for managing services and infrastructure [9]. The BotKube chatbot enables interaction with a Kubernetes cluster. These two chatbots were integrated by obtaining an access token based on the OAuth 2.0 authorization flow offered by Slack. Furthermore, the prototypical implementation relies on Slack's feature of protected data transmissions with the TLS 1.2 encryption protocol to ensure data security during the exchange.

The development of the prototype started with the setup of the IT resources within the AWS cloud computing platform. Therefore, three EC2 instances were launched. One of these instances was hosting a sample docker application and exposing it as a cloud service. The second instance was used to install the necessary monitoring tools (e.g., Prometheus, Grafana, Alertmanager, Blackbox exporter) and the Hubot chatbot. On the third instance, another Prometheus and Alertmanager instance was installed to observe the server with all monitoring tools to detect its unavailability. Apart from AWS resources, a Minikube Kubernetes cluster was installed on a local machine to vary the hardware. The Minikube Kubernetes cluster was used to set up a sample Kubernetes application, BotKube, the Elastic-stack, and the Falco-exporter required to identify security incidents in the cluster. The Prometheus node exporter was installed on all EC2 instances and the local machine to gather availability and capacity metrics (e.g., CPU, RAM, disk utilization).

Next, we set up Slack and configured five different channels based on the different topics handled by the objectives of the prototype. These topics included general alerts (e.g., availability and capacity alerts), log alerts, and security incidents for the internal team. Furthermore, channels for the certification authority and the customer were created to provide information to them. Slack's access control and access policy features provided all relevant stakeholders access to their required channels. Besides the used simple username and password authentication, Slack offers the setup of SAML SSO with providers like OneLogin or Okta,

After the initial installation and setup of the individual components, several components required further configuration to enable communication with other components and enable the desired functionalities. One of these components included Prometheus, which required the definition of both the objects that should be monitored and the alert rules, which specified scenarios with unexpected behavior that would require attention and an alert. The objects that should be monitored were specified by their respective URL and included all monitoring system components and the two example cloud services. Because the implemented cloud services do not expose Prometheus metrics, the

Blackbox exporter performed HTTP requests to check their availability. The defined alert rules are concerned with the capacity metrics, such as CPU, RAM, disk utilization, and the availability of the cloud services, their underlying infrastructure, and the components of the monitoring and logging system. For each alert rule, an expression was defined in PromQL syntax that specifies the condition to be fulfilled for a certain period to send an alert. Within the alert rule, a destination was specified to send alerts to different channels within Slack. The actual mapping of a destination name to a specific Slack channel is done by the Alertmanager, responsible for communicating the provisioned activated alert from Prometheus to Slack.

In addition, the ELK stack had to be configured, as the paths of the necessary log files to be collected had to be provided, and alert rules were created within Kibana to detect deviations in the logs. Furthermore, the previously prepared Slack channel regarding the logs was created as a connector in Kibana by providing its API URL to send the alerts to Slack.

The last two components that required further configuration were the BotKube and Hubot chatbots. Both of them were connected to Slack by providing a generated Slack API token to their configuration file. Once a connection was established, creating the actual automation scripts for specific tasks related to the objectives of the prototypical implementation began. Overall, three different automation scripts were created. These included providing information about the availability of the cloud services automatically and on manual requests and a script that helped the internal team inform and update the certification authority and the customers about security incidents.

7. Evaluation: Prototype assessments

Following an iterative design science approach and the design-evaluate-construct-evaluate pattern [20], we continuously evaluated the results of each phase (activity 5 [8]). For instance, we elaborated on whether our tentative design developed in the suggestion phase fulfills the MRQs; and we critically evaluated whether our prototype implementation aligns with the tentative design. Finally, we performed comprehensive functional and non-functional evaluations to analyze the technical feasibility of our MSC system and the suitability and usefulness of using a ChatOps approach.

Eight test cases were created for the functional evaluation to cover all the required functionalities derived from the objectives. One test case, for example, dealt with the shutdown of the sample application to simulate the unavailability of the cloud service. As a result, three alerts were automatically generated. The first one informed the internal team regarding the unavailability of the sample application (Figure 3). The other two alerts informed the certification authority and the customer after 15 minutes of continuous unavailability as it could, for example, be demanded in an SLA between the cloud service provider and the cloud service customer.

Figure 3. Unavailability alert

Another test case checked whether the certification authority is automatically informed about the availability of the cloud service every 24 hours. The result of the check was that every 24 hours, the certification authority was provided with a percentage of the average availability of the cloud service and the corresponding Grafana graph in a PDF file and the raw data in a JSON file. The certification authority can use this data for internal, automated analyses regarding certification compliance.

Overall, all eight test cases for the functional evaluation validated the expected behavior and can thus be considered successful. Hence, our prototype generally enables the verification of the selected certification criteria in an automated manner and provides the certification authority with transparent information.

Next, we assessed the prototype based on non-functional requirements as well. First, the prototype's ease of use was analyzed by discussing how end-users interact with the prototype. We perceived a high level of user-friendliness because users can directly interact with the chat platform Slack to interact with the chatbots, send commands or get in touch with other stakeholders. Chat platforms like Slack are now well-established and used by many people, which means that end-users do not have to get used to a new interface. Furthermore, end users can interact with the prototype in natural language by integrating the chatbot Hubot, which listens to requests to perform (complex) tasks.

Another criterion for evaluation is efficiency, relating to how efficiently the developed prototype supports the verification of the conformity of cloud services with a set of certification criteria. The prototype (semi-) automates various tasks needed for CSC. On the one hand, there are automatic gathering and analysis processes concerning certification-relevant information from the cloud services and their underlying IT

resources. Through the subsequent automated alerting and information provisioning, relevant stakeholders are informed about the incident or unexpected behavior without human intervention. On the other hand, there is still manual processing of incidents required. Nevertheless, these analyses are supported by automatic mechanisms, such as requesting additional analyses via chatbots on demand. Given the high degree of automation, service providers and certification authorities can save time and effort, so we conclude that the efficiency can be rated as high.

Finally, we assessed the effort required by cloud service providers and certification authorities to implement the ChatOps approach for performing MSC. The effort for the cloud service provider depends heavily on the extent to which it can already meet the certification criteria because it has already collected the certification-relevant information in the course of operating the cloud service. Furthermore, it must be considered whether the cloud service provider is familiar with the fundamental ChatOps approach. Generally, implementing the ChatOps approach for performing monitoring-based CSC represents a high initial effort for the cloud service provider, especially if the provider is new to ChatOps and has not yet collected much of the data relevant to certification. This is because the cloud service provider has to find and set up an appropriate tool or method for gathering certification-relevant data for each certification criterion, create alert rules to detect irregular events, and write scripts for the tasks that the chatbots will automate in operation. After the initial setup, however, the effort is minimized by the high degree of automation and adaptability as new exporters can be easily integrated into the solution. For the certification authority, the effort relates mainly to specify in which form and frequency the evidence for the criteria is required. Afterward, a certification authority can perform automated data analyses to attest the certification adherence.

8. Conclusion

Principal Findings. This study designed and developed a ChatOps-based prototype to clarify how to perform monitoring-based CSC since cloud service providers and certification authorities still struggle to transfer novel knowledge on CSC into practice.

Our iterative evaluations support our propositions that a ChatOps approach is a suitable means in achieving CSC. Our tentative design and the resulting prototype incorporate the three key elements of ChatOps to achieve our design goals and several advantages for service providers and authorities. Our prototype shows that a chat platform eases and automates the collaboration and communication between service providers and certification authorities by not only providing certification-relevant information in an automated fashion but also enable direct communication between employees. For example, a service administrator may directly comment and explain why certain criteria deviation appeared, for instance, due to service maintenance or false positives. Such direct feedback is highly valuable for certification authorities because they perform further (spot check) analyses on non-conformities to understand the reasons and rationales before deciding regarding certification suspension.

Implementing chatbots not only enables the automation of MSC functionalities, such as automated data gathering and aggregation but also provides certification authorities means to perform on-demand auditing. A ChatOps-based MSC system also enables certification authorities to develop their own chatbots that automatically analyze provided data. Certification authorities then do not have to set up their own CSC infrastructure but rely on the providers' chat platform.

Despite all the benefits ChatOps entails, there are a few things to consider to exploit ChatOps' potential fully, such as the signal-to-noise ratio, which is concerned about the ratio between meaningful insights and the potential overload of conversations and alerts, referred to as noise [9]. When the chat platform is constantly updated with new information that needs to be absorbed and processed, it can become hard to follow conversations and maintain awareness of what is going on, consequently doing more harm than being good to productivity. To find the right signal-to-noise ratio and avoid alert fatigue, the alert settings should constantly be adjusted such that all alerts are actionable and redundant alerts are reduced. A ChatOps-based approach also requires high efforts for initial setup. A novel MSC system has to be configured individually since each service infrastructure has its unique composition. Thus, in contrast to test-based CSC approaches that provide higher generalizability and reusability, deploying the same MSC system to novel services might be limited.

Implications for Research and Practice. Our study contributes to extant research on CSC and practice. First, we prove the technical feasibility of an MSC system by developing and evaluating a prototype, thereby complementing monitoring-based CSC research and answering research calls (e.g., [5]). More importantly, we clarify how to perform and implement monitoring-based CSC by building on the novel ChatOps approach. Our tentative design, prototype, and evaluation illustrate that ChatOps' key elements are suitable and useful means to support the process of CSC, such as automated data gathering and transparent information provisioning to certification authorities. With our tentative design, in particular, we guide

future (design-oriented) research on monitoring-based CSC. Our prototype also provides first validation regarding the applicability and usefulness of extant guidelines on monitoring-based CSC proposed by prior research, which was lacking so far.

For practitioners, this study's findings guide the implementation of MSC systems. Cloud service providers that already implemented the ChatOps approach may take our design and implementation recommendations to experiment with CSC. We also inform industry and policymakers currently demanding continuous monitoring and certification but lack the means to do so. For example, ENISA currently develops a candidate cybersecurity certification scheme for cloud services, requiring continuous monitoring of cloud services exposed to high risks. However, how to perform such monitoring is still an open issue and, indeed, a highly discussed topic in the (cloud) cybersecurity community.

Limitations and Future Research. Our study is not without limitations. First, we developed and evaluated a prototype in a test environment only by simulating two different cloud service applications running on different (virtualized) hardware. Future research may implement and evaluate an MSC system in real-world settings to better understand practical applicability and potential unintended side effects for the cloud service infrastructure. Second, we refrained from tackling MRQs regarding data security, integrity, and auditability for the first prototype version [7]. Thus, our prototype may be subject to security risks and, particularly, the risks of malicious providers euphemizing monitoring data (e.g., automatically deleting non-compliant data and reporting compliance only), and data communication vulnerabilities (refer to [1] and [7] for detailed security discussions). Third, all evaluations were solely performed by the researcher team and not with actual users of MSC systems. Future research should evaluate and discuss the prototype with cloud service and certification authority experts to identify integration and operation problems and potential boundary conditions of ChatOps-based MSC systems. Finally, we selected and tested only a small set of security-related certification criteria on an ongoing basis. Future research may analyze which certification criteria can be automatically validated using a monitoring-based CSC approach and whether combining it with a test-based CSC approach achieves greater coverage of suitable criteria.

9. References

[1] Lins, S., S. Schneider, and A. Sunyaev, "Trust is Good, Control is Better", IEEE Trans on Cloud Computing 6(3), 2018, pp. 890–903.

[2] Stephanow, P., and C. Banse, "Evaluating the Performance of Continuous Test-Based Cloud Service Certification", Proc. of the 17th CCGRID, (2017), 1117–1126.

[3] Anisetti, M., C. Ardagna, E. Damiani, and G. Polegri, "Test-Based Security Certification of Composite Services", ACM Trans on the Web 13(1), 2019, pp. 1–43.

[4] Krotsiani, M., G. Spanoudakis, and C. Kloukinas, "Monitoring-Based Certification of Cloud Service Security", Proc. of the OTM 2015 Conferences, (2015), 644–659.

[5] Teigeler, H., S. Lins, and A. Sunyaev, "Drivers vs. Inhibitors", Proc. of the 51 HICSS, (2018), 5676–5685.

[6] Kunz, I., and P. Stephanow, "A Process Model to Support Continuous Certification of Cloud Services", Proc. of the 31st AINA, IEEE (2017), 986–993.

[7] Lins, S., S. Schneider, J. Szefer, S. Ibraheem, and A. Sunyaev, "Designing Monitoring Systems for Continuous Certification of Cloud Services", CAIS 44, 2019, pp. 460–510.

[8] Peffers, K., T. Tuunanen, M.A. Rothenberger, and S. Chatterjee, "A Design Science Research Methodology for Information Systems Research", JMIS 24(3), 2007, pp. 45–77.

[9] Hand, J., ChatOps: Managing operations from Group Chat, O'Reilly Media, Sebastopol, CA, 2016.

[11] Wang, B., B. Li, and H. Li, "Oruta", IEEE Trans on Cloud Computing 2(1), 2014, pp. 43–56.

[11] Ebert, C., G. Gallardo, J. Hernantes, and N. Serrano, "DevOps", IEEE Software 33(3), 2016, pp. 94–100.

[13] Wettinger, J., V. Andrikopoulos, and F. Leymann, "Enabling DevOps Collaboration and Continuous Delivery Using Diverse Application Environments", Proc. of the OTM 2015 Conferences, 2015, 348–358.

[14] Calefato, F., and F. Lanubile, "A Hub-and-Spoke Model for Tool Integration in Distributed Development", Proc. of the 11th ICGSE, 2016, 129–133.

[15] Lane, R., and W. McKeon-White, Harness ChatOps To Empower Remote Collaboration, Forrester Research, Cambridge, USA, 2020.

[15] Regan, S., "What is ChatOps? A guide to its evolution and adoption", Work Life by Atlassian, 2016.

[16] IBM, IBM Cloud Service Management & Operations Field Guide, Armonk, NY, USA, 2018.

[17] Hevner, A.R., S.T. March, J. Park, and S. Ram, "Design Science in Information Systems Research", MIS Quarterly 28(1), 2004, pp. 75–105.

[19] Baskerville, R., "What design science is not", EJIS 17(5), 2008, pp. 441–443.

[20] Vaishnavi, V., and W. Kuechler, Design science research methods and patterns, CRC Press, Taylor & Francis Group, Boca Raton, 2015.

[21] Sonnenberg, C., and J. vom Brocke, "Evaluations in the Science of the Artificial", In K. Peffers, M. Rothenberger and B. Kuechler, eds., Design Science Research in Information Systems. Springer, 2012, 381–397.

[22] Baskerville, R.L., M. Kaul, and V.C. Storey, "Aesthetics in design science research", EJIS 27(2), 2018, pp. 140–153.

[23] Peffers, K., T. Tuunanen, and B. Niehaves, "Design science research genres", EJIS 27(2), 2018, pp. 129–139.