A Report submitted in partial fulfilment of the regulations governing the award of the Degree of BSc (Honors) Computer Science with Web Development at the University of Northumbria at Newcastle

­­

Project Report

Designing and developing a microservice architecture for an IT help desk system

Nicholas Coyles

2020/21

1. **Abstract:**

The focus of this project revolves around the subject of software architecture. Particularly, the microservice architecture and its sudden rise in popularity among the world of software engineering. Investigating the benefits over existing widely used architectures, like the monolithic architecture for example. With reference to the literature discussing the strong proponents of the concept and the reason for its recent prevail over unfashionable architectures. Exploring the significance of the architecture and what advantages they exhibit. Nevertheless, presenting the opposing argument of the complexities and drawbacks that accompany microservices to figure out what has caused this surge in industry acclaim.

The objectives of the project, establish the overall scope. Firstly, to investigating the state of the current literature and congregating the current and existing methodologies and patterns. For both design and development of a microservice architecture. Leading into the second objective, which takes understanding from the literature to guide the design and development of an IT Help Desk system.

The motivation behind this choice in system, is because the requirement for a Help Desk is well defined and established. Therefore, it can be used as a vehicle to demonstrate the process of development of the architecture. Once the requirements for the system were clarified, systems diagrams were created to break down the problem at hand, detailing each of the services deemed necessary to meet the requirements.

An agile development was applied to carry out the rapid development, using each service as an iteration cycle. There was no prior knowledge in technology used for development ASP .NET Core. It was used because the powerful tool such as entity framework that NET can take advantage of. As well as the documentation provided by Microsoft to aid the learning curve. Per development cycle unit testing took place testing the functionality, to minimise errors and improve the overall product.

There were multiple instances where a step back was necessary to re-evaluate the system, to change the design of the services. This was due to the prementioned lack of clarity in the literature, which partially created frustration in the findings across the project. Accompanied with the complexity of what was set out to achieve and the steep learning curve, and limited time a great deal of time and effort was expended to showcase a limited system, but the teaching and evaluation can provide a utilitarian insight to the struggle this project presented. the conclusion is that microservices are still in their infancy and it will take time before there is a consensus among the software engineering community for the best design patterns and how to implement them. Finally, in reflection microservice development is complex, and would take more time than was available to become more familiar with the language to then design and develop to higher standard.

**List of Contents:**

1. Abstract
2. Introduction
   1. Background
   2. Aims and Objectives
   3. Subject of work
   4. Plan of work

Analysis Chapter

1. **Introduction:**

**2.1) Background**

Software engineering is an ever-evolving field, where technologies are continuously being developed and integrated with the goals of solving the problems with previously technologies to high technical standard.

We have seen examples of this transition in the past when the software engineering standards shifted from procedural to object orientated programming. A similar transformation is currently being seen with the industrial norms moving from a monolith architecture, towards a microservice which is in a fact a hybrid between both a procedural and object orientation. These movements are caused by client needs and is comparable to the wider world. In that fact that everything is changing to adapt to the modern needs of people, and this no different to what is happening in software architecture. New methodologies are implements to better and improve the output for the client. Methodologies underpin the development of systems. Although methodologies may be discussed in this project, the focus is the software architecture and the effects the architecture bears on the resulting product.

Object orientation provides many programming benefits with the ability to abstract data, a form of encapsulation where only unscary details are hidden only exposing the important operations to the user (Harrison, 1993). Each class can have their own capabilities which may be completely different from one another, but all respond to the same protocol operations (Stefik, M. and Bobrow, 1985) showing a rich domain, not to be mistaken for an anaemic domain which is style of procedural programming, the opposite to what object orientation is trying to achieve (Fowler, 2003)

Object oriented models, aim to be loose coupled meaning that a class should have one functionality and do to very well, as if a class has more than one responsibility it can become tightly coupled. Where a change to one responsibility will inhibit the ability of all the other classes that depend on it, leading to a fragile design (Martin, 2007).

When objects become loosely coupled in this way, the rich domain is essentially lost due to the extreme complexity and is now a hindrance on the system. Systems gain a large codebase and becomes anaemic in fashion and hard to maintain. Constantly having to maintain a monolithic architecture makes it difficult to keep up with development (Kalske, 2017)

Even in the early stages of development this can become an issue it is difficult for developers to keep insight of all the system components (Fritzsch, 2018). Which makes it challenging to adapt an agile development because now the monolith requires extensive though in the design phase. As well as this there is a knock-on effect of a system becoming technology locked because the system is too complex to easy transition to a new technology and would be very cumbersome to move to a new technology and the system is no longer scalable (Fritzsch, 2018).

The pre-existing issue with the monolithic architecture is forcing an evolution of technologies and methods. Introducing the microservice architecture

It is important to note that a microservice breaks a system down into several distinct services, separated based on business capabilities of the system. Resulting in more flexible scaling and deployment because runs in its own process and can evolve independently (Jin, 2018) without having an effecting the rest of the system (Rademacher, 2017). Which allows of a major benefit of a service being able to incorporate the most suitable technology for the job (Taibi, 2018) amending the shortcoming of the monolithic architecture. Also, these services are typically responsible for their own functionalities (Jin, 2018). Which presents the question, how far has the microservice come from outdated methods such as procedural programming, because service being self-contained and depending on its functionality is an example of the anaemic domain model. From the findings it seems that the evolution of software architecture has progressed full circle and the methods that being used are modern adaptations of older techniques used to meet new the demands of clients.

However, although there are advantages of the architecture addressed. There is not a great deal of depth in the literature relevant to the practical decomposition of a system into its business capabilities. There is lots of theory but not much in practise. The purpose of the project is to address this and analysis the literature searching for the current best practises for designing a system using the architecture. As well as encapsulating the topic of microservices assessing where its place is in the future of software engineering.

The project idea steamed from a meeting with the supervisor where the discussion of current software engineering trends leads to the microservices topic. The challenge plus the relevancy in the current day ultimately led to this project being undertaken. The findings demonstrated a lack of both depth and clarity in the current methods in the literature.

**2.2) Project aims:**

1. “To investigate and analyse the use of different architectures, and the rise of microservices architecture.”
2. “To design and develop an IT Help Desk System built upon a microservice architecture.”

**Objectives:**

1. Review the literature surrounding microservices, comparing to other architectures and include some real-world examples of microservice systems. As well as common patterns used in their development.

The literature will be used through the project, to provide potential solutions to problems in the development of the project. Reviewing the literature in this way creates a useful discussion of different possibilities when it comes to facing problems.

2. Establish requirements to address the points of focus during each stage of the project.

Requirements capture discusses both functional and non-functional requirements.

3. Create an overall design of the system establishing what microservices are needed and how they will integrate.

A systems diagram that shows how each service plays a role in the system and demonstrates the inter connection between services and databases

4. Create suitable design documentation for each individual element in the system.

Design documentation is the basic groundwork for development and implementation chapters. Discussing all the functionalities of each part of the system.

5. Develop and implement the IT Help Service System with some full functioning applications.

Development and implantation of system, commenting on the multiple solutions to complete the problem and discuss why certain choices are taken. Also discussing how problems are solves.

6. Create system documentation including relevant test plans and cases.

Using the use cases to create test plans using scenarios for users’ interactions with the system.

7. Test the system using the documentation as a reference.

Carry out testing of the systems functionality, using the test plans just created as guide, taking note of both successful and unsuccessful tests.

8. Analysis the result of the tests/evaluation.

Summarising the results of the tests, evaluating how the system can be improved and what can happen to make this take place.

9. Evaluate the system.

Evaluate the product created. Including strengths, weaknesses, objects, alternate methods and how what could be done to improve upon the weaknesses

10. Evaluate the project.

Evaluate the project using the requirement specification, to measure success.

11. Evaluate the project process, including a self-reflection.

Evaluate the product process, looking at the initial project plan comparing it with what happened during the product process.

12. Conclude the project and give recommendations.

Finishing the project with an overall conclusion and give recommendations to future work.

**2.3) Subject of work:**

The area of work that is the whole project revolves around is that of the microservices architecture. The first section of the project is the critical research into the current trends in microservices including development and design techniques and the general perception within the software engineering community of microservices. These areas are important to conduct research into because the next main section is the development of an IT Help Desk system that is crated using the microservice architecture and the patterns discovered through the research and how patterns can influence the development of such a project. The scope is to stay on these areas so the resulting product can demonstrate a functioning microservice architecture, as well as presenting several different methods that can be used to meet the goals of the project. The only planned limitation of the project is the timescale, as the methods and techniques used within the project are a relatively new complex concept tied with no prior knowledge from the author the time might cause a potential limitation for implementation all the desired system features.

**2.4) Plan of work:**

The first stage is the analysis including the literature review, research into existing IT Help desk applications to form the requirements specification. Research into tools to develop microservices and selecting the most appropriate technologies to conduct the Synthesis which is the next chapter. The synthesis will include two main parts, firstly the design of the system where systems diagrams and other design documentation will be created from the finding from the literature review. Which will then be used in the design and development stage, the second part of the synthesis, where the product itself will be created along side the subsequent documentation of the process. Once the development has concluding testing will begin where the functionality of the system will be tested using a test plan and methodology where errors can be filtered out, finishing with a final test of the overall system. The project ends with an evaluation of the final product, project process and a closing conclusion including future recommendations.

**Analysis Chapter:**

1. **Literature Review:**

This literature review examines what is a microservice, along with the characteristics of the architecture and why it is becoming the successor to the monolithic architecture. With this work looking into the reasons why the rich domain monolithic systems have fallen out of favour as of late in the development community. In contrast, why microservices are seen as the solution to the issues encountered with traditional object-oriented software development. Illustrating both sides of the argument for the benefits of drawbacks of such an emerging technology. The scope of the review is the focus of design and development of a microservice, examining existing literature surrounding the topic and developing an understanding of the current state of accepted techniques and approaches within this area of research. The analysis of the literature will be needed to form an effective solution using microservices.

* 1. **Microservices**

In recent years, there has been a shift in the methods within system architecture development. One of the big causes being the increasing popularity of cloud computing. Microservices is the newest approach towards systems architecture in software development. It can be said that microservice architecture is a branch of service-oriented architecture (SAO) where the fundamentals of SAO have been built upon with a modern twist to create a new technique. With that aim of combatting the shortcoming of traditional software architecture, in particular the monolithic architecture. Companies are already taking steps to adopt microservices by modernizing their legacy monolith applications, refactoring old systems into microservice. This includes companies like Amazon, Netflix, and Uber (Larrucea, X, 2018).

The term “microservice” is relatively new and was first defined in 2012. A year after it was first discussed at a workshop, a group of software architects. They branded it with the name microservices to describe what was seen by participants as a common architectural style (Fowler, 2014). A vague definition has been created by the likes of Fowler and Newman throughout the years and microservices now has a more established definition. A small, autonomous services that works together to create a single application with each service having the capability of running in its own process and communicating with lightweight loosely coupled mechanisms (Fowler, 2014) (S. Newman, 2015). The functionality of Microservices however isn’t a completely new idea, Mike Amundsen believes that the desired properties that are being seen currently with microservices isn’t particularly new. In fact, these properties can be tied back to the patterns that were seen in SOA. Properties you wish to design for will structure the constraints for the implementation of a system and in ways microservices are the best-practice approach for realizing SOA (Pautasso, 2017). Which brings back the claim made in the introduction that what we are seeing now is the same as what was seen in the past with the development of new technologies taking the best approach from previously existing techniques, and adapting them into a new concept, effectively going full circle. However, it has been put forward by Daya, S and co that the comparison is more complex and somewhat unfair than what has been stated, because the microservice architecture has never actually claimed to be a modern approach for building distributed systems (Daya, S, 2016). In fact, SOA and Microservices have different aims, the prior aims on reusability, whereas the latter’s main goal is manageability (Daya, S, 2016) The argument for the comparison is valid because even though the aims are different fundamentally, they build off the same premises developing service-based applications (Vural, 2017).

The notion of Microservices being a small autonomous application can be better described as an application which has been divided into services and can be tested, deployed, and scaled independently of each other. Each has a single responsibility in the global scheme of the application (Thönes, 2015). As the application has the services function individually in this way, any testing, changes, and deployment made to a service will not have any effect on the other survives keeping the integrity of the system intact. Resulting independence allows a great increase to the scalability, manageability, and updatability of the microservice application (Bao, 2019). The break down into multiple services revolves around the small business capabilities or needs of the application (Daya, S, 2016).

To better understand how a microservice is broken into services can be visually presented in Figure 1 by (Richardson, 2018) In this example of an e-commerce context, the application's functionality has been divided into separate responsibilities. Such as a UI interface communicating directly to the web connected by a REST API to the other services. The other services are the business capabilities, account, inventory, and shipping. This correctly demonstrates the decomposition of services and how the design allows them to interact together via an API.

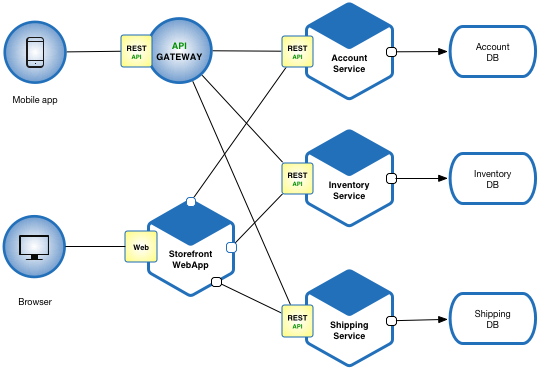


Figure 1: Pattern: Microservice Architecture (Richardson, 2018)

* 1. **Microservices and Monolithic**

Comparing Microservice to Monolithic architecture, there are some major differences in the characteristics and structure of both architectures. Uncovering these differences may support the switch in technologies over the recent years. Monolith architecture is the traditional way to structure software is the traditional way for software development, which used to be employed by world-famous Internet services such as Netflix, Amazon, and eBay, and it advocates encapsulating all functions in one single application. Less complicated monolithic applications have their own strengths, for instance, simple to develop, test and deploy (Chen, 2017). However, this is where the advantages end unfortunately. As this can be an effective software development structure for a very small piece of software it is inevitable that software will go through development changes, because applications are continuously developed, scaled (Villamizar, 2015). Leading to a point where there are more disadvantages than advantages. As it grows a monolith structure becomes more complex with a massive code set and becomes incomprehensible and impractical. The previous team that was managing a small set of code will have to use a tremendous amount of effort to maintain a system that is practically unmaintainable as a single error can cause whole system shutdowns. Resulting in delayed updates, fixes, downtime, and a generally slow system due to the size of the monolith (Chen, 2017). Now this is where microservices come in. As the new architecture Microservices, promises to address these issues with the monolithic structure (Fritzsch, 2018). Therefore, Microservices has many benefits over monolith because it is actively working to correct monoliths shortcomings. However, it cannot be all positive so, it is necessary to see if there are any drawbacks noted throughout the literature to have a fair discussion of the microservices architecture.

* 1. **Monolith first**

An interesting approach is a monolith first development cycle which is a current trend in the literature starting with a monolithic system and refactoring towards a microservice. It is less risky than completely redeveloping the whole system into microservices and easier to start is a monolith base (Larrucea,2018). Which could suggest that to create a microservice from scratch, potentially the developer could design the system in a monolith approach first, then refactor into a microservice. This viewpoint was also suggested by Martin Fowler he says an argument to start with a monolith then split to microservice when monolith becomes an issue is a reasonable approach (Lewis, J., and Fowler, 2014). As stated in the introduction a monolithic system can suffer from locked in technology, but even if a system is monolithic in nature refactoring even part of it to a microservice can unlock it from this lock and open it up to this idea of technology heterogeneity (add chapter number). For example, current books and papers on the issue, monolith to microservices evolution patterns to transform your monolith from Sam Newman (Newman, S., 2019) and from monolith to microservices (Gouigoux, 2017). The fact that this idea of moving from monolith to microservice is still one of the preferred ways to incorporate a microservice architecture could show that the technology is still immature in nature, that there is not as much literature referencing build from the group up. On the other hand, this literature would be very useful in industry where companies still have legacy monolith systems which seems to be the main target audience of this new architecture.

* 1. **Microservices benefits and concerns**

Monolith structures main issues arise from the difficulty with scaling the structure as this can involve an increasing code set which makes it difficult to make changes and update without causing system breaking errors. Whereas Microservices has the self-contained services approach, which Villamizar describes microservices as allowing for a more practical methodology for managing a large codebase on an application as each individual small codebase can be managed by a small independent team that manages the development of the code but also the deployment (Villamizar, 2015). Having small teams working individually inherently limits the scope for errors and bugs within the services. Due to the fact developers can directly test and develop the code in separate environments meaning the functionality is isolated from the rest of the system (Dragoni, 2017). Newman believes this characteristic of microservices can better align an organization, reducing the number of developers working on an individual codebase can find the best balance between team size and productivity (S. Newman, 2015). Effectively, creating a level of efficiency that was not possible previously with a single complex codebase in the monolithic structure.

On the other hand, it is impossible for microservices to be perfect in every aspect there are some potential concerns and challenges that should be addressed. Nadareishvili, has suggested that having a system compiled of multiple interconnected services can become a challenge because if part of a system has an error it may have an unforeseen impact on another part of the system making error handling more difficult (Nadareishvili, 2016) Following on from this an increase in inter communication add more ‘chatty’ communication between services can cause a poor performance for the end user as to meet a single business requirement communication between multiple services is necessary. As well as deciding how services are discovered, for example presented through an API gateway (Alshuqayran, 2016). Esposito believes that due to microservices unique complex characteristics, there is a need for research into technical limitations and security vulnerabilities (Esposito, 2016). The security issue is summarised by Sun, the fact that microservices are comprised of many small applications creates a complex network of activity. Which in turn makes monitoring and securing an application very challenging. Which is even more challenging when a system is deployed to the cloud as the admins only have limited access to the applications network (Sun, Y., Nanda, S. and Jaeger, T., 2015).

* 1. **Technology Heterogeneity**

The separation of services comes with another benefit known as Technology Heterogeneity. Which refers to the ability to compose a system of multiple collaborating services using different technologies. Allowing the unique ability to pick a technology based on the type of job the service is required to do (S. Newman, 2015). Working hand in hand with the small independent team structure of microservices. As heterogeneity, let us different programming languages and databases to be used, a team can cater to their needs by picking a familiar coding language, in turn the development time may increase as training developers may not be needed (Villamizar, 2017). However there needs to be a level of caution with technological heterogeneity. Such that it can counteract the main reason microservices are used. The manageability of the system can become overwhelming as the diversity among services increase. Previously mentioned, development teams choose the language a service is created in, the service becomes somewhat dependent on the developers and their skills in the language. Meaning the output of a service is a representation of the developers, and it would be difficult to bring in a new member of the team if they do not have any prior knowledge in the specific language. A way of combating such effects on a system is by introducing basic standardization where teams are encouraged to use technologies with tools and infrastructure support. While still allowing enough heterogeneity to not hinder future innovation (Bogner, J. and Zimmermann, A., 2016).

* 1. **Scaling**

Scaling microservices is also an easier process than scaling monoliths. Monolithic scaling would require major hardware improvements to be able to manage the requirements of scaling. In result it would need a large investment of both time and money (Taibi, 2017). On the other hand, microservices do not face this problem because of the independent deploy ability characteristic allowing for on-demand scaling, where monolith scaling is dependent on hardware requirements if a microservice experiences similar problems it can be re deployed into an environment with more resources (Nadareishvili,2016). The main disadvantage of microservices stems from the newfound complexity of the system. Increasing the number of interactions as the system was broken down into smaller services the systems complexity increases with more and more cross service interactions happening. Which can make testing, security, and data management more difficult with the new level of complexity. To deal with this it would be ideal to keep the interactions as simple as possible to eliminate unnecessary complexity and stick to a robust set of principles (Chen, L., 2018).

* 1. **Security considerations**

The security of any system is not to be taken lightly, and essentially if the system is being designed for the use of users. If a system has vulnerabilities, then there are chances of a security breach, which would then become a concern with users as data can potentially be stolen or intercepted. Which is no different when it comes to a microservice system, security is a concern. Even though there is limited literature in security there are attempts to address potential security concerns and offer solutions. Firstly, the way microservices function raises areas for concern, REST, XML and JSON to handle the transfer of data across the system (Dragoni, 2017). Yuqiong Sun (2015) discusses the security flaws that inherently come with microservice architecture and proposes a method to increase system security. He confirms what Dragoni says about microservices having inherent flaws, referring to the increasing number of small services within an application creates a complex network of activity, as data will be getting sent back and forth between the services making it increasingly difficult to monitor and securing the network (Sun, 2015).

The other major security concern Yuqiong Sun puts forward is to do with the cloud. The cloud is the most popular way to deploy microservice applications, some of the issues with the cloud were presented in the chapter previously mentioned. The main point in the literature here is that cloud platforms lack or have limited ability to monitor the interactions among distributed microservices (Sun, 2015). Due to the diversity of technologies increases the chances for security vulnerabilities in microservices can create vulnerabilities (Torkura, 2017). Their solutions that are provided is Flow Trap a primitive is designed to be a cloud API that can be invoked by cloud applications essentially the methodology is to forward networks requests to a security monitor which then has the ability to redirect the request if it deems it to be a breach to security (Sun, 2015). However, it is among the themes of being a prototype like most of the ideas present for microservices, even though it does show positive results within a case study. It could be argued that when dealing with security a methodology that is more developed would be the practical solution and the idea presented would need more time to develop before it is used in practice.

That is where Yarygina (2018) suggests since the aim of a microservice system is to be lightweight so should its security solution meaning it can be accessible to developers keeping that statement in mind, Netflix’s Token-based authentication is presented, a security token is created on the server side upon the successful validation of the clients’ credentials and given to the client for subsequent use. Security tokens substitute the client’s credentials within a limited timeframe (Yarygina,2018).

Unlike the previous security solutions, token-based authentication is the first solution that has some sort of depth in the literature surrounding the topic. Encryption being a well-developed area of study which provides some backing to the idea of using token-based authentication. In practice tokens would allow for two-way authentication, between the system and the user allowing for data to be sent across the network encrypted reducing the risk of vulnerable data being intercepted, increasing the overall security of the system.

* 1. **Designing a microservice**

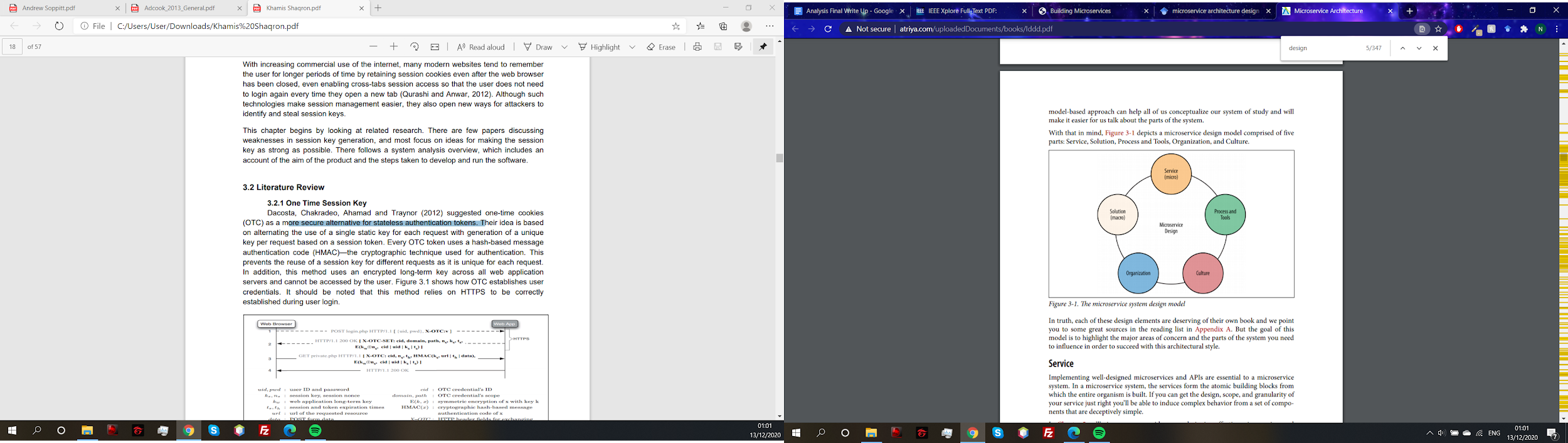
After reviewing the literature there are currently two ways that have been found to be used in practice to design microservices. Like any new technology there are many ideas but only some that seem to become industry standard, because with anything new it takes time before ideas turn into reality and over time the ideas will settle down and become more cohesive. But currently this is the start of this cycle and there are still things that are not fully understood, this section will focus on these ideas that are industry standard and briefly discuss other design ideas that can add extra insight to the research that is taking place.

Before the development phase of a microservice project the necessary time should be spent on designing the system to create a fundamental guide to build the microservice. This section will focus on the current design trends and patterns because microservices must be designed in quality and style but that is not always true

Despite microservices’ popularity, research still lacks disciplined understanding of transition and consensus on the principles (Hassan,2020). Due to the lack of refined requirements for microservices, understanding what is necessary for designing such a system could potentially seem a difficult task. However, there are a couple of ways to tackle this issue. Firstly, by looking at one of the most influential books on Microservices by S Newman, the ideas he puts forward can be used as the baseline needs for designing a microservice system. He puts forward a domain driven modelling technique, a domain driven model takes the main focuses of the system, usually the business capabilities and how these different capabilities provide context for the rest of the system (S. Newman, 2015). The evolution of the boundaries between capabilities changes over time, triggered by knowledge updates about the system domain (Hassan,2017) Suggesting that a domain driven model is potentially flexible in design as the initial system that is created can be small and as the problem grows and complexity the model can unison. Also designing a system with boundaries isolates individual capabilities within the microservice, meaning if the system requires to be changed the isolated capabilities can update individually, allowing for quicker development and deployment time (S. Newman, 2015).

**3.8.1) Model based**

A possible method of capturing a reliable design requirements specification is to examine more recent literature that has been released since Newman’s’ initial requirements were published. This should give a broader opinion, hopefully finding similarities between literature can provide adequate requirements to designing/modelling a microservice system. Irakli Nadareishvili (2016) is another influential book in this topic, and he suggests a model-based approach.

Figure 2: Microservice Architecture (Nadareishvili,2016)

The model takes a microservice and breaks it down into several sections. The intention of the model approach is to visualize how the system connects and helps create a set of principles developers can then use to build a microservice with more structure. This would involve looking at the microservice design as a whole and breaking it down into Services, Process and Tools, Culture, Organization, and solution.

Initially, looking at both proposed methods for modelling a system it may look like they are the opposites of each other. However in fact they are both very similar methodologies the premise of both are focus a system around a specific problem, breaking the problem into services with the goal of creating a system with a design adaptability into the system as a feature (Nadareishvili,2016) The differences are that the model is less a technical approach than the domain driven as domain driven focus solely on the connection of the capabilities. Whereas the model based is more orientated around creating a good design, instead of working to simply produce a solution (Nadareishvili,2016). Modelling the system is more focused on helping the designer understand the whole process including the tools as previously mentioned, creating a microservice that is adaptive but also well structured.

**3.8.2) Domain Driven model**

Another paper by Hippchen suggests using to use the domain driven model like Newman but since some time has passed between Newman’s study Hippchen is aware of the shortcoming of domain driven model “the approaches focus on the development itself but omit the design and maintenance phases” (Hippchen,2019)

Their intent is to provide design details, like the model-based approach, they have done this by a domain driven context map. The goal of using this method rather than just the domain driven is that it focuses on the macro structure of the domain rather than just the business objects (Hippchen,2019). An issue with this method is the lack of integration in the development process and it may be unclear when to create the context map. However, it does provide a better overview of a microservice architecture including the dependencies that will be beneficial for maintaining a microservice system (Hippchen,2019).

With this newer methodology it seems to cover the flaws of the domain driven model by using a content map. However, the methodology they suggest is a new idea and has only been used in a case study, and not practically. This is a flaw with the idea as without any practical evidence to support the implementation, designing a system would be difficult.

A running theme in the more recent literature is that the design principles that they are suggesting are being presented as prototypes, one again it appears the topic of modelling a microservice is still in its infancy and will take time to develop new methods and principles that have been tested in real world projects and the community will then be able to agree on a set of requirements for designing and models microservice systems. For the current time Model driven development is the most tried and tested development style for microservices and is the most designer friendly as it does not omit these stages of development like domain driven.

**3.8.3) Experimental Model**

The QOC model is a re-evaluation by Zdun of a well-known design pattern from MacLean published in 1991. However, Zdun, proposes a use of the model to be used as a design tool for pattern designing microservices. The original suggests, a design space analysis approach to represent design. Essentially is follows a set pattern of questions, identifying key design issues. Options, providing possible solutions to the questions and criteria comparing each of the options (MacLean, 1991). MacLean’s justification for his approach is the bridge between criteria as it can take considerable effort to find the correct criteria when designing and using design space allows the design to phase through the criteria to find the criteria that best meet the desired outcome (MacLean, 1991). Before looking into Zdun’s incorporation of the design methodology, based on the current literature it it’s possible to see why it may be appropriate. One of the main issues with designing microservices is the difficult to know how to manage and define services.

Zdun builds a more abstract design space by using patterns as options and adding an additional mapping between the questions, options, and criteria. (Haselböck, 2018)

It states that the method has been used to API management, even though the model has not been used for modelling the services themselves it has presented an interesting idea and use of a pre-existing design concept such as design space could potentially be used for any kind of design, even microservices. Bringing the idea that because there is not current a solid validated method apart from one of the more popular methods such as model based. That well know methods can be used to try and create a feasible design. In theory MacLean’s methodology could be used alongside the design documentation of a microservice to narrow down a more defined solution.

* 1. **Containers**

Once a system has had a successful design phase, the next step is to the development process. It is believed that the research is still in its infancy phases, and the general focus of the literature is in the general architecture. With a lack of empirical research on the design and development or microservice applications (Aderaldo, 2017). Due to the potential issue of a lack of a validated framework or requirements, could suggest that it is necessary to look at the current trends, compared to the biggest studies in microservice creation Newman and Nadareishvili.

One such growing standard is containers as they enable a microservice to distribute a lightweight infrastructure between the services, supported by technologies such as Docker (Viggiato, 2018). The idea of using a of using containers for microservices is a well-known topic as Newman believes that if a technology such as .NET adding a container for the services will prove beneficial for the system because it becomes easier to manage as a group of clustered services within a container can get support from monitoring tools (S. Newman, 2015). Such monitoring tools could be Consul or Serfnode, these example monitoring tools can be added to an existing container. Which offers robust monitoring and health checking (Stubbs, 2015) With microservice getting more complicated as they grow tools can help keep a system healthy meaning there is a great chance of the system being available, with monitoring a microservice system it would be too time consuming or cause a team a drastic amount of effort to fix a complex system fault.

* 1. **REST API**

So far everything that has been mentioned has been in theory what defines a microservice and how they should function. In practice there needs to be a way to get data in and out of each service (Sill, A., 2016).

\*\*To do

RESTful API is an API that makes calling resources very convenient and intuitive, reducing the complexity of the service. In most cases, microservices use RESTful API to deliver messages.

A key advantage of the RESTful API is that they provide a lot of flexibility. Data doesn't depend on resources or methods, so the RESTful API can handle multiple types of calls and return different data formats. Since the existing mature RESTful API is used in the HTTP environment, use GET, POST, PUT, DELETE and other actions to manipulate resources. Manipulating resources through these actions makes resource calls more flexible and convenient and makes the code more concise. Although this flexible and concise communication method is very suitable for Microservice, it is only a method and cannot be applied to all scenarios

(Hong, 2018)

REST API specification, while inter-service communication is based on lightweight protocols such as the HTTP

(Terzić, 2018)

* 1. **Conclusion**

The future is potentially very positive for microservices, as they continue to grow in popularity more businesses and industry will begin to move away from older technologies like monolith as scaling applications are extremely important in today's day and age. Currently, the state of the research resolves heavily on the migration and refactoring old systems into microservices. Meaning that there is still currently little literature for the creation of a microservice from the ground up, and a lack of information surrounding the design stages as a lot of studies centred on the structure and technical sides of microservices. However, this was to be expected as the research is still in its infancy stages and it will take time before the research is at a point where the research is at a level where there are validated frameworks and requirements. However, there is a solid base of research that already exists from the likes of Newman and Fowler. The fact that microservices are showing that they can already work in the practical sense with major businesses already using them is encouraging for the future of the technology.

1. **Requirements Approach:**

The focus of the development aspects of the project is to create an IT Help Desk application. The following section observes the process of collecting the necessary requirements, including both functional and non-functional subcomponents of the proposed system.

**4.1) Requirements Capture Method**

When selecting a priority for the requirements the MosCow technique will be used. This method takes all the requirement is an example of numerical assignment. There are for priority groups Must, Should, could and wont. All requirement under a particular requirement has the same priority (Vestola, M., 2010). MosCow is probably the best used in the early stages where requirements are not fully specified (Vestola, M., 2010). This method is important is that the priority can be used to show how important requirement should be (Racheva, 2008).

**4.2) Existing IT Help Desk Systems Survey**

Existing IT Help Desk Systems were evaluated with the intent to gather what requirements there are to create this type of system. Several existing software’s were discovered by looking at the top picks of help desk systems for 2020. The research included looked at the list of features listed on the sites.

From conducting this survey, a list of common requirements across IT Help desk systems to discover what are the fundamental requirements, to meet the baseline needs to create this system and make it successful. Other features will be listed if relevant or interesting.