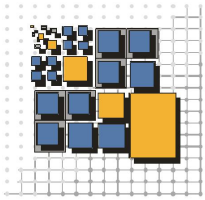

University of Bamberg

Distributed Systems Group

Master Thesis

in the degree programme International Software Systems Science
at the Faculty of Information Systems and Applied Computer Sciences,
University of Bamberg

Topic:

Cloud function Life cycle and tool support

Author:

Muhammad Faisal Imran Khan

Reviewer:
Prof. Dr. Guido Wirtz

Date of submission:
February 10, 2023

Contents

List of Figures

Abbreviations

DNS Domain Name System

DPT Defect Prediction Tool

DT Decomposition Tool

FaaS Function as a Service

GMT Graphical Modeling Tool

IaaS Infrastructure as a Service

IDE Integrated Development Environment

PaaS Platform as a Service

Rest Representational State Transfer

SaaS Software as a Service

SOA Service-oriented architecture

VM Virtual Machine

VT Verification Tool

1 Introduction

Function as a Service (FaaS) is a very popular cloud computing model. It makes its developer almost independent from infrastructure maintenance. Platform as a Service (PaaS), Software as a Service (SaaS) and Infrastructure as a Service (IaaS) are also very popular cloud computing models. These models were developed before FaaS. There is a lot of research work into how PaaS, SaaS and IaaS differ from each other. But there is no clear definition of how FaaS differs from PaaS, IaaS and SaaS. So, I investigated this issue and tried to differentiate them based on my studies.

FaaS frameworks are built upon stateless functions. This function has a single responsibility and a low cost compared to the full system. There is not much research work about the life cycle for these functions. By definition, lifecycle - is the different steps of life for an entity [1]. A software lifecycle is the different steps of software production, from planning to development and development to maintenance etc. In this thesis, I propose a lifecycle of cloud functions.

For this purpose, in this thesis, I have developed a small software named TEXT-ADD. It constitutes a single function that appends text to the input value. I developed the function using python and then hosted it on OpenFaaS [2] and then monitored it with Prometheus [3]. I also proposed a lifecycle for cloud functions based on my experiments.

As a monitoring tool, I have used Prometheus. This tool helps me with visualization of the function's performance over time. It helps to decide on the update, phaseout or termination of the function. I have discussed these steps in detail in this thesis.

There are many traditional software development models - waterfall, agile models etc. One of the traditional software models is the waterfall model [4]. In the waterfall model, every stage starts after the completion of the previous stage and has no re-iteration of the previous stage. I take this model as a baseline and compared my developed cloud function lifecycle with it. I also discussed a staged model of software development and compared my model with it. There are some agile modern software development models like scrum, Xtreme etc. I discussed them also and described why they are not comparable with my developed model.

DevOps is an advanced version of the software development lifecycle which promises much faster software delivery. RADON is a project developed by many organizations in the European region, which focuses on developing a DevOps system for the FaaS cloud computing model. I discussed this system in detail in my thesis.

Continuous Deployment is one of the practices of DevOps I have tried to achieve for cloud functions. I minimized the code required to deploy the system to a production server. Continuous Integration is another practice of DevOps that ensures quick integration of fixes by the developer to the code base. I discussed how one can achieve it by using tools like GitLab.

Thesis Organisation

[1] https://www.oxfordlearnersdictionaries.com/definition/english/life-cycle?q=life+cycle
[2] https://www.openFaaS.com/
[3] https://docs.openFaaS.com/architecture/metrics/
[4] https://www.geeksforgeeks.org/software-engineering-classical-waterfall-model/

In this research, I mainly had four objectives. I formulated my research questions based on my objectives. Then, I used a qualitative approach to solve my research questions.

I have created 4 different sections to discuss the solution to 4 different research questions. In the first research question, I discussed the difference between PaaS, FaaS, SaaS and IaaS. In the second research question, I developed a FaaS monitoring system and proposed a lifecycle out of it. Then I discussed the difference between my lifecycle and other different traditional software lifecycles. Finally, I discussed DevOps architecture, a DevOps framework for FaaS solutions and my approach to improving FaaS monitoring system to a DevOps one via implementing Continuous Deployment. Lastly, I summarize my findings in this thesis and proposed ideas concerning how one can improve the developed system with Continuous Integration and other DevOps practices.

2 Objective and Research Questions

Objective

In this research, I tried to answer many questions concerning the FaaS(Function as a service) cloud computing model. I have four main objectives in this thesis.

1. Formulating the difference between FaaS and other cloud computing models (PaaS, SaaS and IaaS), is the first objective of this thesis.
2. The next objective is to develop a lifecycle for the cloud functions via monitoring a sample FaaS system.
3. The third objective is to discuss the differences between my lifecycle and the traditional software development models.
4. Lastly, studying a developed DevOps FaaS system named RADON, and improving my lifecycle with more automation like DevOps systems.

Specific Research Questions

1. What is the difference between FaaS, PaaS, SaaS and IaaS?

   There is a lot of research work about how PaaS, SaaS and IaaS differ from each other. But how FaaS differs from them is not evident in most research. The goal of this question is where FaaS architecture stands compared to IaaS, PaaS and SaaS following the comparison criteria from other studies.

2. How to use tools to monitor a FaaS system? Develop a FaaS lifecycle from observations.

   The lifecycle for cloud functions is not very clearly defined in many of the publications concerning FaaS development. The tooling used to monitor FaaS systems is also vague in most research works. So the goal of this question is to find the proper tool to monitor the FaaS system and, upon observation, develop a lifecycle for the cloud function that this FaaS system hosts.

3. How the developed model differs from the traditional software lifecycle?

   FaaS functions are stateless, single-responsibility units of code. They are quite different than typical full-fledged software. So, how their lifecycle differs from traditional software lifecycles is quite an interesting question. I will answer this question in this thesis.

4. Discuss a FaaS DevOps system in Detail and How to evolve my system to a DevOps system?

   I have discussed a DevOps system for FaaS named RADON. RADON develops an optimized deployable model of FaaS solutions for different FaaS providers. It also uses Continuous Integration and Continuous Deployment. The steps of the DevOps system are - plan, measure, develop, test, release, deploy, monitor and optimize [Gok21]. These steps can be expanded to continuous business planning, collaborative development, continuous integration, continuous testing, continuous release, continuous deployment, continuous monitoring, continuous customer feedback and optimization [Gok21]. Continuous Deployment enables quick deployment of the software as soon as a new merge is integrated with the codebase so that it gets early feedback from customers [Gok21]. Continuous Integration is another practice of DevOps that ensures quick integration of fixes by the developer to the code base. In this thesis, I tried to achieve Continuous Integration and Continuous Deployment in my developed model so that it can evolve towards the DevOps architecture.

Methodological Approach The research methodology I have used here is a qualitative approach [5].

Some keywords, I have used to search are -

1. Cloud function as a service
2. Cloud function Lifecycle
3. Differentiation between IaaS, PaaS and SaaS
4. Software Lifecycle Etc.

I have excluded research papers which have conflicting information – like this research paper [Kij18] says - Serverless and FaaS are the same, which is conflicting with another research paper I have used in this thesis [VEIST17].

I have excluded some studies which are not directly related to my objective of the thesis, like snafu [Spi17].

I have followed the steps below while answering the research questions,

1. I started my research with Literature Review. I used many platforms to gather information to solve my first research question. I researched the journals from

[5] https://www.gcu.edu/blog/doctoral-journey/what-qualitative-vs-quantitative-study

Semantic Scholar [6], Google Scholar [7], Research Gate [8] and Dblp [9] on the topics of the research questions.

2. Then I researched the implementation of tool support for FaaS systems and then I implemented tool support to monitor a FaaS system. I used OpenFaaS as the host and Kubernetes [10] as the orchestrator and Prometheus for monitoring to create the FaaS environment. Using the tools, I developed a single function that resembles text appending software. This function appends static text to the input text. Then I found out different phases of the cloud function lifecycle via monitoring the FaaS system.

3. For the third question, I researched different traditional software development systems and I compared my TEXT-ADD system with them.

4. For the fourth question, I discussed DevOps systems in general. Then I also discussed a DevOps system for FaaS named RADON. Then I discussed how to improve my TEXT-ADD system as a DevOps one.

3 Related Studies

Here I summarise the research work, I have done for this thesis based on some common topics. To answer the research questions of this thesis, I have combined studies from these topics.

Differences Between different cloud computing models

[HY10, p.11] discusses the differentiation between IaaS, PaaS and SaaS. [AJFOG17, p.207] discusses the differentiation between FaaS and PaaS, the comparative costs for FaaS and PaaS development and the comparison of costs between different FaaS providers like AWS Lambda, Azure Functions, Google Functions and IBM Openwhisk. [WLSW21] discusses different decision criteria to choose between IaaS, PaaS and SaaS. Here, the authors have proposed that IaaS is to be used when more flexibility in customization of sizes of resources is required and the infrastructure of the system needs to be on market on short time notice, PaaS is required when the full system needs to be on market on very short notice and when the company wants to reduce the effort of building the software using outsourcing. Lastly, SaaS is preferable when software needs to be upgraded or developed with fewer human resources possible [WLSW21, p.6291]. [RR14] also discusses the differentiation between SaaS, PaaS and IaaS. It differentiates them on basis of - who is the user, what services it provides and why should one use it etc. It also discusses example applications for different cloud computing models.

[6] https://www.semanticscholar.org/
[7] https://scholar.google.com/
[8] https://www.researchgate.net/
[9] https://dblp.org/
[10] https://kubernetes.io/

Definitions of cloud computing models

[VETT+18, p.3] discusses the definitions of Serverless computing and FaaS and also discusses how cloud computing evolved from IBM VM to a Serverless framework from 1960 to the current age. [VEIST17] discusses the definitions of Serverless, cloud function and FaaS systems. [JSSS+19] describes the early formation of Serverless computing and the limitations and challenges of Serverless computing. [Kol19] also discusses the definitions of PaaS, SaaS and IaaS.

Example of different PaaS, FaaS and Serverless systems

In this research paper [YS15], they have proposed a PaaS platform based on Docker [YS15, p.668]. They also developed a dashboard to help users deploy their applications. [PB20] discusses a special FaaS system, that can resolve the interaction issues of small IoT devices with the FaaS system. This system ensures easy installation for IoT devices and proper communication via dedicated endpoint[[PB20, p.1]. [BBK+19] focuses on machine learning as a Service, to the end user using a Serverless platform called Stratum. Casale [CAVDH+20] discusses a DevOps FaaS system named RADON, which is one of the main focuses of this thesis. It discusses a model that integrates different tools to develop an optimized deployable multi-provider FaaS solution.

Different Software Lifecycles

[RVD+10] describes Software Development Lifecycle (SDLC). It says software development can be done in different ways and each way is called a software development lifecycle. In this paper, they have proposed a new hypothetical model of SDLC. Rajlich [RB00] discusses the staged model for software lifecycle. [dlT16] describes Docker Lifecycle. Docker Lifecycle is a DevOps lifecycle that uses the docker application in all of its steps. The first step of the docker lifecycle includes Code, Run, Test, and Debug in dev Environment [dlT16, p.10-11]. Then Continuous Integration, Integration Test, Continuous Deployment and Deployment happens [dlT16, p.10-11]. Lastly, it monitors and diagnoses the data during running and manages continuously in the outer loop [dlT16, p.10-11].

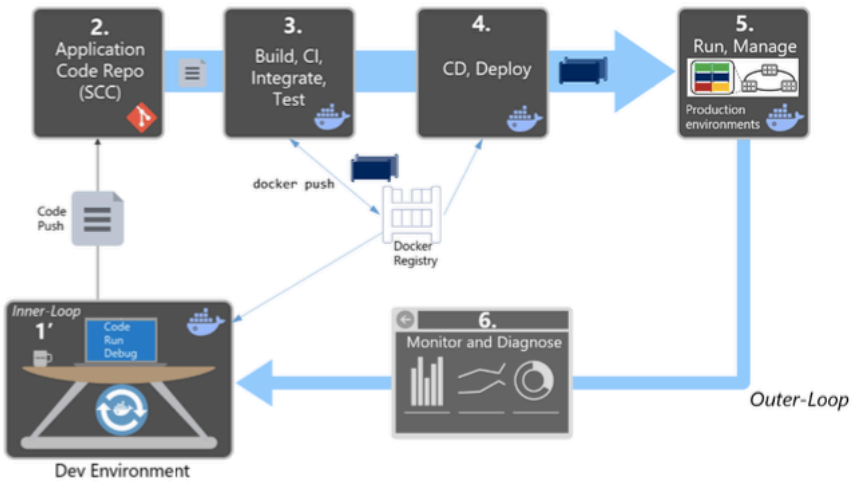

Figure 1: Docker Lifecycle [dlT16, p.09]

[Bec99] describes XTreme programming and [RJ00] describes a software development process for small teams named SCRUM. These two are faster approaches compared to

waterfall software development. [EGHS16] describes DevOps systems. DevOps is a software lifecycle that uses automation, deployment, and monitoring. It unifies development and operations and reduces the communication gap between them. [EK20, p.1] describes early error detection in the software development lifecycle. Early error detection is a cornerstone for a successful software development lifecycle. It reduces the stress on the maintenance step of the software development life cycle and reduces the cost of overall software development. But for this purpose, different companies use different policies [EK20, p.1]. [EK20] tries to summarise their policies by doing systematic reviews of existing research.

Differentiation Between FaaS platforms

Maciej et al. [MFGZ17, p.10] have developed a benchmarking system and have compared cloud functions from different providers like AWS Lambda, Azure Functions, Google Cloud Functions(GCF) and IBM Openwhisks based on criteria such as CPU share vs. memory allocation [MFGZ17, p.10]. Their study shows that AWS has a linear relationship between them. For GCF, it is not linear. For Azure and IBM, there is no relationship. The developer of the Cloud function is mostly unaware of how much memory is consumed by their developed function [Spi20, p.1]. [Spi20, p.1] has developed tools to determine memory consumption by functions developed in Docker, OpenFaaS and many other configurations.

4 Answer to Research Question 01

To answer research question 01. I start by briefly discussing Cloud computing, Virtualization, IaaS, PaaS, SaaS, FaaS, Serverless Architecture, Cloud Functions etc. Then I discuss whether FaaS is suitable for database-centric applications or not from different perspectives. Then, I propose a differentiation between IaaS, PaaS, SaaS and FaaS based on my studies.

Cloud Computing

In 1960 IBM Virtual Machine (VM) started the Cloud computing journey. The idea of Functional Programming, Concurrency, and Event sourcing also got popular at the same time [VETT$^+$18, p.3]. Then in 1970, the idea of remote procedure call was generated [VETT$^+$18, p.3]. In 1980, Domain Name System (DNS) was invented [VETT$^+$18, p.3]. In the 1990s, there was the rise of Service-oriented architecture (SOA) [VETT$^+$18, p.3]. In the 2000s the idea of Representational State Transfer (Rest), AWS cloud and Google App Engine [VETT$^+$18, p.3] was introduced. In 2010, there is the rise of Docker container, IaaS, PaaS and SaaS and also event-driven architecture [VETT$^+$18, p.3]. Later, Docker gave rise to Container Orchestration and Rest, URI etc gave rise to Microservices [VETT$^+$18, p.3]. Later, they combined and gave rise to FaaS. FaaS then combined with event-driven architecture and gave rise to the Serverless framework [VETT$^+$18, p.3].

Virtualization in Cloud Computing

Virtualization is the process of creating or imitating the original server, desktop or oper-

ating systems [11].

It is a process where a physical instance is shared between many people as a virtual instance [11]. It is done by offering a pointer of the actual resource to the vendors [11].

The physical machine is called the Host Machine and the machine that represents it, is called the guest machine [11].

Different Cloud Computing Models

Some of the Prominent Cloud Computing models like SaaS, PaaS, FaaS, IaaS Serverless etc. are discussed here.

SaaS is built on top of PaaS. PaaS is built on top of IaaS.

Infrastructure as a service (IaaS)

IaaS is the cloud computing model that provides its customers with mainly virtualized server instances [Kol19, p.22]. Virtualized server instances have virtualized CPUs [Kol19, p.22]. The number of CPUs is configurable by customers [Kol19, p.22].

IaaS also offers storage [Kol19, p.23].

Storage can be of two types:

Block storage: stores a huge amount of data [Kol19, p.23]

Object storage: stores data as an object with a unique identifier [Kol19, p.23].

IaaS also offers network management on a small scale [Kol19, p.23].

The IaaS market is dominated by AWS, Microsoft Azure, Alibaba cloud, and Google Cloud [Kol19, p.23] [12].

Platform as a service (PaaS)

PaaS gives a platform consisting of (a runtime environment, hardware infrastructures) etc., where developers can run their applications [Kol19, p.24].

In PaaS, a platform is offered as a service [Kol19, p.44]. The consumers plug and play their applications on top of this platform.

The platform offered by PaaS can be abstract or at a much lower level [Kol19, p.45].

For the abstract case, it is much more user-friendly, as a user doesn't need to bother much about the configurations. It is often called the black box model [Kol19, p.44].

For the low-level type, people need to do some configurations by themselves before deploying applications [Kol19, p.44]. The developer develops their projects in the local environment [Kol19, p.44]. Then host it to this type of PaaS [Kol19, p.44]. It offers much more customization, so developers love to use it. It is often called the white box model [Kol19, p.44].

[11] https://www.javatpoint.com/virtualization-in-cloud-computing
[12] https://www.gartner.com/en/documents/3803767

Software as a service (SaaS)

SaaS providers provide software as a service and the customer buys access to the service [Kol19, p.25]. Customers do not require any installation as the software is mainly web-based [Kol19, p.25]. Service providers provide support via continuous web updates to meet requirements from different customers [Kol19, p.25]. Thus, it tries to maintain consistency and security among many people, at the same time [Kol19, p.25]. SaaS uses technology like Ajax, CSS and Javascript [Kol19, p.26].

An example of SaaS is a study portal, where students enroll for different courses. Each student can see her course list. When multiple students try to access the same course at the same time, it does not break down the full system.

Serverless Architecture

Serverless cloud architecture demonstrates 3 main ideas – [VEIST17, p.2]

1. Only when the application runs, consumers need to pay [VEIST17, p.2].
2. Operation logic maintained by the service provider [VEIST17, p.2].
3. Serverless applications are short-lived and they are alive as long as they are needed [VEIST17, p.2].

Serverless can be seen as an intermediate cloud computing model between PaaS and SaaS [VEIST17, p.2]. There are some exceptional cases, as some PaaS or SaaS solutions are also Serverless[VEIST17, p.2]. For example, (Auth 0) Serverless authentication and (Fauna) Serverless database [VEIST17, p.2].

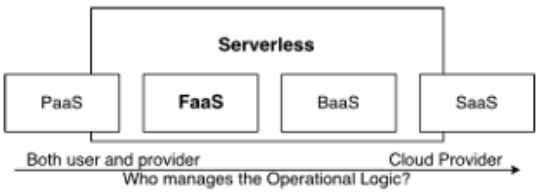

Figure 2: Serverless [VEIST17, p.2]

There are two new architectures which fully conform to Serverless criteria - FaaS and BaaS [VEIST17, p.2]. FaaS is all about cloud functions. BaaS is a much more business-specific solution provider [VEIST17, p.2].

Serverless is a very powerful software technology because it enables the developers to quickly build a prototype of a cloud-based application and it has less concern for infrastructure management.

Cloud Function

Cloud function plays a key role in FaaS systems.

Cloud function has four main properties -

1. Cloud functions are not a huge program but a small part of the full program [VEIST17, p.2].
2. Cloud function doesn't have a state, which means it does not store data in the database [VEIST17, p.2].
3. Cloud function works only for pursuing a single purpose [VEIST17, p.2].
4. Cloud function is also deployed on demand. That means it is alive as long as it is needed [VEIST17, p.2].

FaaS

FaaS is the cloud computing model, which enables developers to host functions in the cloud. FaaS requires almost no operational logic from the developer side [VEIST17, p.2].

FaaS consists of 3 layers: Event layer, Function layer and logic layer.

The Event layer determines at which event the middle layer (Functions) will be executed [VEIST17, p.2].

Google Functions have more overall costs than AWS Lambda, Azure Functions and IBM Open Whisk [AJFOG17, p.210].

The difference between PaaS and FaaS is that in PaaS the developer hosts the full program in the cloud at once and for FaaS, the program is divided into small functions and hosted separately from each other [AJFOG17, p.208]. So, when the user or customer wants a service, only the necessary functions are executed [AJFOG17, p.208]. That results in comparatively low cost for FaaS models [AJFOG17, p.208]. These functions also run for a short time [AJFOG17, p.208]. As soon as the request is closed, the function execution is terminated [AJFOG17, p.208].

If the developer wants to change a feature, changing only the required function is enough [AJFOG17, p.208]. But for PaaS, to change a small feature, one may need to update the full project [AJFOG17, p.208].

There are two types of arguments for using FaaS with database-centric applications. One says database FaaS is not suitable for database-centric applications and, in another case, FaaS can be used with database-centric applications. I observe both cases -

First Case: FaaS - Connection Pooling Limitation [13]

FaaS is regarded as a solution that will infinitely scale [13]. The function in the FaaS can scale infinitely but the resources upon which the function depends do not scale infinitely [13].

A relational database has concurrent connections which are required for FaaS [13].

In FaaS, each instance of a function lives in a stateless environment [13]. When it connects to a relational database (PostgreSQL, MySQL, Oracle), it will use a connection pool to avoid reconnecting with DB [13]. The number of the connection pool is limited for a

[13] https://www.freecodecamp.org/news/the-serverless-series-mistakes-you-should-avoid-9ec1ca6b9dff/

database (usually 20) [13]. If there are more than 20 instances of the function, then the database will be exhausted [13]. This is the main disadvantage of FaaS regarding database connections [13].

So the author in [13] recommended not using FaaS when communicating with relational DB. Instead, he suggested these options [13]:

1. BaaS
2. PostgreSQL offers plugins that solve the problem by multiplexing the number of available concurrent connections.

How do FaaS systems work without a database? An example solution:

In a stateless system, the state of the previous transaction is not stored and a new transaction starts from scratch [14] [15]. And the stateful system is the opposite.

In the web context, suppose a user logs into a shopping cart website. She bought something that is stored in the database. Later logs in again, and she can see her purchase history. So this is an example of a stateful function.

But suppose a simple timer that gives you a beep after a certain time. After this function is executed, it does not need to save the value in the database for later usage. This type of function is stateless.

When a user logs in again after a certain time and then uses the timer again. This second-time usage of the timer has no dependency on the previous time usage of the timer. This is an example of a stateless function.

Suppose a typical system consists of a login system, a timer and a timer history tracking system.

Only the stateless functions are taken out from the main system to constitute the FaaS system. For the rest, a replacement PaaS system is being developed.

A PaaS system consists of a login system and a timer history tracking system. And FaaS system consists of the login system and the timer.

The benefit of this FaaS is that, if I want to update the timer function, I can do it independently of other resources.

For the function in the PaaS system, if I change the function, I need to upgrade the database and, ultimately, the full system.

IaaS vs PaaS vs FaaS vs SaaS

PaaS is a much simpler version than IaaS, from a user perspective. For PaaS, users don't need to be concerned about infrastructure like IaaS.

Also, FaaS is a much simpler version than PaaS for users. Because FaaS uses stateless functions, there is no need for database management. Because I assume the database

[14] https://www.spiceworks.com/tech/cloud/articles/stateful-vs-stateless/
[15] https://www.redhat.com/en/topics/cloud-native-apps/stateful-vs-stateless

is managed by supporting (BaaS or PaaS) systems as described above. So, ultimately, the database is abstracted from the developer of FaaS. The developer only needs to be concerned about the stateless function to be hosted in FaaS.

From traditional IT, more automation is applied to IaaS systems. Then, the PaaS system comes, which is much more automated. Then comes FaaS, only stateless functions, and function execution on demand rather than a full application. Then comes SaaS, which is more automated than FaaS. Even the applications are already developed in SaaS.

Previously, Microsoft [HY10, p.11] proposed a model with Traditional IT, IaaS, PaaS and SaaS.

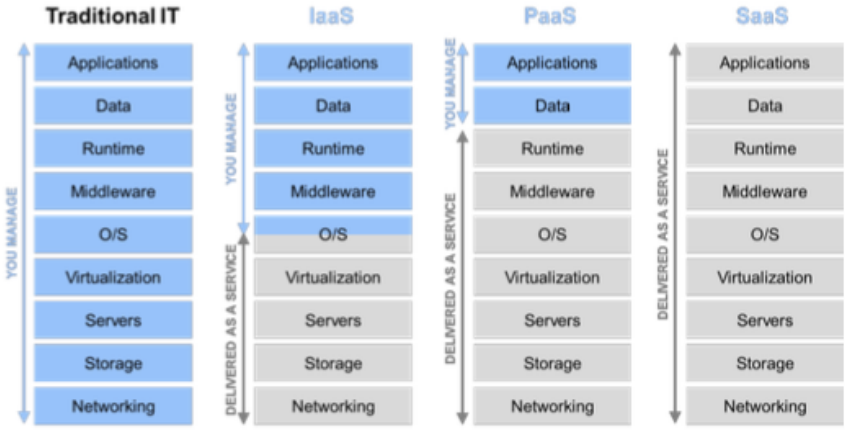

Figure 3: IaaS vs. PaaS vs. SaaS [HY10, p.11]

With FaaS, I can extend this model as in this figure -

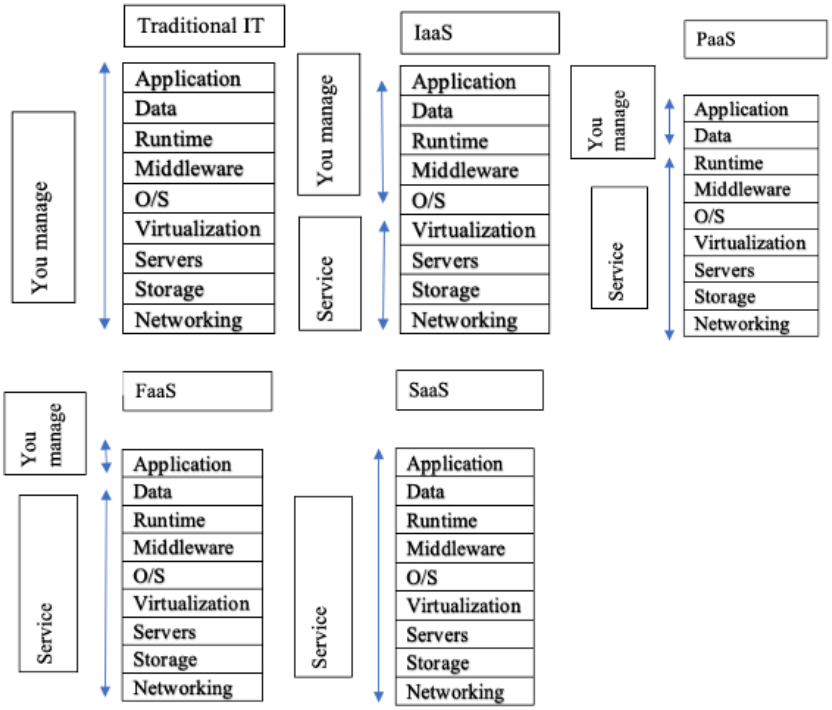

Figure 4: IaaS vs. PaaS vs. FaaS vs. SaaS

Second Case: FaaS can be used with database-centric applications

There are also some arguments that says - FaaS can be used with database-centric applications, One of the ways to do it is by queuing database operations and having a fixed-size fleet of instances doing DB operations - so decoupling cloud functions and the database. Here PaaS and FaaS do not show any visible difference.

IaaS vs PaaS vs FaaS vs SaaS final comments

The main difference between FaaS compared to other models, it hosts small functions like microservices [16] whereas other models host mainly complete system.

5 Answer to Research Question 02

To answer research question 02. I have developed a system named TEXT-ADD that consists of a simple function in python. Then I tested this function and deployed it in OpenFaaS. Then I monitored the function invocation using Prometheus. Then I updated the function to simulate the maintenance step. Executing all these steps, I come up with a lifecycle for the cloud function. Sourcecode: [17].

Implementation

I have implemented different phases for a simple python [18] function hosted on OpenFaaS.

1. Development Phase: In this phase, I develop my function in the Microsoft VS code. demonstration video for test, deployment, update/maintenance phase: [19]
2. Test Phase: In this phase, I changed the output of the cloud function in the test file. As a result, when trying to build the function, it gives me an assertion error. Later, I make the test function output identical to the original function output. As a result, the function is built without any errors.
3. Deployment Phase: Before this deployment, I need to push the built image into the docker hub. After the push is successful, I need to authenticate to the OpenFaaS cli. Then, I can deploy my function to the OpenFaaS cloud. After successful deployment, I can see my functions in the OpenFaaS Cloud. If I invoke the function with a request body parameter, it will output me the result of the cloud function.
4. Monitoring phase: In this phase, I can see the output of the invocation of my cloud function in various metrics using Prometheus. It is automatically installed when I install OpenFaaS in my first step. I need to port forward to an outside port (Example: 9090) so that I can see the Prometheus DashBoard. In the demonstration video, I used the "gatewayfunctionitalinocationtotal" function. It shows me the total invocation of my function compared to time. Demonstration video for monitoring phase: [20]. This monitoring phase can lead to the update phase. In the graph view, one can see how frequently a specific function is invoked at a specific time.

[16] https://microservices.io/
[17] https://github.com/imrankhan15/Implementinglocal-Serverless-functions-deployment
[18] https://www.python.org/
[19] https://youtu.be/M6Mfh89AWeE
[20] https://youtu.be/YqoMHX5ybWk

Suppose, there is a counter function and it should be called after a certain time. But in the graph, if I see that it is not being called after a few days, I can decide there is an error in the function and I can decide to go to the update phase. I can also go to the Phaseout Phase from monitoring if I decide not to update anymore.

5. Maintenance/Update Phase: In this phase, I develop the code again and I change the function and its tests. Then I repeat the build, push and deploy again. Then the function in the OpenFaaS dashboard will be changed, and it will give me the output for the changed function.

6. Phaseout Phase: From my monitoring, if I find, updating the function is commercially not beneficial, I go to the phase-out phase and the function continues to serve the customers in its old version.

7. Termination Phase: Then, if I want to remove the function from the market, I go to the termination phase. On the OpenFaaS dashboard, one can delete the function. It will destroy the function and the lifecycle is over for that function.

I have come up with this lifecycle for a cloud function based on my implementation.

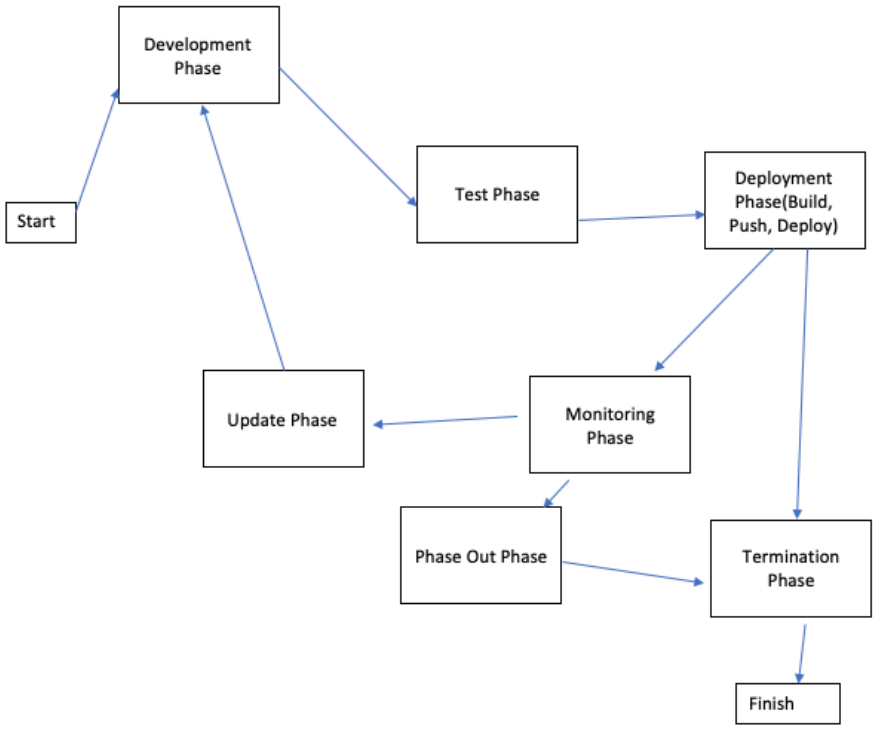

Figure 5: Lifecycle of a cloud function in TEXT-ADD system

How I use tools to develop lifecycle for cloud functions

In my demonstration video, [21] I used Prometheus to monitor the cloud function performance in the FaaS system. I invoked the "gatewayfunctioninvocationtotal" function and

[21] https://youtu.be/YqoMHX5ybWk

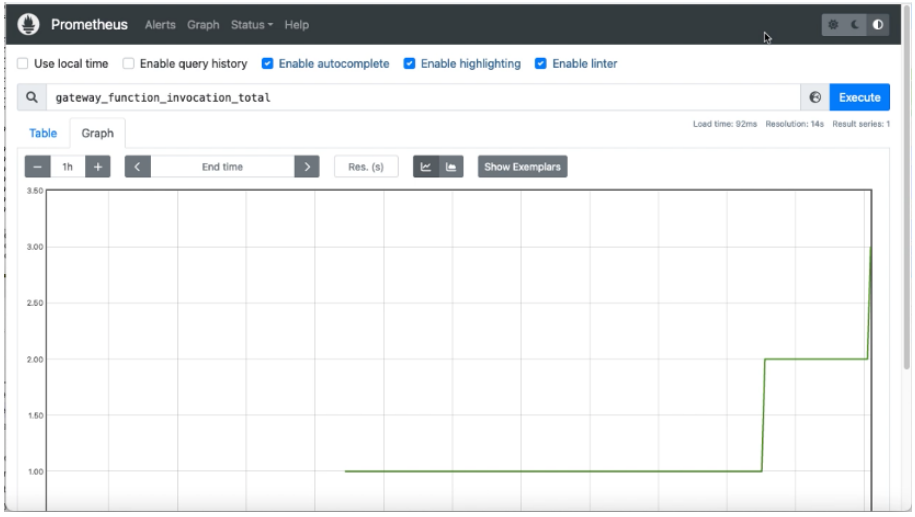

Figure 6: Prometheus Output

it gives me a constant result. That means my function is working properly and does not need an update. But if there is any discrepancy, like I get different values at a different time, then I understand that the function is not working correctly. Then I have two options. I can go to the update phase or phase-out phase. If I find out fixing the function is economically feasible, I go to the update phase and if I find out it is not feasible, I go to the phase-out phase.

Thus the graphs of Prometheus help me decide between different phases of the lifecycle for cloud functions.

6 Answer To Research Question 03

To answer my research question 03, I differentiate my developed model from different software development models. I discussed at first the waterfall model. Then I tried to compare it with my TEXT-ADD system. Then, I discussed another similar model called - the staged model. Lastly, I discussed agile software development models: Scrum, Xtreme etc. and then compared my model with them.

Waterfall Model [22]

In this model, software development is divided into phases [22]. After one phase is fully finished, then the next phase starts [22]. So the development of software in this model looks like stairs or a waterfall to be precise [22].

Let's discuss different phases in the Waterfall Model:

1. Feasibility Study [22]:

   The main target of this phase is to understand if it is justified to undertake this project economically and technically [22]. So the first step is to understand the prob-

[22] https://www.geeksforgeeks.org/software-engineering-classical-waterfall-model/

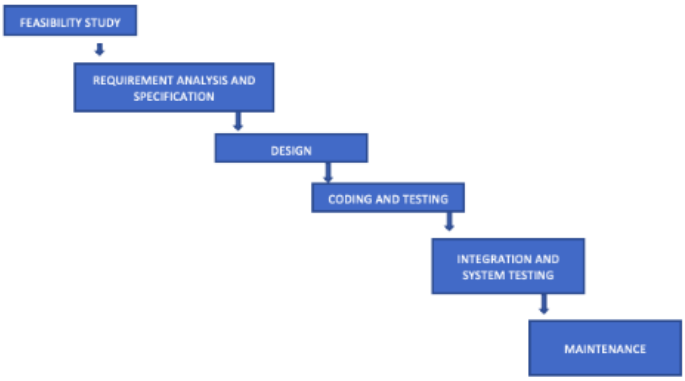

Figure 7: Waterfall model by GeekforGeeks [22]

lem and find way-outs that can be a solution to this problem [22]. Then the best way out is chosen and it becomes the starting point for the next steps [22].

2. Requirement Analysis and Specification [22]:

   This step is about gathering enough data from the customers and then putting it into a document [22]. This phase can be broken into two branches: [22]

   (a) Requirement Gathering and Analysis [22]: In this step, all the needs of the clients are summed up [22]. Then they are filtered from three types of error-prone requirements [22] - incorrect requirements, incomplete requirements and requirements which do not conform to other requirements [22].

   (b) Requirement Specification [22]: Requirements gathered from the previous step are summed up into a special type of document [22]. This document is called software requirement specification(SRS) [22]. In future, if there is any misunderstanding between the software manufacturers and the buyers, this document can justify the validity of their claim [22].

3. Design [22]: In this phase, the requirements documented in the SRS document are transformed into a design [22]. Design can be like a hand sketch or like a prototype of the original software [22].

4. Coding and Unit testing [22]: In this phase, the design is translated into a full-fledged computer program [22]. Each small portion of the design is transformed into a part of the code base [22]. Then each portion is testified vigorously with unit testing to find out the conformance of the design and the code [22].

5. Integration and System testing [22]: The ultimate test is called Integration Test [22]. In this test, one test the full project, and all the modules completely [22]. At the start, one take a single module test it, then merge another module and test it and this cycle continues [22]. If this is working properly, that means the full project can work together and the system is complete and tested [22]. Additionally, there is another set of tests [22]. They are called system tests. System testing is comprised of 3 categories:

   (a) Alpha testing: The full system is tested in Alpha testing by the original developers of it [22].

(b) Beta testing: Some people, who are not part of the original developer team, like known customers or well-wishers, test the system again [22].

(c) Acceptance testing: After the deployment is complete to the customers, they check out if the full software is running or not [22].

6. Maintenance:

   The most important step of software development is the last step - maintenance. The lifecycle spans about 60 percent of the time in the maintenance step [22]. It can be categorized into 3 types: Corrective, Perfective and Adaptive [22].

   (a) Corrective Maintenance: Suppose the application consumer found a bug in their installed software and then she sends it back to the software company to fix it. And the software company fixes it. It is called corrective maintenance [22].

   (b) Perfective Maintenance: Suppose the customer wants a new button in the user interface or a page for the website, so she sends it to the software company. And the company builds the new feature. It is perfective maintenance [22].

   (c) Adaptive Maintenance: Suppose the customer bought a new office in another city and wants to upgrade the software so that both branches can use the same software. In that case, the software company will upgrade the software to support both branches. It is called Adaptive maintenance [22].

Comparison between the Waterfall Model and my model

1. Feasibility study, requirement analysis and specification and design steps of the Waterfall model are equivalent to my first step - coming up with an idea of a cloud function that is good for the demonstration of lifecycle steps.

2. I have implemented the Coding and Unit Testing [23] phase, as I did an assertion unit test for my simple function.

3. I did not implement integration and system tests, I omitted them because the assertion unit test was sufficient for my simple function.

4. I did maintenance. As for the maintenance step, I did perfective maintenance. Because I wanted new feature in my functions.

5. I additionally have deployment, monitoring, phaseout and termination phases which are not available in the traditional waterfall model.

6. The most important distinction is that in the Waterfall model, there is no iteration from the maintenance phase to the coding and testing phase, but in my developed model, I go from the update phase to the development phase through iteration.

Staged Model of Software Lifecycle

Staged model for the software life cycle – starts with initial development and produces the first running version [RB00, p.67]. Then, in the evolution phase, it evolves a few

[23] https://www.techtarget.com/searchsoftwarequality/definition/unit-testing

rounds [RB00, p.67]. In this phase, developers develop the software according to the client's requirements [RB00, p.67]. Then, when it cannot be evolved anymore, it is put into servicing phase [RB00, p.67]. In this phase, the developers defect repairs [RB00, p.67]. When service is discontinued, it is moved to the phase-out phase [RB00, p.67]. In this phase, the company doesn't service the software anymore and makes revenue from customers for as long as possible [RB00, p.67]. Then it is switched off and put in the close down phase [RB00, p.67]. In this phase, the company removes the software from the market and proposes an alternative solution to the previous customers [RB00, p.67].

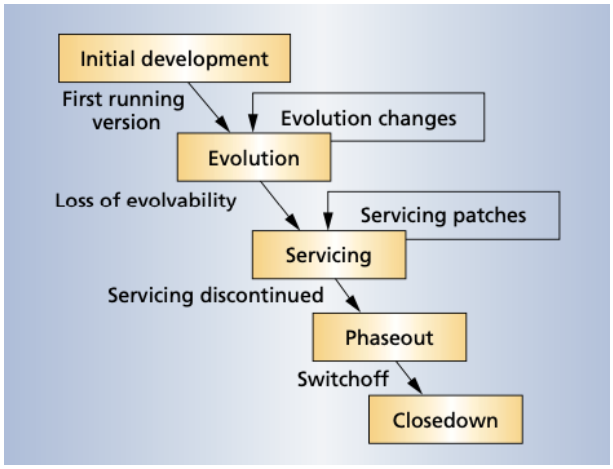

Figure 8: Staged software cycle model by Rajlich [RB00, p.67]

Comparison with staged Model and my model

1. The development, test and deployment phases of the staged model are similar to the initial development phase in my model.
2. Evolution phase is similar to the monitoring and updating phase in my model. Servicing is equivalent to the update phase.
3. Then the close-down phase is similar to my termination phase.
4. The difference between the staged model and my model is - the staged model has no iteration from servicing to the evolution phase but my model has iteration from update to development.

Xtreme and Scrum

Xtreme is a software lifecycle which has small releases rather than a complete system at once [Bec99, p.71]. It maintains a process that has no duplicate code [Bec99, p.71]. It has the least possible number of classes [Bec99, p.71]. In this production system, two people work parallelly to develop the same unit of code [Bec99, p.71]. This system has Continuous Integration [Bec99, p.71]. Also, during production in this system, a customer always stays onsite with the development team [Bec99, p.71].

Scrum starts with an initial planning phase [RJ00, p.30]. In this phase, the project team develops a plan and selects a chief architect for the project [RJ00, p.30]. The architect defines the target of the project and controls and communicates with the developers [RJ00,

p.30]. After initial planning, small incremental steps called sprint - happens [RJ00, p.30]. The team takes all the current issues and assigns them to the backlog [RJ00, p.30]. Before each sprint, teams collect prioritized tasks from the backlog and solve them in the sprint [RJ00, p.30]. A sprint normally consists of 1-4 weeks [RJ00, p.30]. The delivery date and the task to be done in the sprint are fixed [RJ00, p.30]. During the sprint, the team daily holds scrum meetings with all onsite and remote developers [RJ00, p.31]. It serves a daily update and a team-building purpose [RJ00, p.31].

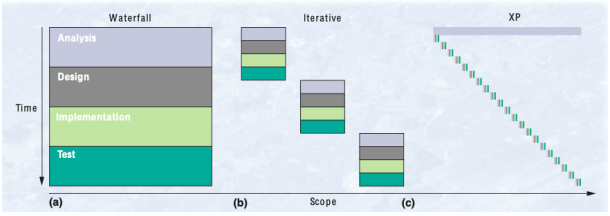

Figure 9: Xtreme programming [Bec99, p.70]

Comparison with Xtreme, Scrum and my model

Xtreme and Scrum are mainly designed for enterprise applications and designed for more than one developer. The model I developed for this system is a single cloud function, developed as a personal academic project where I don't have additional functionality like Pair Programming (Xtreme) or scrum meetings (Scrum) etc. in my developed model.

7 Answer to Research Question 04

In this section, I discussed the DevOps Model. I also discussed the RADON model, which is a developed FaaS system that adopts the DevOps philosophy. Then, I reconstruct the deployment part of the TEXT-ADD system to meet Continuous Deployment criteria in DevOps. Lastly, I propose future work to enhance my model to meet Continuous Integration criteria in DevOps.

DevOps Definition [24]

DevOps is a software development process that performs software delivery comparatively faster than traditional development [24]. It maintains some particular philosophies and practices to achieve it [24].

In DevOps philosophy, the development team and the operation teams are unified into a single team [24]. Everyone in the team is responsible for code checks to find security loopholes in the codebase [24].

The first of the areas for DevOps is Planning. Plans for DevOps are small, and changes are continuously related to business changes. It ensures management of requirements, change and work [Gok21]. DevOps ensures a pipeline of software development [Gok21]. Unit Tests are written by developers and these tests are executed repeatedly along with Continuous Integration [Gok21]. This step ensures rapid software development and quick

[24]https://aws.amazon.com/devops/what-is-devops/

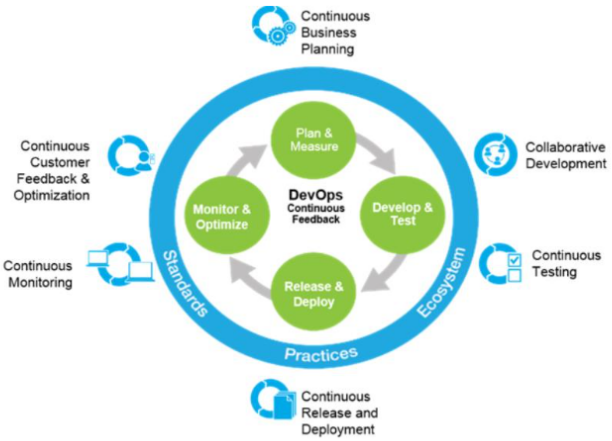

Figure 10: DevOps scope [Gok21]

feedback from customers [Gok21]. The performance and availability of the application are monitored continuously - using tools like logging [Gok21].

Additionally, there are some common practices of DevOps.

Dev Ops practices [24]

1. Developers perform very frequent but small updates [24] .
2. Developers merge updates into a base system like Git continuously [24].
3. After merging code changes are tested, built into a single unit, released and then deployed to the customer's machine [24].
4. Large, complex systems are broken into simple, independent Microservices [24]. Each Microservice works for a single function [24].
5. Developers can configure the system programmatically, rather than manually set up and configure resources [24].
6. Organizations monitor and log the end user's experience of their applications [24]. Based on this monitoring and logging, they set up their next update or release for the application [24].
7. The communication and collaboration between the development team, operations, marketing, sales etc. are made more robust using technologies [24].

Waterfall vs DevOps [25]

I discuss the differences between traditional IT and DevOps.

1. For Waterfall, Batch Size is Big, whereas for DevOps it is micro [25].
2. Scheduling is centralized in Waterfall, whereas for DevOps it is decentralized [25].

[25]https://devops.com/comparing-devops-traditional-eight-key-differences/

3. Waterfall follows the culture of sure success and ensures that it will never fail [25]. For DevOps, the culture is to fail early [25].

4. In Waterfall, each step is executed only once [25]. So there is no chance to recover from failure [25]. For DevOps, there is iteration to the same step [25]. If it fails in the first iteration, then in the later iterations it will fix the error [25].

My System vs DevOps

1. In my developed model, if I change my code and update it, it is still on the local desktop. In the DevOps situation, after updating the code on the local desktop, there should be a command that will automatically merge the changes in the local desktop to the code base in the cloud.

2. My developed function has only a single purpose. So, it is also a microservice that matches the DevOps system.

3. One of the practices of DevOps is Continuous Deployment, which means - deploying the system in a faster way so that it can get customer feedback faster and can gain customer satisfaction. In my developed system, I developed the python function in VS Code and then called from terminal 'build', 'push', and 'deploy' commands to deploy it to the OpenFaaS host. Here, If I replace these 3 commands with a single command, then I can deploy it to the server much faster. I have worked on achieving this feature in this thesis later.

DevOps FaaS system "RADON" study [26]

A DevOps framework for FaaS is an Open Question, RADON wants to address this issue [CAVDH+20, p.77]. RADON's vision is to use open-source technologies along with low-cost FaaS technologies to host applications, to help small and medium-sized enterprises [CAVDH+20, p.78].

RADON is a 3 year EU research project with 8 organizations: Imperial College, JADS, XLAB, Engineering Informatica, U. Tartu, ATC, U. Stuttgart, and Efficode [26].

RADON WorkFlow From User Perspective

This figure describes how RADON works from a user perspective [27].

[26] https://www.youtube.com/watch?v=-PDWknaXTM8
[27] https://radon-h2020.eu/wp-content/uploads/2021/09/D2.7-RADON-integrated-framework-II.pdf

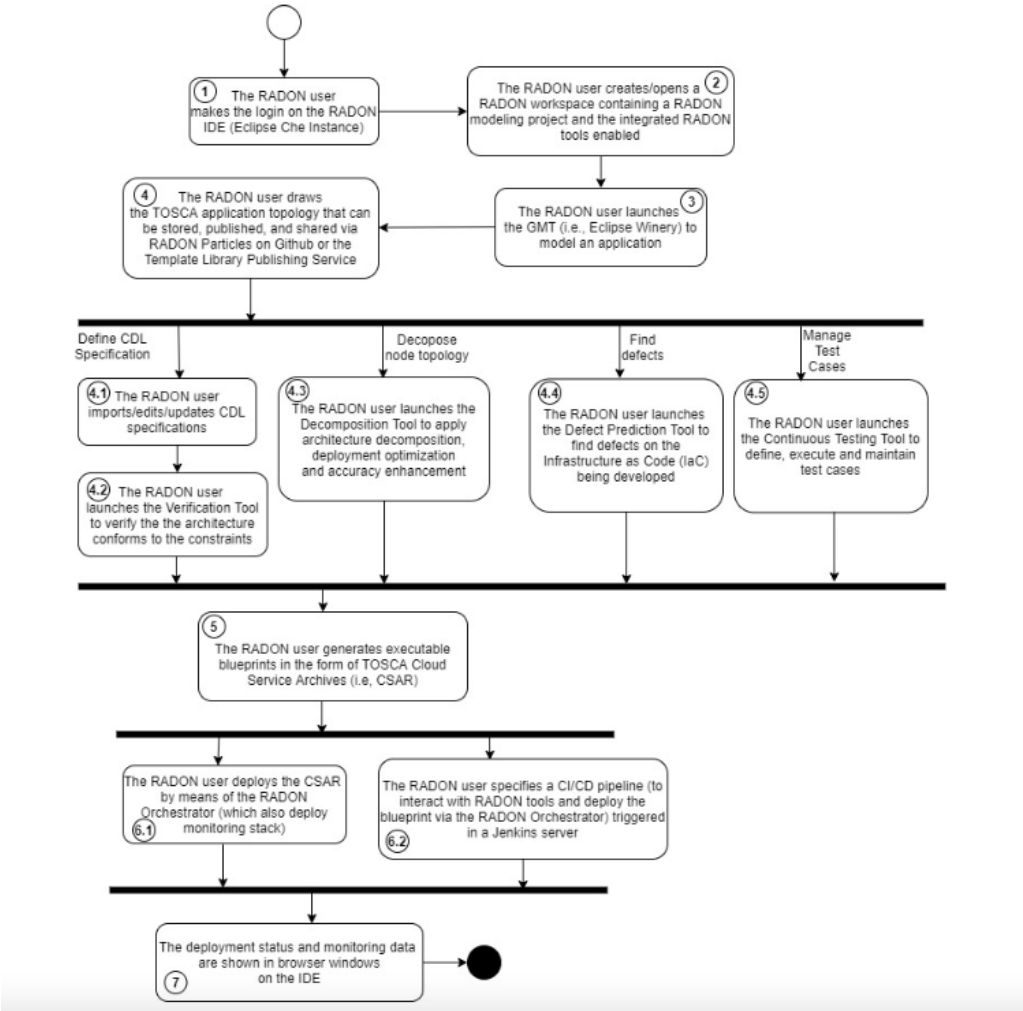

Figure 11: RADON Workflow from User Perspective [27][p.17]

Important Features of RADON Workflow

RADON users can use the GMT tool to design the full application cycle. Users can decompose the model with decomposition tools so that, it can be deployed at a minimum cost. RADON users can deploy the same application to different vendors like OpenFaaS, Microsoft Azure etc. RADON users can use Jenkins Jobs (CI/CD) to automate the deployment process to customer machines.

RADON Overall Execution and RADON Particles in Detail

Overall execution of RADON: [28]

This use-case diagram [29] shows what RADON users and RADON administrators can do with the RADON IDE.

[28] https://radon-h2020.eu/wp-content/uploads/2019/07/D2.1-Initial-requirements-and-baselines.pdf
[29] https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/

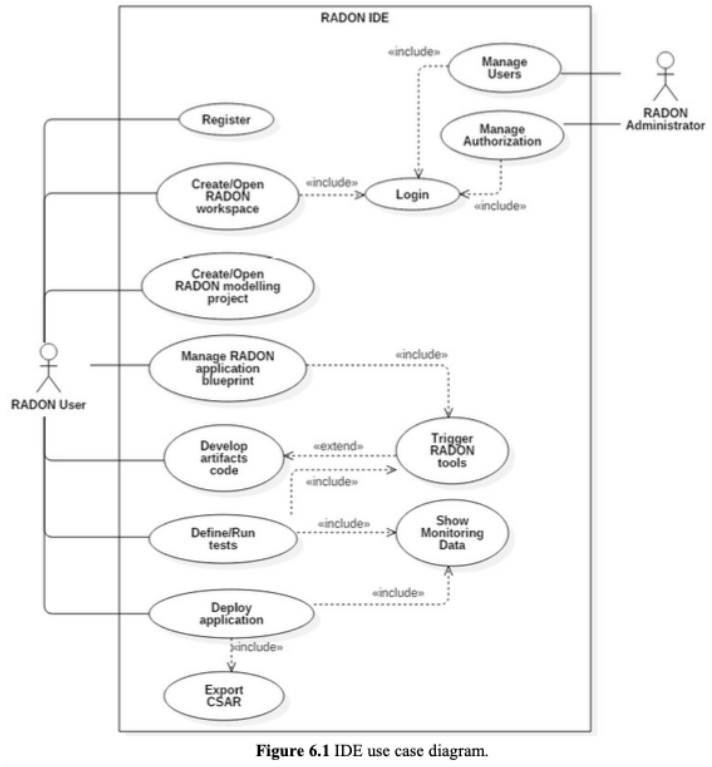

Figure 12: Use Case Diagram for Overall execution of RADON workflow[28][p.49]

Now I discuss different components of RADON in detail and use case diagrams in some specific cases :

1. Eclipse Che: [30]

   The Entry Point of the RADON is Eclipse Che [30][p.30]. Via this Integrated Development Environment (IDE), RADON users register into RADON and can see the main dashboard [30][p.30].

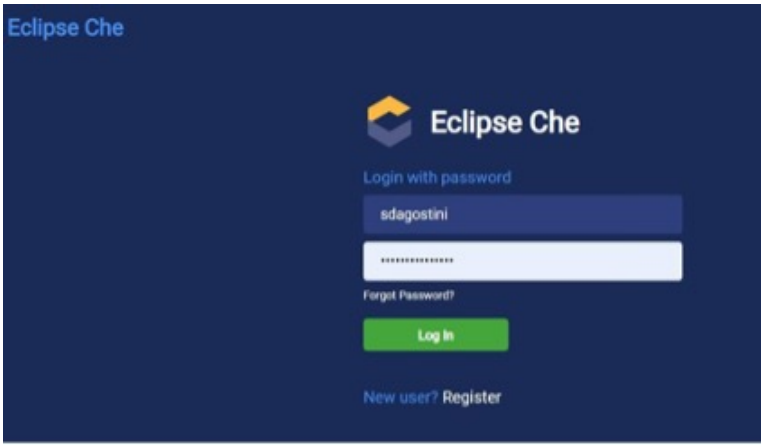

Figure 13: Eclipse Che Login [30][p.30]

On the main dashboard, the RADON user starts a new workspace [30][p.31]. When the workspace is created, the user gets access to RADON particles like Graphical

[30]https://radon-h2020.eu/wp-content/uploads/2020/07/D2.6-RADON-integrated-framework-I.pdf

Modelling Tool(GMT), Verification Tool(VT), Decomposition Tool(DT) and Defect Prediction Tool(DPT) [30][p.31].

2. Graphical Modeling Tool (GMT)

   Using GMT, the user can create a new application model or can reuse an existing one [30][p.32].

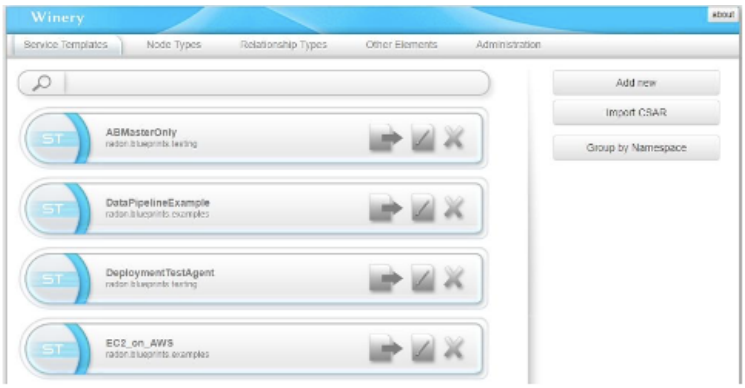

Figure 14: GMT View [30] [p.32]

Use case diagram for GMT

This diagram presents what RADON users can do with the GMT tool in RADON IDE.

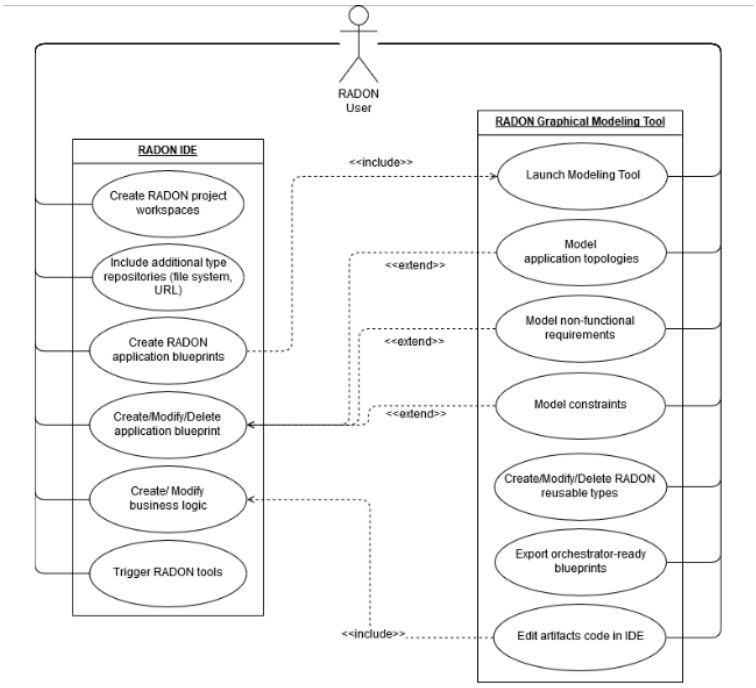

Figure 15: Use Case diagram for RADON and GMT [28] [p.58]

Eclipse Che represents the IDE for RADON and Eclipse Winery represents the IDE for GMT [28][p.57].

When a user opens a workspace in Eclipse Che, it automatically opens a window in Eclipse Winery [28][p.58]. In the window in Eclipse Winery, the user can create the

blueprint for the application model [28][p.58]., can drag and drop different components and design the model [28][p.58], and can define different requirements and properties of the component in CDL specification.

Running Example for GMT

Figure 16: GMT [28][p.34]

In this example, the user selects the AWS S3 bucket for storing the uploaded image [28][p.35]. Then select the AWS lambda function to process the image (create thumbnail) [28][p.35]. After processing it again, save it in another S3 bucket [28][p.35].

Whenever the first S3 bucket has an image uploaded, it will call the AWS lambda function for processing [28][p.35].

3. (Verification Tool (VT))

   To describe security policies and architectural decisions of an application, RADON uses a specific language called CDL [28][p.30]. RADON verifies if the CDL is correct or not using a Verification Tool [28][p.30]. The verification results are shown in the RADON Verification Tool output panel.

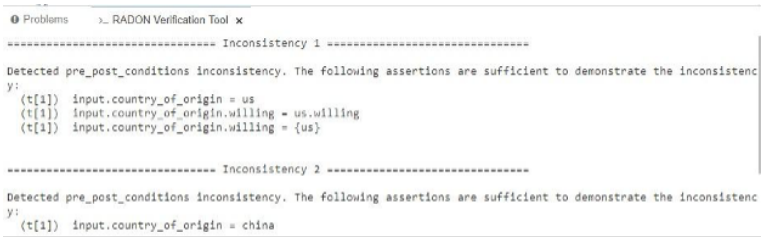

Figure 17: RADON Verification Tool [30][p.35]

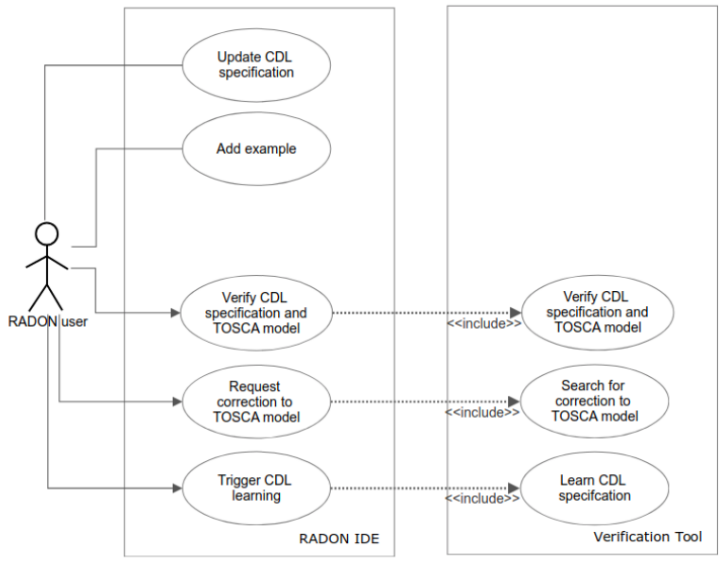

Figure 18: Use Case Diagram for RADON Verification Tool [28][p.56]

Running Example for CDL and Verification Tool

In this example, I create a constraint - if the thumbnail is generated in the European Region, it can be stored in the European Region [28][p.30]. But if it is generated in another region (US), then it can not be stored in a bucket from European Region country [28][p.30].

In the first sequence, the user creates the thumbnail in the US but tries to store it in the EU. So it will be an invalid sequence and the Verification tool will detect it as an error [28][p.31]. For the second sequence, the user creates the thumbnail in the UK and tries to store it in EU [28][p.31]. So it will be a valid sequence and the verification tool will not detect errors [28][p.31].

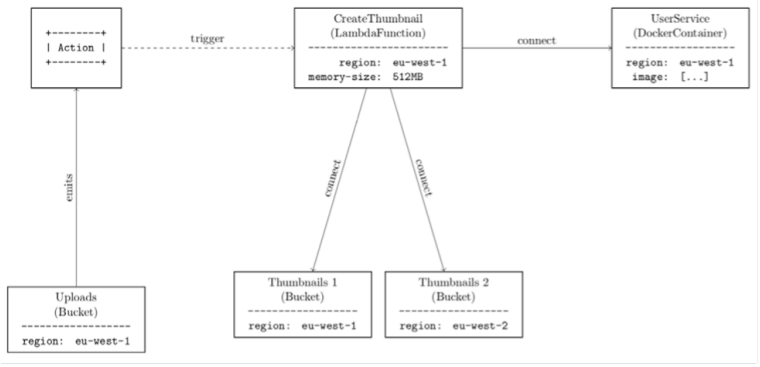

Figure 19: Verification [28] [p.31]

The verification tool can also suggest a solution in the case of invalid sequences [28][p.31]. Like, it might suggest I need to add a bucket from the US if I want to store the thumbnail [28][p.31].

4. (Decomposition Tool (DT))

The Decomposition Tool in the RADON IDE optimizes the deployment of a RADON model [30][p.36].

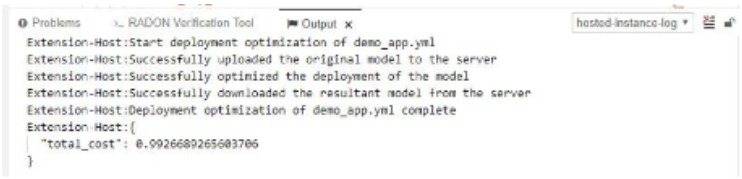

Figure 20: RADON Decomposition Tool [30][p.36]

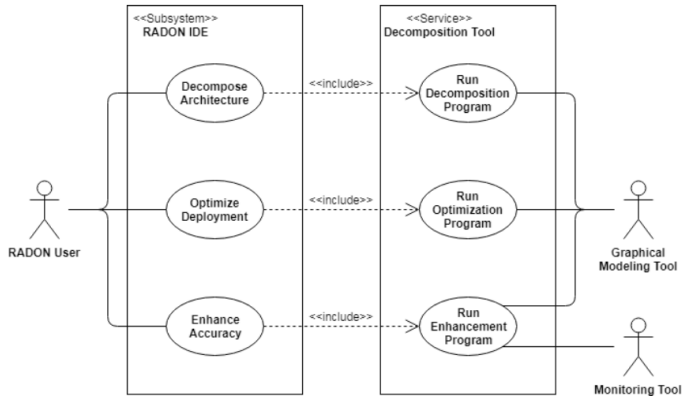

Figure 21: Use Case Diagram for RADON Decomposition Tool [28][p.51]

RADON users can divide a single model of RADON into small models using this decompose feature [28][p.51]. It divides the RADON model into many Serverless functions with its storage [28][p.51].RADON users can optimize the original model to a new one so that the cost of deployment is minimized [28][p.51]. It does it with the optimized deployment option [28][p.51].RADON can increase the accuracy of deployment using monitoring tools [28][p.51].

5. (Defect Prediction Tool (DPT))

   The Defect Prediction Tool checks defects in an IaC(Infrastructure as Code) script [30][p.36]. The result is some metrics that demonstrate the defect proneness of the system [30][p.37].

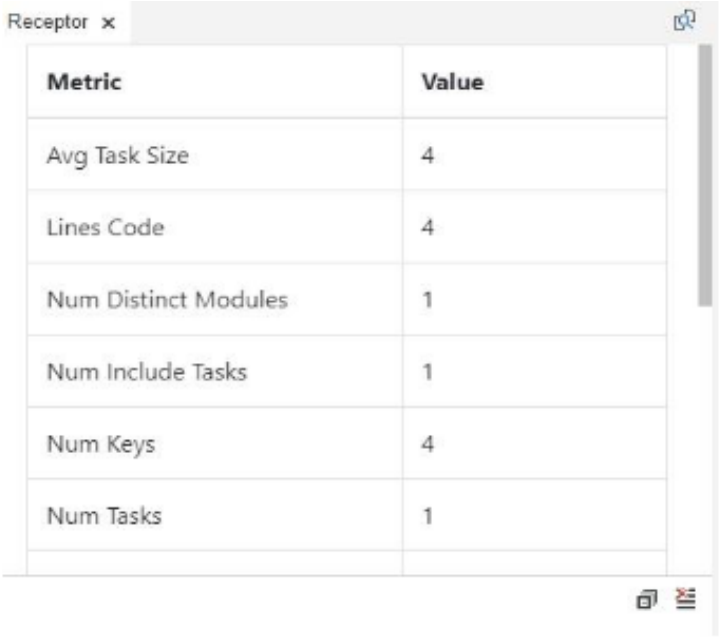

Figure 22: RADON Defect Prediction Tool [30] [p.37]

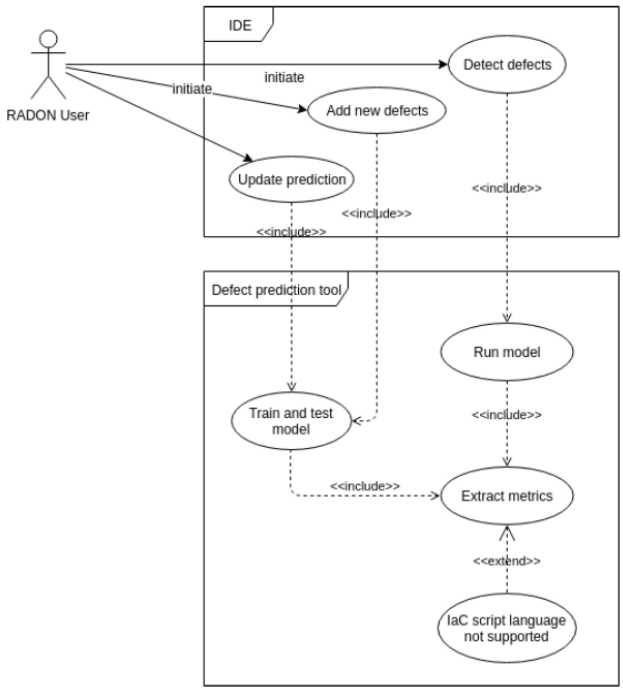

Figure 23: Use Case Diagram for RADON Defect Prediction Tool [28] [p.54]

Defect Prediction Tool verifies the IaC script and returns the class where the defect is found [28][p.54]. The defect prediction tool uses machine learning to enhance its

capability [28][p.54]. It merges the new data with the training data and then trains itself with the whole data [28] [p.55].

Running Example for Defect Prediction Tool

The Defect prediction Tool identifies code smells or error-prone code in IaC scripts using machine learning [28][p.38].

In my example, the thumbnails are generated by the AWS lambda function and then stored in the AWS S3 bucket [28][p.38]. During this process, the credentials for AWS S3 can be saved in the application.yaml file, to ensure login to AWS [28][p.38]. Which is a common practice but can expose to security threats [28][p.38].

The defect prediction tool will find out these lines and will mark them [28][p.38].

```
node_templates:
    uploads:
      type: radon.nodes.ObjectStorage
      properties:
        name: opentosca-uploads-1535712682
        aws_access_key: "AKIAJ2ASDF435PWKTC5RA"
        aws_secret_access_key: "AsdEfG1iF2T0AsEdFgQ37HhGLuERQjhPAC2vS"
      requirements:
        ...
```

Figure 24: Defect Prediction [28][p.38]

6. Deploy the application

   The RADON user can deploy the application after modelling [30][p.37].

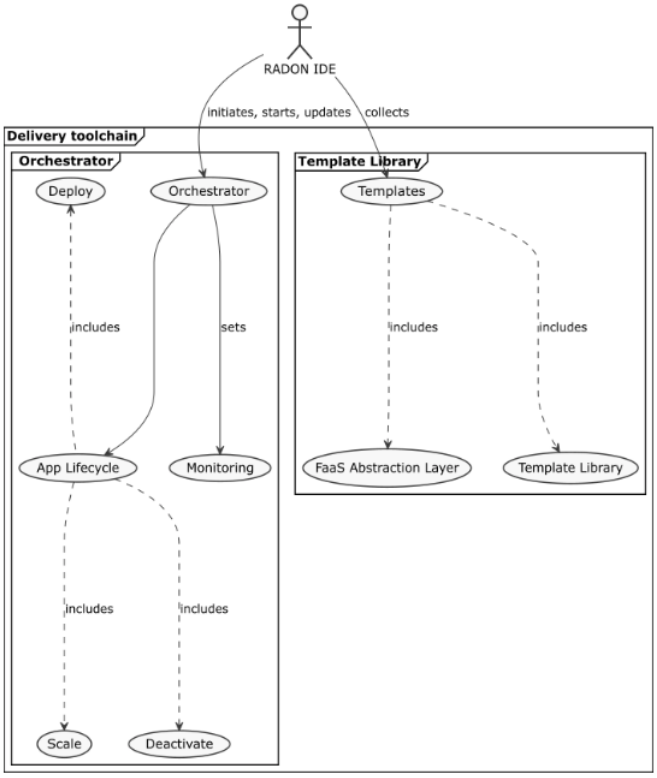

Figure 25: Use Case diagram for RADON Deployment Orchestrator Tool [28] [p.60-61]

RADON IDE manages the Orchestrator and Templates. The Orchestrator manages the Application lifecycle and also monitors it. The application Lifecycle includes Deploy, Scale and Deactivate phases. Templates include FaaS Abstraction Layer and Template Library.

After the user plan and designs the application, RADON IDE collects different templates from the template library [28][p.60-61]. The application made by the user is also stored in the template library [28][p.60-61]. Then, RADON IDE adds different FaaS abstraction layer with the previously collected templates, so that the same application can be hosted by different FaaS platforms like AWS lambda, Azure and OpenFaaS [28][p.60-61]. The orchestrator deploys it to the targeted customer [28][p.60-61]. This is the starting point of the Application lifecycle [28][p.60-61]. The lifecycle ends when a user selects to deactivate [28][p.60-61]. In this mode, it destroys all instances of the application [28][p.60-61].

7. CI/CD

   The main CI/CD provider for RADON is Jenkins [27][p.24]. Another one is CircleCI [27][p.24].

8. Testing

   RADON uses a test-driven approach with many unit tests [27][p.27]. These unit tests validate the set of functionalities implemented by the RADON tools [27][p.27]. There is also code coverage, which shows how much of the code is covered by the tests [27][p.27].

At first, tests are defined to test different functionalities by RADON [27][p.27].

Then the tests are hosted in the CI/CD pipelines so that they are automatically executed when a new update in the code base [27][p.27].

Running Example for Testing Tool

The application model of the software has requirements that need to be satisfied for proper execution [28][p.33]. Like the uploads function in the testing tool, has an upper limit of the average response time of 12 seconds.

If this is satisfied via unit tests, then the process continues [28][p.33].

```
policies:
    - uploads_average_response_time:
        type: radon.policies.Performance.AverageResponseTime
        properties:
            upper_bound: 12 s
        targets:

- uploads

    - uploads_workload:
        type: radon.private.Workload.OpenWorkload
        properties:
            interarrival_time_distribution:
                mean: 5 s
                scv: 1.0
        targets:

- uploads
```

Figure 26: Testing for the application [28][p.33]

9. Continuous Testing

   This use case diagram shows how the RADON IDE interacts with the testing tool IDE.

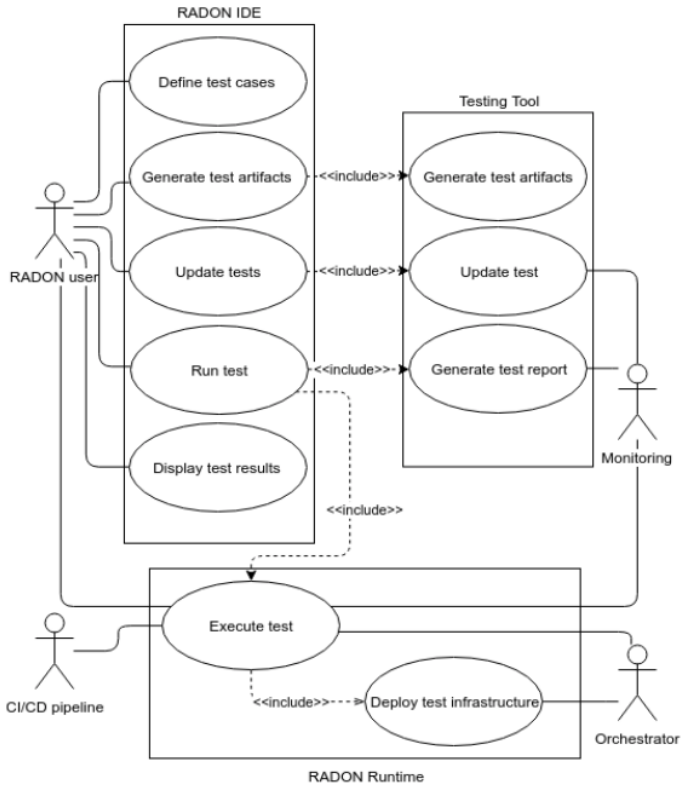

Figure 27: Use Case Diagram for RADON Continuous Testing Tool [28] [p.52-53]

With this tool, users can manage different stages of tests and display test results [28] [p.52-53].

In Continuous Testing, At first, the Monitoring tool monitors the test. After the test is finished, monitoring data can update the test. The test can be run automatically in CI/CD pipeline [28] [p.52-53].

Ambient Assisted Living (SARA) An Usecase for RADON [CAVDH+20, p.85]

SARA assists in taking care of the health of the old people at home [CAVDH+20, p.85]. It uses robots and IoT technologies [CAVDH+20, p.85]. It adopts the RADON DevOps Model [CAVDH+20, p.85]. It consists of these devices:

1. Smart phone - mobility detection and risk management of the patient [CAVDH+20, p.85].
2. Robotic Rollator - a wheeled walking frame that can identify the patient and can monitor her/his monitoring and navigation [CAVDH+20, p.85].
3. Robotic Assistant - to give the patient updates about her/his health status and upcoming trips to doctors [CAVDH+20, p.85].
4. Environment Gateway- track the patient using smart devices [CAVDH+20, p.85].

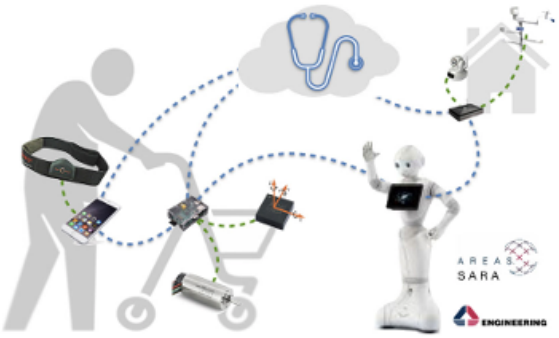

Figure 28: Ambient Assisted Living [CAVDH+20]

SARA has two distinct features of RADON.

1. SARA uses modeling tools of RADON [28][p.62].
2. SARA uses OpenFaaS with IoT applications [CAVDH+20, p.85].

SARA tracks the patient's movement using remote gait analysis [26]. Remote gait analysis continuously collects the sample of patient tracking [26]. The sample is called Gait Sample [26]. Then it sends to the Gait Server[26]. Gait Server is available as a service [26]. And Gait Sample is the distance of legs from the scanner. It then cluster this data as a time series so that the practitioner can assess the patient [26]. The gait sampler sample, sequences and encodes the data and sends it to the gait server [26].

Remote gait analysis uses OpenFaaS and BLE devices. It uses OpenFaaS to host Serverless functions that help track the patient [26]. In SARA, the developers factor out the stateless part of the gait sampler and make it a function in the OpenFaaS [26].

Functions in the OpenFaaS SARA model [26]

1. Create time series cluster

2. Encode-time series
3. Start-a-session
4. Add-gait-sample
5. Stop-ga-session

Adding DevOps features to my TEXT-ADD system

I have tried to upgrade my previously built TEXT-ADD system with some features in DevOps. DevOps uses microservice architecture. My TEXT-ADD system only uses functions or microservices. DevOps uses Continuous Deployment. My TEXT-ADD system uses 3 commands - build, push, and deploy to build the system. So, I upgraded it, so that it uses only one command - up, to deploy the system. Here is the video demonstration [31]. DevOps uses Continuous Integration. I tried but could not achieve Continuous Integration for the TEXT-ADD system. But I did some research on how one can achieve it.

Here, I describe it in detail.

1. With OpenFaaS, CI/CD is possible. This video [32] shows how to integrate GitLab with OpenFaaS. I tried to implement this. But some roadblock I faced is that image of the function should be hosted in the docker hub. But I developed the image on a local docker desktop. So, I need to push it every time to the docker hub before using this CI/CD. Also, the code developed in the local VScode needs to be pasted manually in GitLab.
2. CI/CD is very easily implemented for the AWS Lambda function. As the AWS console has a built-in pipeline for CI/CD, it has some limitations as it will pay charges after a certain number of function calls. [33].

My System vs RADON

1. My system is developed to propose a cloud function lifecycle whereas RADON is designed to commercially develop FaaS applications for multiple vendors. So, RADON has many more features compared to my developed model.

8 Summary and Final Words

Findings

The precise answer to the research questions is summarised below -

[31] https://youtu.be/8hfCwS1bG5Y
[32] https://www.youtube.com/watch?v=WYyonInGNsw
[33] https://www.youtube.com/watch?v=mIky1niHGdY

1. How does FaaS differ from other cloud computing models?

   Solution: The main difference proposed is that FaaS host small functions like microservices whereas other models host mainly complete system.

2. How can use a tool to monitor a FaaS system and develop a lifecycle for cloud functions?

   Solution: I have used Prometheus to monitor cloud functions in the FaaS system. Prometheus gives me a graph representing the status of a function in different time frames. Based on that, one can decide if a function is working correctly or not. if it does not work correctly, then I can take two paths. can go to the update phase or can go to the phase-out phase. If I think changing the function is economically feasible, I go to the update phase. If I find out it is not economically feasible, I go to the phase-out phase. Thus, I have developed different phases of the lifecycle for cloud functions.

3. How does a cloud function lifecycle differ from a traditional lifecycle?

   Solution: The cloud function developed is almost identical to the waterfall and staged model but does not match with other agile models like scrum and Xtreme. Because agile models are designed for enterprise applications and I developed my model by experimenting with a simple solo academic project.

4. Discuss a FaaS DevOps system in Detail and How to evolve my system to a DevOps system?

   Solution: I discussed a FaaS DevOps system named RADON in this thesis. This model ensures FaaS systems are developed in a DevOps way so that they include Continuous Integration, Continuous Deployment and other DevOps principles in the development process.

   To evolve my model (TEXT-ADD) to a DevOps model, I need to ensure the model has Continuous Integration, Continuous Deployment, Microservices, Continuous Testing, Continuous Monitoring, Collaborative Development, Continuous Customer Feedback etc. My developed FaaS monitoring system already uses micro-services, because the cloud function is regarded as a micro-service. I also used single commands for deployment, thus ensuring faster deployment of my system. For Continuous Integration, I discussed how it can be achieved using Gitlab. Continuous Testing can be integrated before Continuous integration. Like running my tests before Continuous Integration. Then changing the tests based on monitoring the old data. Continuous Monitoring can be done using the Prometheus graph as I did. Collaborative Development can be added when more than one worker works on the same project using Gitlab. Continuous Customer Feedback can be gathered using software like slack.

Future Development

In future development, one could try to achieve the Continuous Integration, Continuous Testing, Collaborative Development and Continuous Customer Feedback feature for the TEXT-ADD system.

Final Words

In this research, I discussed many open questions about FaaS and cloud functions. First, I discussed some related research concerning this thesis topic. Then discussed my specific research questions and my methodology to do this research.

In my first research issue, I tried to differentiate FaaS from other cloud computing models.

In the second question, I developed a system named TEXT-ADD to monitor FaaS and proposed a lifecycle for cloud function based on my monitoring using Prometheus. Then argued how the graph representation from Prometheus helps me decide on developing different phases of the FaaS lifecycle.

Then I differentiate my developed system from traditional software development systems and DevOps.

Then I studied a FaaS system for DevOps named RADON. Then implemented Continuous Deployment in my TEXT-ADD system. In the end, discussed how to make it more like DevOps by adding other features to TEXT-ADD.

Lastly, I discuss my findings in this thesis and future development ideas.

FaaS is one of the most promising cloud computing models and will become more popular day by day. This research makes a significant effort to clarify some unknown aspects of FaaS systems. This research can help people who are interested in FaaS, want to develop FaaS systems or want to do research on FaaS in general.

References

[AJFOG17] Lucas F Albuquerque Jr, Felipe Silva Ferraz, RF Oliveira, and SM Galdino. Function-as-a-service x platform-as-a-service: Towards a comparative study on faas and paas. In *ICSEA*, pages 206–212, 2017.

[BBK+19] Anirban Bhattacharjee, Yogesh Barve, Shweta Khare, Shunxing Bao, Aniruddha Gokhale, and Thomas Damiano. Stratum: A serverless framework for the lifecycle management of machine learning-based data analytics tasks. In *2019 USENIX Conference on Operational Machine Learning (OpML 19)*, pages 59–61, 2019.

[Bec99] Kent Beck. Embracing change with extreme programming. *Computer*, 32(10):70–77, 1999.

[CAVDH+20] Giuliano Casale, Matej Artač, W-J Van Den Heuvel, André van Hoorn, Pelle Jakovits, Frank Leymann, Mike Long, Vasilis Papanikolaou, Domenico Presenza, Alessandra Russo, et al. Radon: rational decomposition and orchestration for serverless computing. *SICS Software-Intensive Cyber-Physical Systems*, 35(1):77–87, 2020.

[dlT16] Cesar de la Torre. Containerized docker application lifecycle with microsoft platform and tools. 2016.

[EGHS16] Christof Ebert, Gorka Gallardo, Josune Hernantes, and Nicolas Serrano. Devops. *Ieee Software*, 33(3):94–100, 2016.

[EK20] Shokhista Ergasheva and Artem Kruglov. Software development life cycle early phases and quality metrics: A systematic literature review. In *Journal of Physics: Conference Series*, volume 1694, page 012007. IOP Publishing, 2020.

[Gok21] Mayank Gokarna. Devops phases across software development lifecycle. TechRxiv, 2921.

[HY10] Rolf Harms and Michael Yamartino. The economics of the cloud. *Microsoft whitepaper, Microsoft Corporation*, 3:157, 2010.

[JSSS+19] Eric Jonas, Johann Schleier-Smith, Vikram Sreekanti, Chia-Che Tsai, Anurag Khandelwal, Qifan Pu, Vaishaal Shankar, Joao Carreira, Karl Krauth, Neeraja Yadwadkar, et al. Cloud programming simplified: A berkeley view on serverless computing. *arXiv preprint arXiv:1902.03383*, 2019.

[Kij18] Kijak. Challenges for scheduling scientific workflows on cloud functions. In *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, pages 460–467. IEEE, 2018.

[Kol19] Stefan Kolb. *On the Portability of Applications in Platform as a Service*, volume 34. University of Bamberg Press, 2019.

[MFGZ17] Maciej Malawski, Kamil Figiela, Adam Gajek, and Adam Zima. Benchmarking heterogeneous cloud functions. In *European Conference on Parallel Processing*, pages 415–426. Springer, 2017.

[PB20] Tobias Pfandzelter and David Bermbach. tinyfaas: A lightweight faas platform for edge environments. In *2020 IEEE International Conference on Fog Computing (ICFC)*, pages 17–24. IEEE, 2020.

[RB00] Václav T Rajlich and Keith H Bennett. A staged model for the software life cycle. *Computer*, 33(7):66–71, 2000.

[RJ00] Linda Rising and Norman S Janoff. The scrum software development process for small teams. *IEEE software*, 17(4):26–32, 2000.

[RR14] Dimpi Rani and Rajiv Kumar Ranjan. A comparative study of saas, paas and iaas in cloud computing. *International Journal of Advanced Research in Computer Science and Software Engineering*, 4(6), 2014.

[RVD+10] PK Ragunath, S Velmourougan, P Davachelvan, S Kayalvizhi, and R Ravimohan. Evolving a new model (sdlc model-2010) for software development life cycle (sdlc). *International Journal of Computer Science and Network Security*, 10(1):112–119, 2010.

[Spi17] Josef Spillner. Snafu: Function-as-a-service (faas) runtime design and implementation. *arXiv preprint arXiv:1703.07562*, 2017.

[Spi20] Josef Spillner. Resource management for cloud functions with memory tracing, profiling and autotuning. In *Proceedings of the 2020 Sixth International Workshop on Serverless Computing*, pages 13–18, 2020.

[VEIST17] Erwin Van Eyk, Alexandru Iosup, Simon Seif, and Markus Thömmes. The spec cloud group's research vision on faas and serverless architectures. In *Proceedings of the 2nd International Workshop on Serverless Computing*, pages 1–4, 2017.

[VETT+18] Erwin Van Eyk, Lucian Toader, Sacheendra Talluri, Laurens Versluis, Alexandru Uță, and Alexandru Iosup. Serverless is more: From paas to present cloud computing. *IEEE Internet Computing*, 22(5):8–17, 2018.

[WLSW21] Frederik Wulf, Tobias Lindner, Susanne Strahringer, and Markus Westner. Iaas, paas, or saas? the why of cloud computing delivery model selection: Vignettes on the post-adoption of cloud computing. In *Proceedings of the 54th Hawaii International Conference on System Sciences, 2021*, pages 6285–6294, 2021.

[YS15] Xugang Yin and Yanlei Shang. Research and implementation paas platform based on docker. In *2015 4th National Conference on Electrical, Electronics and Computer Engineering*, pages 668–673. Atlantis Press, 2015.

In accordance with § 9 Para. 12 APO, I hereby declare that I wrote the preceding master's thesis independently and did not use any sources or aids other than those indicated. Furthermore, I declare that the digital version corresponds without exception in content and wording to the printed copy of the master's thesis and that I am aware that this digital version may be subjected to a software-aided, anonymised plagiarism inspection.

Bamberg, February 10, 2023 Muhammad Faisal Imran Khan