Baltic J. Modern Computing, Vol. 5 (2017), No. 1, pp. 124–135
http://dx.doi.org/10.22364/bjmc.2017.5.1.08

Systematic Literature Review of the Cloud-ready Software Architecture

Olesia POZDNIAKOVA, Dalius MAŽEIKA

Vilnius Gediminas Technical University, Saultekio al. 11, LT-10223 Vilnius

olesia.pozdniakova@vgtu.lt, dalius.mazeika@vgtu.lt

Abstract. Many companies are adopting cloud for hosting applications delivered as a service over the Internet. An application service delivered in this way is referred as Software as a Service (SaaS). The SaaS providers get benefits from getting cloud resources provisioned on-demand and through pay-as-you-go billing models. These capabilities enable faster application development and deployment with lower upfront investment into infrastructure. To benefit the most from the cloud, software architecture must be designed with consideration that software will run on the cloud. However, a legacy application is often developed by using monolithic architecture approach and might not get all advantages provided by a cloud computing. A growing monolithic application gets less flexible in development, has longer provisioning time, slows down the speed of innovation and lowers economy of scale. As result of such problems, not only commercial companies, but also the various academic researchers aim to design software as a "cloud-native" application. This kind of software has specific non-functional requirements that define scalability, reliability, fault tolerance and other cloud-specific requirements. This systematic literature review is based on the case studies, published articles and other literature related to the cloud-native or cloud-ready applications. It generalizes common traits of non-functional requirements defined in analyzed resources. Also, it walks through architectural styles of the applications running on a cloud and presents recently developed architectures that are used for large-scale software services delivered on a cloud.

Keywords: cloud-native; cloud-ready; cloud-aware; architecture; microservice; literature review

1 Introduction

Cloud computing essential characteristics, such as on-demand self-service, broad network access, resource pooling, rapid elasticity and measured service, made the cloud convenient for software systems deployments and software delivery as a service. Systems deployed on the cloud are often referred as Software as a Service (SaaS). To benefit the most from the cloud, software must be designed with the consideration that it will run on the cloud. Applications developed in such manner are commonly called

cloud-native or cloud-ready applications and sometimes cloud-aware. For the past few years a cloud-native term is industrialized by companies like VMware (see Web, a) or Pivotal (see Web, b) and is used to define containerized applications or applications developed using microservices. This term was used in Andrikopoulos et al. (2012) long time before microservices and Docker or similar solutions came up. Ambiguity of this term is also realized by Kratzke and Quint (2016), and Leymann et al. (2016). To avoid confusions in this study, "cloud-ready" term is to define application that are developed specifically to run on the cloud, non cloud-ready applications will be called "conventional". "Cloud-native" term is used only when referencing to a particular study that uses this term.

Wisely chosen software architecture helps to overcome potential problems and allows to take advantages provided by the cloud. For the cloud software architect or developer it is essential to understand what cloud-ready application is and what requirements it must meet.

This systematic literature review aims to summarize information available in studies related to cloud-ready applications architecture development by answering to the following research questions:

- *Question 1.* What is a cloud-ready application and how it differs from conventional applications?
- *Question 2.* What non-functional requirements are raised for cloud-ready applications?
- *Question 3.* What architectures are currently used for cloud-ready application?

Similar research questions were found in works created by Sodhi et al. (2011), Toffetti et al. (2016), Kratzke and Quint (2017). However, they address different aspects of cloud-ready software development.

Sodhi et al. (2011) discuss their proposed design approaches: cloud-aware and cloud-agnostic. These designs are used for development of application that will run on the cloud. They evaluate how application properties like security, portability, scalability and similar are affected by a choice of each of these approaches. Toffetti et al. (2016) work is more concentrated on cloud-native application development than on research questions raised in this study. Kratzke and Quint (2017) released an article where they exam an origin and meaning of "cloud-native application" (CNA) term together with CNA properties and engineering traits. The main difference between their study and ours is that Kratzke and Quint use only one specific term: CNA, while we apply a broader approach.

This study aims to provide more generic overview on cloud-ready software development.

2 Review methodology

This systematic literature review was conducted using guidelines defined by Kitchenham and Chaters (2007).

2.1 Search strategy

In order to answer the research questions, initial searches for primary studies were done using digital libraries, conference proceedings and other publicly available resources defined in Table 1. The search was limited by search engines on the Internet and access to digital libraries listed in Table 1. English language was selected for publication search. No time limits were applied to the publications release dates because research topic is relatively new. Further studies were identified by examining the reference lists of all included articles, searching relevant websites or looking through conference recordings.

Table 1. Data sources and search strategy.

Resourse	Total hits	Excluded	Duplication	Included	Search date
IEEE XPlore	71	63	0	8	2017-01-22
Elsevier ScienceDirect	82	78	0	4	2017-01-22
Springer Digital Library	67	63	0	4	2017-01-23
Thomson Reuters Web of Science	65	57	7	1	2017-01-23
Google Scholar	50 out of 2920	34	6	1	2017-01-22
References	–	–	–	8	2017-01
Internet	–	–	–	9	2017-01

The literature search used the following terms (with synonyms and closely related words): “cloud-ready” OR “cloud-native” OR “cloud-aware”. The search string was used to search in resource databases defined in Table 1. The total number of hits is shown in the same table. Queries for “cloud-native” or “cloud native” terms were providing the same results. Total amount of proposed articles was not high, for that reason it was decided not to refine the results by using other search terms like “application”, “architecture” and etc. This allowed us to minimize a possibility of missing the relevant studies. Querying Google Scholar provided about 2980 of various resources. Combining initial search string with terms like “application”, “software”, “architecture”, “design” and “style” or combination of those did not improve the situation. Amount of suggested studies was too high to evaluate all of them. As a result only first 50 resources were considered for analysis in this study.

2.2 Study selection

We have followed the study selection process proposed by Meline (2006) which includes the following steps:

- Step 1: Apply inclusion/exclusion criteria to titles and abstracts.
- Step 2: Eliminate studies that clearly meet one or more exclusion criteria.
- Step 3: Retrieve the full text of the remaining studies.
- Step 4: Evaluate the remaining studies for inclusion and exclusion.

Table 1 shows the number of papers excluded based on headings, abstracts, summaries analysis or due to access restrictions to the resources. We included studies that contain definitions of cloud-ready applications, define cloud-ready application architectures, development models, experience reports, non-functional requirements, characteristics, guidelines, principles.

The selected studies cover topics of cloud-ready application architecture, deployment, redesign of conventional applications to cloud-ready ones, migration of legacy applications to the cloud. However, studies related to the cloud platform, infrastructure, service development, security or other non-software engineering topics were excluded.

Various types of resources were analyzed as part of this review. More than half of the selected primary studies are peer-reviewed journal articles, conference proceedings or books. Selected Web resources were provided by widely known IT industry practitioners or companies. This variety of resources provides a holistic picture of current traits in a cloud software development area and helps to minimize a publication bias of this study. This is considered to be an indication of good quality study.

In order to provide answers to the research questions, the studies are organized in groups by relevance to the research questions. Table 2 shows which studies were considered to be relevant to a specific research question (RQ). Some of the studies might cover several or all of the questions.

Table 2. Studies to research questions mapping

Research question	Study
RQ1	Andrikopoulos et al. (2012), Brown and Capern (2014), Fehling et al. (2014), Inzinger et al. (2014), Kratzke and Peinl (2016), Kratzke and Quint (2017), Leymann et al. (2016), Ritter and Fehling (2013), Toffeti et al. (2016), Wilder (2012), Web b, a.
RQ2	Brunner et al. (2016), Fehling et al. (2011, 2014), Hole (2016), Kavis (2014), Kourtesis et al. (2012), Kratzke and Peinl (2016), Kratzke and Quint (2017), Leymann et al. (2016), Peinl et al. (2016), Retter and Fehling (2013), Rousev et al. (2016), Sodhi et al. (2011), Stine (2015), Toffetti et al. (2016), Weinman (2016), Wilder(2012), Zimmermann (2017), Garcia-Gomez et al. (2012), Casper at al. (2014).
RQ3	Andrikopoulos et al. (2013), Balalaie et al. (2016), Capelli and Scandurra (2016), Hole (2016), Inzinger (2014), Jambunathan and Kalpana (2016), Kourtesis et al. (2012), Kratzke and Peinl (2016), Kratzke and Quint (2017), Leymann et al. (2016), Linthicum (2016), Newman (2015), Peinl et al. (2016), Sill (2016), Toffetti et al.(2016), Vijaya and Neelanarayanan (2015), Weinman (2016), Wilder (2012), Ardagna et al. (2012), Lewis and Fowler (2016), Roberts (2016), Casper et al. (2014), Schaefer(2016), Meshenberg (2016).

These selected studies were analyzed and evaluated. Results of the analysis are presented in following subsections.

3 Analysis of studies

3.1 [RQ1] What is a cloud-ready application and how it differs from conventional applications?

As it was mentioned in Introduction section, multiple terms are used to define applications that are designed and developed to run specifically on the cloud. Here are just few examples of our findings. Andrikopoulos et al. (2012), Toffetti et al. (2016), Wilder (2012), Kratzke and Peinl (2016), Kratzke and Quint (2017) are using "cloud-native" term in their works and each of them provides a definition for it. "Cloud-ready" is mentioned by Brown and Capern (2014) from IBM, Kavis (2014), and Weinman (2016). "Cloud-aware" term is met in Open Datacenter Alliance report (Casper et al. (2014)) and Sodhi et al. (2011).

Andrikopoulos et al. (2012) define cloud-native applications as "... applications that are specifically designed and developed on top of a constellation of Cloud services, and which can fully exploit the characteristics of Cloud computing". He also defines Cloud-enabled software as software that was specifically adopted to be suitable for the cloud (Andrikopoulos et al. (2013)).

Toffetti et al. (2016) in their work make this definition even more universal: "Cloud-native application ... is an application that has been specifically designed to run in a cloud environment". What is similar to definition provided by Wilder (2012): "A cloud-native application is architected to take full advantage of cloud platforms."

Due to lack of common definition in academic literature, Kratzke and Peinl (2016) aim to define cloud-native application more explicitly: "A cloud-native application is a distributed, elastic and horizontal scalable system composed of (micro)services which isolates state in a minimum of statefull components. The application and each self-contained deployment unit of that application is designed according to cloud-focused design patterns and operated on a self-service elastic platform". In later published work Kratzke and Quint (2017) provides more detailed definition and explanation of the terms like "self-contained deployment unit" used in this definition.

Cloud-native definition provided in Kratzke and Peinl (2016) is more in alignment with current industry trends, like microservices and containers, which will be discussed in Subsection 3.3. Definitions provided in Andrikopoulos et al. (2012), Toffetti et al. (2016) and (2012) are more universal and technology agnostic. Authors of (Leymann et al. (2016)) argues, that term "cloud-native" should be applicable to applications developed using containers or microservices.

Even though there are many definitions of cloud-ready applications found in literature and authors do not agree on some implementation details (i.e. mandatory use of (micro)services and statefulness of components), there are a few common traits. First, a cloud-ready application is running on the cloud, which is a distributed system. And architects, designing applications that will run on the cloud compute, will face similar problems that are common for all distributed systems (even not cloud-ready, conventional, ones). Second, cloud-ready applications must be elastic. Herbst et al. (2013) provide definition of what elasticity in cloud computing means: "Elasticity is the degree to which a system is able to adapt to workload changes by provisioning and deprovisioning resources in an autonomic manner, such that at each point in time the available

resources match the current demand as closely as possible". Most of conventional applications were not design to run in environment that automatically scale as a number of transactions increases, not all conventional applications were designed to be elastic.

The Figure 1 provides a good illustration of how traditional applications development ties up cloud elasticity and economy of scale. Similar example was used by Retter and Fehling (2013) in their presentation from the 9th annual SATURN conference.

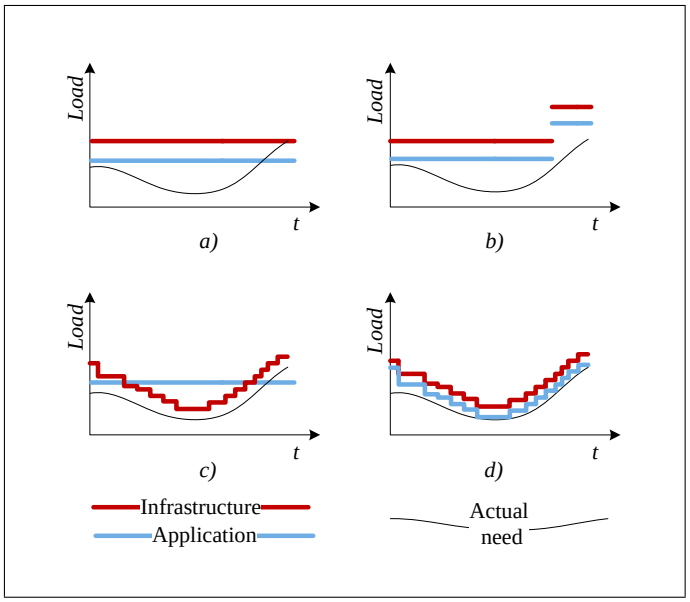

Fig. 1. Cloud platform and application elasticity levels of a conventional application running a) on a physical infrastructure, b) on a virtual infrastructure, c) on the cloud infrastructure, and d) a cloud-ready application running on the cloud infrastructure.

Other studies listed in Table 2 did not contain explicit definitions of the cloud-ready applications. Instead of definitions, these studies defined how applications should be developed to run on the cloud platform and what characteristics it should have. This overlaps with our second research question. Our findings in this area will be discussed in subsection 3.2.

Open Data Center Alliance team (see Casper et al., 2014) proposes the way to assess the level of cloud maturity of an application. It uses 3 maturity levels. Level 0 evaluates application capability to run on virtualized platform. Level 1 is given to loosely coupled applications. Applications that can run on any cloud platform, are stateless and robust to failures of dependent services have second level of maturity. The highest level of maturity is achieved when an application can elastically scale in respond to the load and can be migrated between cloud service providers without interruption.

Automation and elasticity can be treated as essential characteristics of a cloud-ready application (and which distinguish it from conventional applications), however these are not the only properties that makes an application cloud-ready.

3.2 [RQ2] What are non-functional requirements raised for cloud-ready applications?

An overview of existing cloud-ready applications definitions is provided in subsection 3.1. In this section we discuss what non-functional requirements are raised for cloud-ready applications. It turned out, that different non-functional requirements and their quantity are raised in various papers. The most common ones are scalability, elasticity, automated deployment, vendor lock-in avoidance (Brunner et al. (2016), Sodhi et al. (2011), Toffeti et al. (2016), Inzinger (2014), Kavis (2014), Carcia-Gomez et al. (2012), Fehling et al. (2014) and others). Among other requirements were loose coupling, statelessness, fault-tolerance. Even though, analyzed studies were raising a need for vendor lock-in avoidance and unobstructed migration between cloud providers, this was not reflected in any of cloud-ready application explicit definitions that were discovered and previously presented in subsection 3.1. It is worth mentioning that these requirements were commonly provided in a form of guidelines or principles for the cloud-ready applications development.

A lot of work in cloud-ready applications development area is done by team from Institute of Architecture of Application Systems (IAAS) of University of Stuttgart. Andrikopoulos et al. (2012) evaluate the effect of design decisions on the consistency, availability and partitioning (CAP) properties of cloud-ready applications. Fehhling et al. (2011, 2014) works discuss various cloud computing application patterns, challenges of deploying these patterns in the cloud and proposes possible solutions to overcome these challenges. They also define IDEAL properties for a cloud-native application. IDEAL is an acronym, where "I" stands for an isolated state, "D" is for distributed, "E" – elastic, "A" – automated, "L" – loosely coupled. Figure 2 summarizes cloud-ready application characteristics defined in the above mentions IAAS team studies.

Wilder (2012) defines eleven properties of cloud-native applications like loose-coupling, horizontal and automatic scale, fault-tolerance, resiliency (upgrades, migration and faults should happen without downtime). He proposes 13 patterns that enable these properties in cloud-ready applications.

O'Reilly's report written by Stine (2015) provides a set of characteristics that cloud-native applications should have, such as fault-tolerance, state and fault isolations, horizontal scalability, automatic recovery, statelessness.

Kratzke and Peinl (2016) in their cloud-native application reference model for enterprise architects, called ClouNS, raise an importance of vendor lock-in avoidance in cloud-ready application design. A systematic mapping study prepared by Kratzke and Qiuin (2017) provides a cloud-native application principles, like need of automation platforms, software defined infrastructure, migration and interoperability between the clouds.

The conducted review of RQ2 studies listed in Table 2 showed out, that the basic cloud-ready application design principles or properties, such as horizontal scalability, loose coupling, isolation of state, need for elasticity, distribution and automation haven't

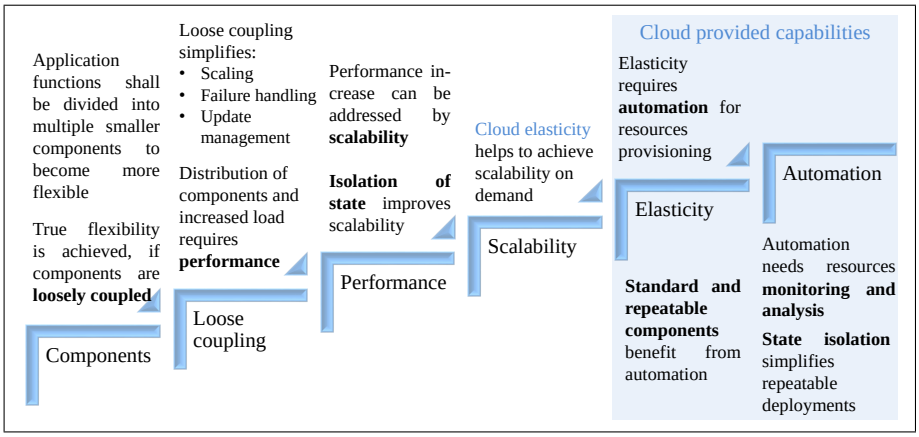

Fig. 2. Cloud-ready application characteristics.

changed since defined by Fehling et al. in 2011 or Wilder in 2012. However, since then cloud computing technologies has evolved and new architectural styles were developed. In next section we will walk through architectural styles and recently developed architectures for cloud-ready applications.

3.3 [RQ3] What architectures are currently used for cloud-ready application?

It turned out that there were no articles released till 2016 which would explicitly define cloud-ready application architecture. However there were multiple articles that were discussing architectural approaches for development of cloud-ready applications (Sodhi et al. (2011)), principles (Andrikopoulos et al. (2012)), patterns (Wilder (2012), Fehling et al. (2011, 2014)). Later works in this area were more related to conventional application migration to the cloud (Andrikopoulos et al. (2013)) or development methods or frameworks (Inzinger et al. (2014), Ardagna et al. (2012)).

Major part of selected studies that are relevant to cloud-ready application architecture were released in 2016. Most of these articles are mentioning microservice architecture as architecture for cloud-ready applications. Solutions like Docker or Rocket containers together with Docker Swarm, Mesos or Kubernetes automated container management currently act as enablers of this architectural style (Peinl et al. (2016)).

The microservice architectural style was proposed in 2014. It is relatively new and, as result, various interpretations of what this architectural style means are met in literature (Balalaie et al. (2016), Lewis and Fowler (2016), Nadareishvili et al. (2016), Linthicum (2016)). A lot of work in microservice architecture area is done by ThoughtWorks team. Members of this team, Lewis and Fowler, give the following definition of this style: "...the microservice architectural style is an approach to developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API. These services are built around business capabilities and independently deployable by fully automated

deployment machinery. There is a bare minimum of centralized management of these services, which may be written in different programming languages and use different data storage technologies".

This architectural style is commonly compared to Service Oriented Architecture (SOA) (Sill (2016), Leymann et al. (2016)), as some of SOA key principles like loose coupling, autonomy, reusability, composability, statelessness are common with the microservice architecture (MSA) (Erl (2005), Newman (2015)). However there are several key deferences. SOA is concentrating on enterprise needs and services, while MSA concentrates on development of one application. MSA components can be treated as a smaller scale service, that serves a specific function that is required by application.

The services in SOA are governed by these (but not only by these) key principles: abstraction and adherence to a service contract (Erl (2005)). According to Newman (2015), microservice development principles are against abstraction: it should be concrete and perform a specific function. Also there are no limitations in choosing a communication methods between the microservices, an application development team needs to agree on what method to use.

The analyzed articles (Balalaie et al. (2016), Jambunathan and Kalpana (2016), Toffeti et al. (2016), Brunner et al. (2016), Kratzke and Quint (2017) and etc.) and practitioners use cases (see Schaefer (2016), Meshenberg (2016)) showed that commonly MSA is proposed as an alternative to monolithic application architecture. MSA solves a set of problems like fault-tolerance, speed-of-development, horizontal scalability, but also brings operational and organizational complexity. The development and operational complexity of the microservice architecture makes it more suitable for large scale application deployments or for delivery of Software as a Service (Leymann et al. (2016)).

Microservices can be scaled if and only if their associated load requires it (Stine (2015)), as result, microservice architecture enables more granular application elasticity, similar to one presented in Figure 1 d), which would be more difficult to achieve running an application on virtual machines.

Also, another approach for cloud application development that was found, is "serverless" (Weinman (2016)). "Serverless" architecture application is developed as a custom code that runs in ephemeral containers (see Roberts (2016)) and performs a specific functions. Amazon uses term "serverless" to define applications developed using this approach, as customer does not need to procure virtual machines instances to run a code. The applications are invoked by various triggers to serve a specific function.

Even though this approach has obvious drawbacks, "serverless" solution that runs on public cloud might be more cost effective solution than running containers or virtual machines.

4 Summary

We have discussed what a cloud-ready application is, how it differs from conventional applications and what architectural approaches could be used to develop applications for the cloud. There are several terms used to define such applications. The most commonly used are "cloud-native" or "cloud-ready". "Cloud-native" term became ambiguous in

past years. Even though it is used to define applications that developed to run on the cloud, it also defines specific way of development, which is based on containers or microservices.

A cloud-ready application is developed as distributed system that uses loosely coupled components, is designed to be horizontally scalable and run on an automated and elastic platform. Ideally, it should be possible to migrate these applications between various cloud platforms without service interruptions. The highest cloud-ready application maturity level is achieved, when all of mentioned above characteristics are met. Elasticity and automation are essential cloud-ready application characteristics which make it different from conventional applications.

In 2014 the microservice architectural style was proposed. The microservice architecture and development principles like loose coupling, autonomy, vertical scalability, isolation of state and use of automated platform enables development of mature cloud-ready applications. No wonder that this architectural style is often followed by "cloud-native application" term. Another application development approach that currently more common for applications running on public or hybrid cloud (but also can be used on private cloud) is "serverless". This approach uses function-as-a-service services, which allow to execute applications only when those need to perform a specific function with a short lifespan. This allows to optimize usage of infrastructure resources and increase elasticity, together with economy of scale. Microservice and the cloud-ready application design principles are applicable to the "serveless" architectural approach. Microservice characteristics and operations complexity make it more attractive for development of large scale applications and SaaS. "Serverless" seems more universal, as it can be incorporated with conventional applications to fulfill any specific function, that previously required procurement of virtual machine instance. Several analyzed studies showed that microservice and "serverless" architectural approaches are not mandatory for a cloud-ready application, however they enable development of cloud-ready applications with higher maturity level.

References

Andrikopoulos, V., Binz, T., Leymann, F., and Strauch, S. (2013). How to adapt applications for the Cloud environment: Challenges and solutions in migrating applications to the Cloud. *Computing*, 95(6):493–535.

Andrikopoulos, V., Strauch, S., Fehling, C., and Leymann, F. (2012). CAP-Oriented Design for Cloud-Native Applications. In *Proceedings of the 2nd International Conference on Cloud Computing and Service Science, CLOSER 2012, 18-21 April 2012, Porto, Portugal*, pages 365–374. SciTePress.

Ardagna, D. et al., (2012). MODAClouds: A model-driven approach for the design and execution of applications on multiple clouds. In *Proceedings – 2012 4th International Workshop on Modeling in Software Engineering, MiSE 2012*, pages 50–56.

Balalaie, A., Heydarnoori, A., and Jamshidi, P. (2016). Microservices Architecture Enables DevOps: Migration to a Cloud-Native Architecture. *IEEE Software*, 33(3):42–52.

Brown, K. and Capern, M. (2014). Top 9 rules for cloud applications. *IBM Middleware Technical Journal for Developers*, (17.2).

Brunner, S., Blochlinger, M., Toffetti, G., Spillner, J., and Bohnert, T. M. (2016). Experimental Evaluation of the Cloud-Native Application Design. In *Proceedings – 2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing, UCC 2015*, pages 488–493.

Capelli, S. and Scandurra, P. (2016). A framework for early design and prototyping of service-oriented applications with design patterns. *Computer Languages, Systems and Structures*, 46:140–166.

Casper, D., Bette, C., and Louie, K. (2014). Best Practices: Architecting Cloud-Aware Applications. Available at: `https://www.opendatacenteralliance.org//docs/architecting_cloud_aware_applications.pdf` [Accessed January 25, 2017].

Erl, T. (2005). *Service-oriented architecture: concepts, technology, and design*. Prentice Hall Professional Technical Reference.

Fehling, C., Leymann, F., Mietzner, R., and Schupeck, W. (2011). Universität Stuttgart A Collection of Patterns for Cloud Types, Cloud Service Models, and Cloud-based Application Architectures Institute of Architecture of Application Systems Daimler AG. Technical report, Institute of Architecture of Application Systems.

Fehling, C., Leymann, F., Retter, R., Schupeck, W., and Arbitter, P. (2014). *Cloud Computing Patterns*. Springer Vienna, Vienna.

Garcia-Gomez, S. et al., 2012. *4CaaSt: Comprehensive Management of Cloud Services through a PaaS. 2012 IEEE 10th International Symposium on Parallel and Distributed Processing with Applications*, pages 494–499.

Herbst, N. R., Kounev, S., and Reussner, R. (2013). Elasticity in Cloud Computing: What It Is, and What It Is Not. In *Presented as part of the 10th International Conference on Autonomic Computing*, pages 23–27, San Jose, CA. USENIX.

Hole, K. J. (2016). Toward an Anti-fragile e-Government System. In *Simula SpringerBriefs on Computing*, volume 1, pages 57–65. Springer International Publishing, Cham.

Inzinger, C., Nastic, S., Sehic, S., Vogler, M., Li, F., and Dustdar, S. (2014). MADCAT: A methodology for architecture and deployment of cloud application topologies. In *Proceedings – IEEE 8th International Symposium on Service Oriented System Engineering, SOSE 2014*, pages 13–22.

Jambunathan, B. and Kalpana, Y. (2016). Multi Cloud Deployment with Containers. *International Journal of Engineering and Technology*, 8(1):421–428.

Kavis, M. J. (2014). *Architecting The Cloud.* John Wiley & Sons, Inc., Hoboken, NJ, USA.

Kitchenham, B. (2007). Performing Systematic Literature Reviews in Software Engineering. Technical report.

Kourtesis, D., Bratanis, K., Bibikas, D., and Paraskakis, I. (2012). Software co-development in the era of cloud application platforms and ecosystems: The case of CAST. In *IFIP Advances in Information and Communication Technology*, volume 380 AICT, pages 196–204.

Kratzke, N. and Peinl, R. (2016). ClouNS-a Cloud-Native Application Reference Model for Enterprise Architects. In *Proceedings – IEEE International Enterprise Distributed Object Computing Workshop, EDOCW*, volume 2016-Septe, pages 198–207. IEEE.

Kratzke, N. and Quint, P.-C. (2017). Understanding cloud-native applications after 10 years of cloud computing – A systematic mapping study. *Journal of Systems and Software*, 126:1–16.

Lewis, J. and Fowler, M. (2016). *Microservices Guide*. Available at: `http://martinfowler.com/microservices/#what` [Accessed November 27, 2016].

Leymann, F., Fehling, C., Wagner, S., and Wettinger, J. (2016). Native Cloud Applications: Why Virtual Machines, Images and Containers Miss the Point ! In *Proceedings of the 6th International Conference on Cloud Computing and Service Science (CLOSER 2016)*, pages 7–15. SciTePress.

Linthicum, D. S. (2016). Practical Use of Microservices in Moving Workloads to the Cloud. *IEEE Cloud Computing*, 3(5):6–9.

Meline, T. (2006). Selecting Studies for Systematic Review: Inclusion and Exclusion Criteria. *Contemporary Issues in Communication Science and Disorders*, 33:21–27.

Meshenberg, R. (2016). GOTO 2016. Microservices at Netflix Scale: Principles, Tradeoffs & Lessons Learned. Available at: `https://www.youtube.com/watch?v=57UK46qfBLY` [Accessed January 29, 2017].

Nadareishvili, I. et al., 2016. *Microservice architecture: Aligning principles, practices, and culture* O'Reilly Media, Inc.

Newman, S. (2015). *Building Microservices*. O'Reilly Media, Inc, 1st edition.

Peinl, R., Holzschuher, F., and Pfitzer, F. (2016). Docker Cluster Management for the Cloud – Survey Results and Own Solution. *Journal of Grid Computing*, 14(2):265–282.

Retter, R. and Fehling, C. (2013). Applying Architectural Patterns for the Cloud: Lessons Learned During Pattern Mining and Application.

Roberts, M. (2016). Serverless Architectures. Available at: `http://martinfowler.com/articles/serverless.html` [Accessed January 29, 2017].

Roussev, V., Ahmed, I., Barreto, A., McCulley, S., and Shanmughan, V. (2016). Cloud forensics–Tool development studies & future outlook. *Digital Investigation*, 18:79–95.

Schaefer, R. (2016). GOTO 2016. From Monolith to Microservices at Zalando. Available at: `https://www.youtube.com/watch?v=gEeHZwjwehs` [Accessed January 30, 2017].

Sill, A. (2016). The Design and Architecture of Microservices. *IEEE Cloud Computing*, 3(5):76–80.

Sodhi, B. and Prabhakar, T. V. (2011). Application architecture considerations for cloud platforms. In *2011 Third International Conference on Communication Systems and Networks (COMSNETS 2011)*, pages 1–4. IEEE.

Stine, M. (2015). Migrating to Cloud-Native Application Architectures. Technical report, Sebastopol.

Toffetti, G., Brunner, S., Blöchlinger, M., Spillner, J., and Bohnert, T. M. (2016). Self-managing cloud-native applications: Design, implementation, and experience. *Future Generation Computer Systems*.

Vijaya, A. and Neelanarayanan, V. (2015). Framework for platform agnostic enterprise application development supporting multiple clouds. In *Procedia Computer Science*, volume 50, pages 73–80.

Weinman, J. (2016). Migrating to - or away from - the Public Cloud. *IEEE Cloud Computing*, 3(2):6–10.

Wilder, B. (2012). *Cloud Architeture Patterns*. O'Reilly Media, Inc., first edit edition.

Zimmermann, O. (2017). Architectural refactoring for the cloud: a decision-centric view on cloud migration. *Computing*, 99(2):129–145.

Web (a). Cloud-Native Applications: VMware. Available at: `https://www.youtube.com/watch?v=gEeHZwjwehs` [Accessed January 30, 2017].

Web (b). Cloud-Native. Available at: `https://pivotal.io/cloud-native` [Accessed January 25, 2017].

Received February 1, 2017 , revised March 13, 2017, accepted March 14, 2017