Assignment Module Software Engineering

WASP Software Engineering and Cloud Computing

August, 2023

NAME: Aleksandra Obeso Duque

1. Introducing Research Topic

I am an industrial PhD student working in collaboration with a large telecom company and academia. My research project belongs to the field of networked distributed systems, specifically in the intertwine between mobile networks and cloud systems, which is also known as Mobile Edge Computing/Cloud (MEC).

In this regard, I look into how to meet strong and complex non-functional requirements that need to be fulfilled in order to host and manage future use cases such as autonomous transport, industrial IoT, and augmented reality; Through the different methods I propose, I study the tradeoff between performance and usability requirements. On one hand, since such Mobile Edge Cloud - based systems are usually quite large, heterogeneous, and dynamic, there is a need to provide higher levels of abstraction and automated mechanisms that can ease the job of application developers and system operators. On the other hand, there is a high level of criticality associated with many of the targeted use cases in terms of deterministic performance, huge scalability, and strong resiliency. These types of methods are shaping what we call the future Mobile Edge Cloud Platform.

There may be many control mechanisms or subsystems to achieve these types of requirements. I specifically investigate and propose easy-to-use mechanisms for optimized traffic management across distributed application components based on technologies such as Service Mesh [2]. To ensure or guarantee the desired performance, I investigate how to introduce intelligent control over the traffic forwarding logic across the different components of a distributed application considering a concerted management of the network-compute resources that are part of the mobile edge cloud. Examples of such mechanisms are intent-based approaches and AI/ML-based cloud control.

In this context, my research project entails three main goals:

- Propose traffic management approaches to enable easy cloud-native network programmability.
- Propose intelligent traffic optimization for mobile edge cloud workloads with stringent requirements.
- Enhancement of the Service Mesh architecture in terms of non-functional features such as performance, scalability, adaptability, and resilience.

2. **General Concepts - Main Lectures**

Behavioral Software Engineering studies cognitive, behavioral, and social aspects associated with the practice of software engineering. An important aspect of Behavioral Software Engineering states that *a new process/method is not effective purely by how it reorders work but by how it affects the involved engineers and their communication*.

As I mentioned in the introduction of my research project, one of the important aspects of my PhD studies is the evaluation of the trade-off between the performance vs. usability of the mechanisms we propose as enhancement of the Mobile Edge Cloud Platform. In terms of usability, it is always interesting to understand the ways in which the technology evolution impacts or influences the way application development and operation teams behave and interact.

To understand the kind of mechanisms we propose and evaluate, let's look at general examples. Layering methods have been widely used to design both networking and cloud stacks. They provide higher levels of abstraction, standard protocols, and interfaces that define unified ways of communication across the different networking layers. However, they have the disadvantage of introducing an associated performance overhead. It is interesting to see how newer approaches that are driven by higher performance demands, e.g., delayering the networking stack, can pose new challenges in terms of usability, life-cycle management, and other software engineering procedures. Development cycles of the kernel networking stack are getting more flexible and agile with new technologies such as eBPF and DPDK, which may require more advanced and secure approaches than current Continuous Delivery mechanisms.

A more specific aspect relates to the Behavioural Software Engineering challenges at the group/team -level. *Group cohesion and polarization* is an important aspect in this regard which can influence the effectiveness and satisfaction of team members and their dynamics. In this sense, the cloud stack has been traditionally designed as a layered architecture (i.e., infrastructure, platform, and software layers) which has introduced new ways of software/system development and operation not only from a technical but also from a social point of view. Cloud platform frameworks such as Service Mesh have, to a great extent, changed the way software teams are organized and how responsibilities are split among them. Before the introduction of Service Mesh, traffic management logic was considered, designed, and implemented as an inherent feature of a distributed application. There was no clear split of responsibilities of what system components needed to be considered during the development or during the operative stages of the application. With Service Mesh an important decoupling of the concerns between application developers and operators was introduced facilitating the delimitation of responsibilities across different engineering roles.

3. **Specific Techniques - Guest Lectures**

It is hard to find connections between the aspects learned during the guest lectures and my research project. The closest relation I can find is related to the lecture about *DevBots - AI/ML for*

supporting software developers. Since my research project mainly deals with operational aspects of distributed applications, the closest would be Operational Bots (OpsBots) which could assist the process of defining, analyzing, and validating performance requirements on intent-driven mobile cloud systems in a proactive way. Due to the broad variability of performance requirements that may exist across the different use cases enabled by a generic mobile cloud system and the high level of dynamicity of this type of system, it would be valuable to have a digital assistant that can help with the process of (re-)negotiating performance constraints given specific characteristics of the supported infrastructure or the status of a system at a given time. If the operational system is intent-driven, it would be even more valuable to have human-like type of interactions to carry out such a process. This could have great benefits in terms of efficiency and usability of the system operation.

Another important aspect mentioned in the lecture is that *the presentation or interaction between humans and the system/software is in some cases more important than the performance of the system itself*. Having an OpsBot as part of a future mobile edge cloud system would help alleviate the complexity of the system, keep track of when the system cannot fulfill stringent performance requirements due to e.g., temporary failures, and take a proactive approach that allows the re-negotiation of such performance constraints in special conditions and the implementation of mitigation actions that can lower the performance degradation of the system.

4. Selected Topics

- Automated Software Testing: In general, Software Testing refers to the process of evaluating software by following and comparing its real vs. the expected behavior; in this way, its behavior and operation can be validated. During this process, a series of bugs or faults are encountered and reported for them to be further fixed. Traditionally, this process had been carried out in a manual way but lately, this type of process is becoming more and more automated. Automated Software Testing can be achieved e.g., by using scripting methods or model-based methods in which tests are automatically extracted from an existing model. In terms of non-functional testing, there are few techniques and tools specifically addressing aspects such as Performance and Stress Testing [1]. Performance Testing makes sure that the software meets specified performance constraints whereas Stress Testing is intended to analyze how the software performs at maximum and beyond the load for which it was designed.

  In this regard, one of the main issues related to software testing is the fact that tests are usually contextual- and domain-specific, which may make them very complex to evaluate, in particular when dealing with performance testing of large-scale, heterogeneous, distributed software systems, as in my research project. The intent-based platform we intend to develop for the operation of future mobile cloud systems needs to be highly flexible in order to

cope with the dynamicity and variability of such a system; thus, it is complex to perform thorough testing [5]. The platform itself may be capable of adapting or re-configuring itself (via system actuators such as load balancing, circuit breaking, etc.) to fulfill performance requirements under different system statuses. Furthermore, due to the criticality of the use cases that can make use of our mobile cloud platform, it is important to ensure the completeness of the testing process. AI/ML can be used to perform proactive testing when the test coverage area changes. In this regard, a dynamic risk metric that is continuously evaluated during system runtime can serve as a guideline for when to trigger new tests that need to run in a timely manner.

- Requirements Engineering: Refers to the process of systematically managing system/software requirements along their lifecycle. It entails aspects such as elicitation, analysis, specification, and validation of software requirements [1]. The elicitation process collects the needs stakeholders have on the way the final software should behave [4]. From this elicitation process, a list of a well-defined and agreed-upon set of requirements is specified. These requirements serve as a guideline that leads the subsequent software development and testing process. The main goal of these consolidated requirements is to create a clear and complete description of the desired software and to make sure that the resulting software implementation is effective and efficient.

  Software Requirements can entail both functional and non-functional needs [1]. Functional requirements are those that define the logic of the application and, thus usually shape the different functions or methods an application implements. In other words, it describes the software behavior as a list of features or capabilities. On the other hand, non-functional requirements specify needs or constraints related to the overall software or system operation with aspects such as performance, security, reliability, or scalability.

  In the context of my research project (self-adaptable mobile cloud platforms), performance requirements might vary depending on the targeted use case, in particular, when the underlying system is based on non-supervised AI/ML methods such as Reinforcement Learning which are not data-driven [3]. In this sense, the requirement management process may become less explicit, rigid, and more complex; in particular, for intent-driven systems whose requirements are specified at much higher levels of abstraction than traditional systems. In this regard, there might be a need for completely new ways of elicitating, specifying, and validating non-functional requirements including automated negotiation processes to output dynamic agreed-upon set of system requirements.

5. **Future Trends**

As a computer scientist pursuing a research career in the telecom industry, I see that AI/ML is and will be a central enabler for the evolution of mobile networks in 6G and beyond. Indeed, some

of that transformation is already happening in the next releases of 5G systems. The design, development, and operation of such complex softwarized and programmable systems require higher levels of autonomy to reduce Operational Expenditures and to be able to fulfill the high, strict, and diverse requirements of future use cases that will be enabled by next-generation mobile networks. Technology trends such as digital twins, data-driven modeling and control, and intent-driven development and operation powered by AI/ML will be more widely adopted, thus increasing the need for new practices and paradigms of software engineering in the years to come.

REFERENCES

[1] BOURQUE, P., AND FAIRLEY, R. E., Eds. *SWEBOK: Guide to the Software Engineering Body of Knowledge*, version 3.0 ed. IEEE Computer Society, Los Alamitos, CA, 2014.

[2] DUQUE, A. O., KLEIN, C., FENG, J., CAI, X., SKUBIC, B., AND ELMROTH, E. A qualitative evaluation of service mesh-based traffic management for mobile edge cloud. In *2022 22nd IEEE International Symposium on Cluster, Cloud and Internet Computing (CCGrid)* (2022), pp. 210–219.

[3] HABIBULLAH, K. M., AND HORKOFF, J. Non-functional requirements for machine learning: Understanding current use and challenges in industry, 2021.

[4] JIN, Z. Chapter 2 - requirements engineering methodologies. In *Environment Modeling-Based Requirements Engineering for Software Intensive Systems*, Z. Jin, Ed. Morgan Kaufmann, Oxford, 2018, pp. 13–27.

[5] WAN, Z., XIA, X., LO, D., AND MURPHY, G. C. How does machine learning change software development practices? *IEEE Transactions on Software Engineering 47*, 9 (2021), 1857–1871.