Universidade Federal da Bahia
Departamento de Ciência da Computação

Programa de Pós-Graduação em Ciência da Computação

DESIGNING SMART CITY MOBILE APPLICATIONS: A GROUNDED THEORY

Roselane Santana Silva

DISSERTAÇÃO DE MESTRADO

Salvador, Bahia - Brasil
December 20, 2019

ROSELANE SANTANA SILVA

DESIGNING SMART CITY MOBILE APPLICATIONS: A GROUNDED THEORY

M.Sc. Dissertation submitted to the Master Program in Computer Science of Math Institute at Federal University of Bahia in partial fulfillment of the requirements for the Master degree in Computer Science.

Advisor: Eduardo Santana de Almeida
Co-advisor: John D. McGregor

Salvador, Bahia - Brasil
December 20, 2019

I dedicate this dissertation to God, my parents, my husband, my brothers, friends, and professors who gave me all the necessary support to get here.

ACKNOWLEDGEMENTS

God is the greatest provider of this opportunity and strength in my life. All honor and glory belong to Him!

The execution and completion of this cycle represents an effort that I could only have accomplished with the support of very important people in my life that always remarked a presence during special moments of my professional and personal life.

I would like to warmly thank my beautiful family, specially my parents and my two brothers. Without their caress, attention, and incentive, I would not have reached this point. My loving husband, Eber Farias, who showed me a new way of seeing life, always with patience, fondness, and love. Thank you for always standing by me.

I want to thank all the professors that supported me during the tracking of my path at the Federal University of Bahia. specially to my advisor Dr. Eduardo Almeida, who always believed in me, and always encouraged me to pursue an academic career. I also want to express my gratitude to my co-advisor and friend, Dr. John McGregor, for always being there for me, patiently pointing the way for me during the dissertation writing. I have no words to describe how blessed I am for the opportunity to work with these two extraordinary professionals.

I could not forget to thank all the practitioners and reviewers who significantly supported this dissertation, specially the interviewees for collaborating with our study and the RiSE research group/LES friends for their valuable comments and suggestions. I also would like to thank CAPES for financial support, which helped me during my master's degree. Without this support, I could not spend my time researching and trying to do my best to conduct this dissertation.

In short, someone wisely said that anyone who has a friend has everything. I totally agree because I was blessed with so many friends in my life that it would not be fair to mention only a few names here. They all know how important their friendship is to me.

Muito Obrigada, todos vocês! ♡

"We know that all things work together for good for those who love God, for those who are called according to his purpose."

—ROMANS 8:28

RESUMO

A comunidade de arquitetura de software tem desempenhado um papel fundamental no desenvolvimento de aplicações móveis. Muitas das ideias utilizadas no *design* destes sistemas vieram da arquitetura de software tradicional e têm contribuído para o que a computação móvel tem se tornado: uma tendência poderosa, adaptável, e robusta. Ao mesmo tempo, a computação móvel tende a ser uma área muito desafiadora. Aplicações no contexto de cidades inteligentes precisam operar dentro das limitações de energia da bateria, velocidade de processamento e capacidade de armazenamento dos dispositivos móveis, mas também lidar com as demandas exatas dos requisitos críticos de cidades inteligentes e operar em um ambiente exposto e de constante mudança, que nem sempre é confiável. Como não existem modelos de *design* amplamente aceitos para esse tipo de software, arquitetos e desenvolvedores recorrem a decisões arquiteturais alternativas que atendam a todas as demandas, o que requer tempo e experiência. Por esta razão, este estudo tem como objetivo construir uma teoria sobre o processo de *design* de aplicativos móveis no domínio de cidades inteligentes na perspectiva do time de desenvolvimento de software. Visando mitigar a falta de informações verificadas sobre o *design* de aplicativos móveis, conduzimos um estudo de caso múltiplo com 9 aplicativos no contexto de cidades inteligentes desenvolvidos por 4 times de desenvolvimento de software. Desses aplicativos, 6 foram submetidos a uma engenharia reversa para expor a arquitetura de cada aplicativo. Com base em todos os dados coletados, um modelo emergente de teoria fundamentada foi construído para explicar como o processo de *design* de arquitetura bem construído pode gerar um aplicativo com características desejadas. A teoria fundamentada desenvolvida através desta pesquisa e o processo pelo qual ela foi construída foram submetidos a um processo de avaliação baseado na literatura de Engenharia de Software e na experiência do pesquisador. Essa avaliação nos permitiu aperfeiçar o modelo emergente gerado e verificar que o processo experimental foi aplicado corretamente, gerando resultados válidos. Na avaliação da teoria também foi abordada algumas das ameaças à validade, como a influência do pesquisador. Para mitigar ainda mais as ameaças, esse processo incluiu coleta e análise de dados de projetos adicionais. A teoria resultante oferece explicações sobre como times de Engenharia de Software tem projetado aplicativos móveis para o domínio de cidades inteligentes. Esse conhecimento servirá como base para uma melhor compreensão dos fenômenos e definições de processos de *design* e desenvolvimento mais eficazes.

Palavras-chave: Aplicações Móveis, Arquitetura de Software, Cidades Inteligentes, Teoria Fundamentada, Estudo de Caso, Entrevistas

ABSTRACT

The software architecture community has played a crucial role in the development of mobile software. Many of the ideas used in the design of these systems came from traditional software architecture and those ideas have contributed to what mobile computing has become: a powerful, adaptable, and robust trend. At the same time, mobile computing tends to be a very challenging area. Applications in the context of smart cities need to operate within the battery power, processor speed, and capacity limitations of mobile devices, but also the exacting demands of life-critical smart city requirements, and operate in a constantly changing and exposed environment, which may not always be trusted. Since there are no widely accepted design models for this type of software, architects and developers resort to primitive design decisions to meet all the needs of these applications, which takes additional time and expertise. For this reason, this study aims to build a theory about the design process for mobile applications in the context of smart cities from the perspective of software development time. Aiming to mitigate the lack of verified information about designing mobile apps, we conducted a multi-case study with 9 smart city mobile applications developed by 4 software development teams. Six applications were reverse engineered to expose the architecture of each application. Based on all the data collected, an emergent grounded theory model was constructed to explain how the selected design process produces an app with the desired characteristics. The grounded theory developed through this research, and the process by which the theory was developed, were subjected to an evaluation process developed from the literature and the researchers' experience. That evaluation allowed us to refine the emergent model and verify that the experimental process was correctly applied there creating valid results. The evaluation also addressed some of the threats to validity such as the influence of the researcher. To further ensure validity, this process included gathering and analyzing data from additional projects. The resulting theory offers explanations for how software engineering teams design mobile apps for smart cities. This knowledge will serve as a basis to further understand the phenomena and advances towards more effective design and development process definitions.

Keywords: Mobile Applications, Software Architecture, Smart city, Grounded Theory, Case Study, Interviews

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

ACME	Architectural Description of Component-Based Systems
API	Application Program Interface
APK	Android Package Kit
CS	Case Study
EMSE	Empirical Software Engineering
SCMA	Smart City Mobile Application
GT	Grounded Theory
GPS	Global Positioning System
NAND	Neither agree nor disagree
NFR	Nonfunctional Requirements
QA	Quality Attribute
QDA	Qualitative Data Analysis
SA	Software Architecture
SE	Software Engineering
UI	User Interface

Chapter 1

INTRODUCTION

Mobile computing has the potential to aid in resolving significant societal issues that benefit from crowd-sourcing to provide large volumes of timely data. This potential may be achieved by facilitating wide-spread participation in problem identification, data collection, and analysis in the face of a rapidly expanding population. This research explores issues related to the structure of applications intended to contribute with data to smart city infrastructures.

1.1 MOTIVATION

According to the last United Nations report (NATIONS; AFFAIRS, 2019), the world's population had reached nearly 7.7 billion in June of 2019. It is expected to be 8.5 billion by 2030 and 9.7 billion by 2050. Medium-variant projections indicate that roughly 83 million people are being added to the world's population every year. Also expanding rapidly is citizens' access to mobile computing through ownership of smartphones. Nowadays, it is estimated that more than 5 billion people have mobile devices, and over half of these connections are smartphones, although it varies by country, and the economic power of the country (TAYLOR; SILVER, 2019). Nevertheless, according to OpenSignal's 2016 report, "*One thing is clear from our results: a decent mobile data connection isn't hard to find in a majority of the world's countries* (OPENSIGNAL, 2016)".

As cities become increasingly crowded, several urban issues are exacerbated, such as traffic jams, natural resource consumption, epidemics, sustainability problems, and waste management. Mobile computing has the potential to address some of these challenges by connecting people, improving citizens' quality of life, and reducing costs, among other benefits. For example, we examined a mobile application for citizens to use for reporting problems, such as traffic accidents and fires, to authorities on the road waiting for a quick response.

Mobile computing offers opportunities to make smart cities even smarter by making data instantly available for real time analysis, feedback, and control. In addition, cities are becoming smarter as sensors become cheaper and are able to measure an ever increasing variety of attributes, such as air pollution, foot traffic in a particular area, or water leaks in transmission lines (NEWCOMBE, 2014). This data can be used by software applications (apps) to aid citizens, business leaders, and government officials in making a number of decisions and in providing a wide variety of services.

1.2 PROBLEM STATEMENT

Mobile computing faces a number of challenges including constraints on battery power, processor speed, and capacity limitations of mobile devices, the exacting demands of life-critical smart city requirements, and the constantly changing and exposed environment, which may not always be trustworthy. These unique challenges have been addressed by a number of frameworks, architectures, and design strategies (MEHDI et al., 2018), but there is a lack of detailed information on how those specialized mobile device software architectures are implemented to address these challenges (DINIZ et al., 2016).

1.2.1 Objective

To provide much needed information on experiences in structuring mobile apps, this study aims to investigate what *characteristics*[1] influence the architecture design of mobile software in the smart city domain through a multi-method research approach.

Based on the results, we provide a theoretical model and a set of recommendations in order to help improve the design of Smart City Mobile Applications (SCMA).

1.3 RESEARCH METHOD

As can be seen in Figure 1.1, we split this investigation into the following activities: a *multi-case study* of several SCMAs to collect empirical evidence from the stakeholder's perspective; *reverse-engineering* of those apps to extract more information from the architecture of the mobile apps, *grounded theory building* process to analyze the collected data and derive an emergent model for designing SCMA, and a *grounded theory evaluation* process to evaluate and to provide direction for refining the theoretical model. Each approach will be covered in Chapters 3, 4, and 5.

We gathered data by interviewing 19 stakeholders of nine mobile applications intended for the smart city domain developed by four innovation development teams spread across Brazil. The main selection criteria for choosing these apps included having ready access to the development teams for interviews and having access to the source code of the applications for analysis. Besides using interviews, we reversed engineered six apps that were available in Google play and then re-interviewed some developers who participated in the development of the reverse-engineered apps.

These data were used to conduct an in-depth qualitative analysis using Grounded Theory (GT) procedures. GT is a suitable research technique for understanding and explaining a phenomenon (STOL; RALPH; FITZGERALD, 2016). In our case, the

[1]In this dissertation, we refer to *characteristics* in terms of design decisions, architectural styles, limitations of mobile devices that may affect prioritization of non-functional requirements, technologies used, among other aspects in the software development process.

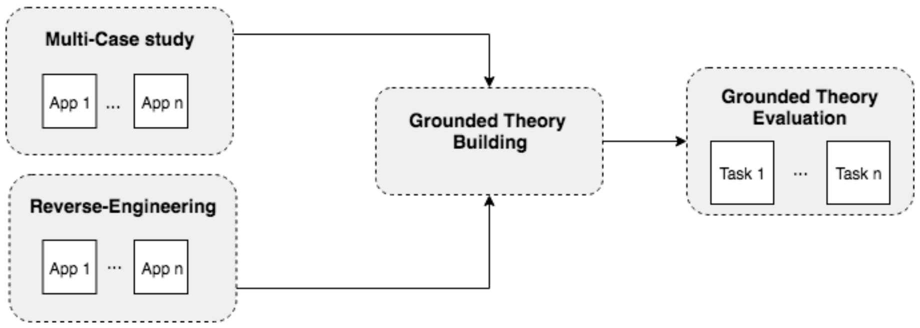


Figure 1.1 High-level research activities

phenomenon to be explained is the process of designing SCMA.

We employed coding, a technique for performing data analysis that is divided into three phases: open coding, axial coding, and selective coding (CORBIN; STRAUSS, 2008). GT will discussed in more detail in Section 4.

Through this theoretical model, researchers will be able to suggest to software architects processes and tools suitable for a specific project.

1.4 STATEMENT OF THE CONTRIBUTIONS

The research reported in this dissertation has resulted in a number of contributions to the community:

- **Empirical information about a set of application development projects:** We performed a multi-case study with teams of nine mobile applications. The applications covered in this research were similar in that they all related to issues in emergency management but differed in features and objectives. This study provided insights into the specification and design of mobile applications.

- **A theoretical model created using grounded theory:** This research resulted in a theoretical model that can support future research that uses, refines, or extends the knowledge captured in the model. The model contains 21 constructs, 17 propositions, and describes the phenomena of designing SCMA.

- **Models of six SCMA architectures:** We provided descriptions of the architectures for six SCMA obtained using reverse-engineering techniques and report on the elements of these architectures. The accompanying discussion provides guidance and data for future research and development of SCMA.

- **A rigorously derived set of findings about deployed SCMA:** We described in detail the techniques and data collected in support of the theoretical model. This description should guide future researchers in replicating our work and in structuring their own research. We formalized these contributions in a published paper in the journal Empirical Software Engineering (EMSE).

1.5 DISSERTATION STRUCTURE

The remainder of this dissertation is structured as follows.

Chapter 2 provides background information to facilitate understanding of the rest of this dissertation.

Chapter 3 outlines our empirical research approach.

Chapter 4 describes a grounded theory of the development of SCMA derived from empirical research data, and discuss the generated model. It presents the results of a study, which used reverse engineering, which extracted more information from the architecture of the mobile applications used in this work.

Chapter 5 presents the theory evaluation design, tasks and results. A set of recommendations for the SE team is also presented in this chapter.

Finally, **Chapter 6** presents a summary of the dissertation, the main conclusions, and future work.

Chapter 2

FOUNDATION

The work discussed in this dissertation is supported by previous research in four areas: software architecture, mobile software, smart cities, and grounded theory. This chapter provides background information on these subjects necessary to understand the research reported in this dissertation. Section 2.1 defines Software Architecture and presents important concepts related to this field, such as architectural styles, and quality attributes. Section 2.2 presents the key concepts of mobile software and a brief overview of the essential building blocks of Android applications. Section 2.3 presents the main definitions of a smart city and section 2.4 discusses about Grounded Theory, and how it was applied in this study.

2.1 SOFTWARE ARCHITECTURE

Software architecture (SA) plays a crucial role in the design and development of high-quality systems. Taylor, Medvidovic and Dashofy (2009) defined software architecture as the set of principal *design decisions* made during its development and any subsequent evolution, given that the stakeholders are responsible for determining which aspects are considered ‘principal’. In this definition, design decisions include how the system is organized into subsystems and components, how functionality is allocated to components, and how components interact with each other.

In short, SA is a high-level design that describes the business decisions and the requirements to be satisfied by the system. It does not define details of implementation, but rather addresses the interactions and the behavior of components.

Architectural styles and patterns define how to organize the components of the system so that one can build a complete system and satisfy the customer’s requirements in a given development context (SHARMA; KUMAR; AGARWAL, 2015). A software architectural style is a collection of architectural design decisions that are applicable in a given development context, constrain architectural design decisions that are specific to a particular system within that context, and elicit beneficial qualities in each resulting system (TAYLOR; MEDVIDOVIC; DASHOFY, 2009). There are many architectural styles used by software developers, so one needs to understand which particular architectural style will be appropriate for each project. In other words, architectural styles consist of a set of shared assumptions and constraints across a set of architectures to solve common problems of design. Some examples of architectural styles are presented in Table 2.1.

Table 2.1 Example *Architectural Styles* taken from Taylor, Medvidovic and Dashofy (2009)

Client-server	It is a network architecture which separates the direct interaction with users (a client) from the major domain computation (a server). Each instance of the client software can send requests to a server. This server processes each request and sends a response to the client
Pipe-and-filter	Components take input streams and transform (filter) the input data into output data that is streamed. Connectors are conduits (pipes) for data streams from one filter to another. Style invariants are: filters are independent from other filters (no shared state) and a filter has no knowledge of filters from which it receives data nor of those to which it sends data.
Publish-subscribe	Subscribers register/deregister to receive specific messages or specific content. Publishers send those message types, either synchronously or asynchronously, to subscribers who have registered an interest.
Peer-to-peer (P2P)	Each node is equal to every other node and any node may send requests to any other node. This network does not have the notion of clients or servers, only equal peer nodes that simultaneously function as both client and server to the other nodes on the network. P2P networks are typically used for connecting nodes via largely ad-hoc connections.

The ability to manage information is key in the design of mobile systems, and novel software designs are required to assist this effectively by addressing the decisions, constraints, and goals of the mobile setting. The objective of designing an architecture is to understand the essential design decisions and the key system quality attributes, which are discussed in Section 4.3.1. In this context, there is a set of architecture principles attributed to mobile applications, presented by Bagheri et al. (2016), that have played a significant role in the design process of mobile applications: *architecture building blocks, hierarchical (de)composition, architectural styles, software architecture models, architecture implementation and deployment, and support for non-functional properties.* For the purposes of this research, the architectural styles of the investigated apps are discussed in Section 4.4.

2.2 MOBILE SOFTWARE

Mobile software has evolved a lot, and the systems keep getting even smaller, smarter, more pervasive, more powerful, and more integral to our lives. Recent statistics show that the number of mobile apps is on the rise and they are expected to increase even more in the near future (SAIFI, 2017). As evidence of this, the number of available apps in the Google Play Store was most recently placed at 3.6 million in March 2018 (STATISTA, 2018a). The number of mobile app downloads worldwide has been steadily increasing. In 2017, users downloaded 178 billion apps. That number is projected to grow to 258 billion in 2022, a 45 percent increase over five years (STATISTA, 2018b).

With this explosion in the development of mobile software, it has become an important area for software engineering (SE) researchers, and studies have been addressing different facets of mobile software systems. Recent studies have been conducted to help researchers and practitioners to quickly get an overview in the area of mobile software systems addressing different facets in this domain (NAGAPPAN; SHIHAB, 2016). In this dissertation, we focus on the architecture design of this type of software.

One facet in mobile applications that has been investigated is the set of challenges that directly impact the software development process (WASSERMAN, 2010; NAGAPPAN; SHIHAB, 2016; BIØRN-HANSEN et al., 2019), such as limited resources, need to integrate with other applications, sensor handling, peculiar infrastructure, security and testing issues, and so on. Such process challenges have motivated our work on the concept of analyzing how this kind of software has been designed in order to encourage developers to adhere to the important principles of abstraction and modularity that are built into the platform architecture.

Android Applications

As aforementioned, one of the contributions of this work was to provide descriptions of the architecture of the set of selected apps. To achieve that, we used a reverse-engineering technique developed by Bagheri et al. (2016) for Android applications.

We investigated the architecture of Android apps. This choice is driven by the open-source nature of Android, which allows researchers greater access to system internals. The Android community has produced a large body of literature as well as techniques for reverse engineering and other analyses. Next, we provide a brief overview of the essential building blocks of Android applications. Android has the following core framework components that define its architecture (GOOGLE, 2018a):

- `Activity:` is a class with user interaction. An app may have many independent activities that form user interface components.
- `Service:` is a class that runs in the background to perform tasks without affecting user interface (UI) components.
- `Broadcast Receiver:` is a component that enables the system to deliver events, such as a low battery warning, to the app asynchronously. This component allows the app to respond to broadcast announcements.
- `Content Provider:` is an abstraction of a database that allows storage and retrieval of data within the app with the use of an Android database.

The first three component types (*activity, service*, and *broadcast receiver*) are activated by an asynchronous message called an *intent.* Intents allow the app to interact with components within the application as well as with outside applications.

2.3 THE SMART CITY CONCEPT

The concept for a smart city was first conceived in the nineties, in the context of Internet adoption, and it has evolved gradually ever since as infrastructure control and planning have become more automated. Although there has been an exponential increase in interest for systems that control more of everyday life and data from the sensors (COCCHIA, 2014), the definition of smart city is still emerging, and the work of defining and conceptualizing the breadth and depth of the concept is still in progress (AIRAKSINEN et al., 2017).

This is partially due to two visions of what constitutes "smart". One view is an area in which industries are sufficiently advanced that they require intensive information services and infrastructure, such as that available in the Silicon Valley area in California, USA (KITCHIN, 2015). A second view, and the one that more closely reflects our view is an area in which the infrastructure for everyday living takes advantage of technologies. This view is of a city that collects and analyzes data to aid in both immediate decisions, such as control of traffic signals, and in the long term planning decisions, such as locations of new electric vehicle charging stations. This is the view motivating our work. Due to the growing evolution of the expression "smart city" and the ambiguity in definitions, architectures for systems operating in a smart city are being designed without an agreed upon prioritization of quality attributes and without clearly stated constraints. At the same time, traditional analysis and design processes cannot satisfy the requirements called for in this always changing, sometimes safety-critical, context. Furthermore, collecting data is getting easier whereas finding an effective way to store, manage, and analyze the data became a very important challenge for software engineers (WENGE et al., 2014).

2.4 GROUNDED THEORY (GT)

GT is a method for qualitative research, proposed by sociologists (GLASER; STRAUSS, 1967), that focuses on an interpretative process through the analysis of meanings and concepts used by actors in real contexts in the sense of understanding actions of a phenomenon. Corbin and Strauss (1990) defined systematic methods and procedures for grounded theory research, which will be described later in this section. The expected end result of applying GT methods is a set of propositions, which is totally supported by the data and that richly describes the phenomenon under investigation. GT has become an accepted approach for qualitative research. We chose to use this approach for two main reasons. First, there are several sources explaining the concepts and how to apply this approach (GLASER, 1998; CORBIN; STRAUSS, 1990; SEAMAN, 1999; CORBIN; STRAUSS, 2008; SJØBERG et al., 2008). Second, many researchers have used the technique frequently to conduct data analysis (ADOLPH; HALL; KRUCHTEN, 2011; HODA; NOBLE; MARSHALL, 2012; SOUSA et al., 2018).

The theory generation process involves using a method named *constant comparison*, which starts by attaching labels (codes) to pieces of text which are descriptive of the topic of the study. Then, the field notes are grouped into categories based on the codes and subcodes they have been assigned. Next, the researchers write memos, i.e, a synthesized idea of the coded data, setting the researcher's expectations when reading each code. Finally, the next step in the process is to find patterns in the categories that have been generated (SEAMAN, 1999).

Corbin and Strauss (2008) divided the coding portion of the method into the following procedures: *open coding*, *axial coding*, and *selective coding.* To illustrate the use of each procedure, a real example from the literature is presented in this section, as well as an example of its use from our interview transcripts is given in Section 3.5.

Open Coding

It refers to the process of generating codes (text strings) and associating those codes to pieces of text, which are relevant to describing concepts of certain themes that are of interest in the study. The coding is "open" in that the researcher is able to create any code he wishes. These codes become categories by grouping codes that refer to similar constructs. The following is an example extracted from Hoda, Noble and Marshall (2013) on Self-Organizing Roles on Agile Software Development Teams:

- `Interview quotation:` *"We had [Mentor] as well at the time [the team started Agile practices] so...It made it easy...having [Mentor] there as a backup... [it has] been really good to have that guidance from [the Mentor]."* — (I8, Tester).
- `Key Points:` "Coach providing guidance in initial stages"
- `Codes:` Providing initial guidance
- `Concept:` Providing initial guidance and support
- `Category:` Mentor

Coding is an interactive process, that is, it involves reading through the text, creating and assigning codes, and then reading through it again to make sure that they are being used consistently. Categories can be added, deleted, merged, or modified during the analysis. The process continues until the saturation point is achieved when analyzing new text.

Axial Coding

It refers to the process of identifying relationships between categories. Now, relationships between concepts (conditions, interactions, and consequences) and categories of interest are evaluated to enable the identification of relationships among them.

Figure 2.1 shows several concepts that were related to the *mentor* category. The researchers condensed all these concepts by saying that the mentor encourages, supports, and oversees the team as they begin to practice agile software development on a day-to-day basis.

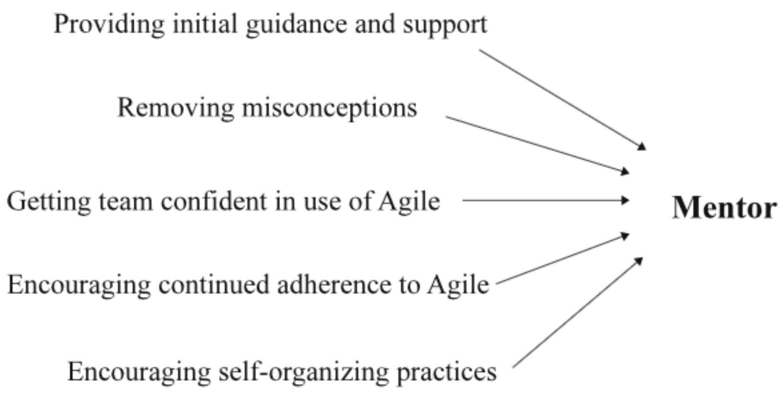

Providing initial guidance and support
Removing misconceptions
Getting team confident in use of Agile
Encouraging continued adherence to Agile
Encouraging self-organizing practices
Mentor

Figure 2.1 Example of emergence of the category "Mentor" from underlying concepts. (HODA; NOBLE; MARSHALL, 2013)

Selective Coding

It refers to the process of choosing one or a few major categories, which is/are related to other categories, integrating and refining the identified ones in the process of building the theory. It can be an existing category, or a new category can be created.

Concluding the example, a grounded theory on self-organizing Agile teams presented three main categories/themes that emerged from the study: self-organizing roles, self-organizing practices, and critical factors influencing self-organizing teams.

The grounded theory methodology was used to study day-to-day reports of architectural decision-making from software engineering teams (SOUSA et al., 2018), which leads us to believe that Grounded Theory is well suited to exploring how software engineering teams design mobile apps for smart cities.

2.5 CHAPTER SUMMARY

In this chapter, we presented an overview of the main topics addressed in this dissertation. We began by introducing the main area of this work, software architecture. We also provided brief introductions to mobile software and smart cities as major influences on content and grounded theory as a systematic research method used to analyze the data collected during this study and to generate a theory. The next chapter presents the details of the research approach assembled to provide empirical evidence on designing smart city mobile applications.

Chapter 3

RESEARCH DESIGN

This chapter outlines the multi-method approach we assembled during this work to provide empirical evidence regarding the design of smart city mobile applications (SCMA) from the perspective of software engineering (SE) teams. The integration of multiple complementary research methods allowed us to triangulate the results increasing the empirical knowledge about this area (MORSE, 2003). We conducted a multi-case study with nine SCMA developed by four software development teams and used the results of this study to hypothesize a theory about how SCMA are developed.

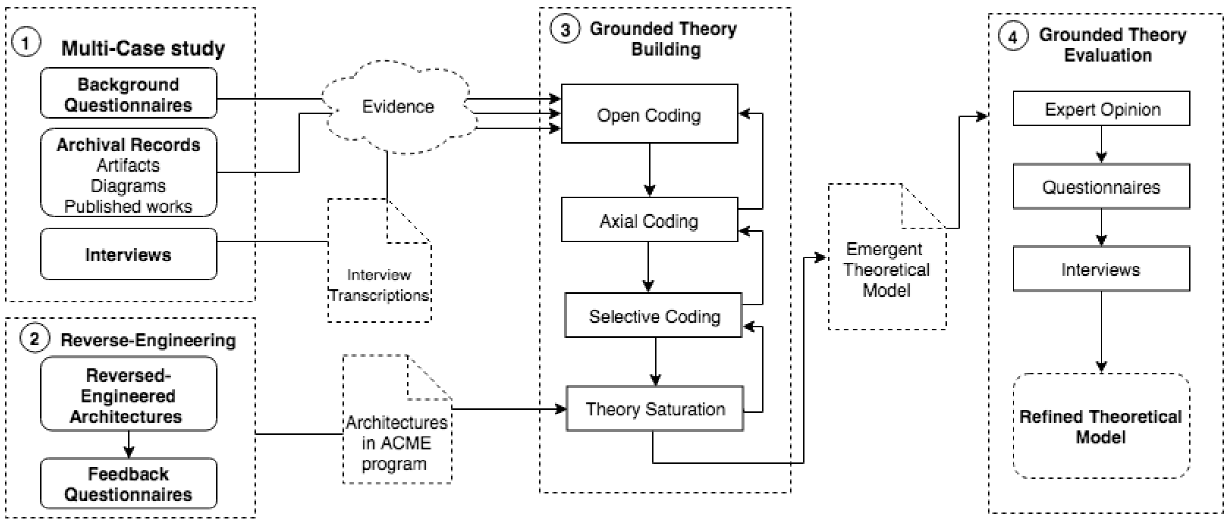

1 Multi-Case study
Background Questionnaires
Archival Records Artifacts Diagrams Published works
Interviews
Evidence
Interview Transcriptions
2 Reverse-Engineering
Reversed-Engineered Architectures
Feedback Questionnaires
Architectures in ACME program
3 Grounded Theory Building
Open Coding
Axial Coding
Selective Coding
Theory Saturation
Emergent Theoretical Model
4 Grounded Theory Evaluation
Expert Opinion
Questionnaires
Interviews
Refined Theoretical Model

Figure 3.1 Research Analysis Process

Figure 3.1 describes our research analysis process in terms of the activities and flow of information. The studies performed can be divided into four stages. In stage 1, we conducted a *multi-case study* with four development teams using background questionnaires, archival records of nine mobile applications and 19 interviews with stakeholders of SE teams. In stage 2, we *reverse-engineered* the six apps that had apk files in the

Google Play store since we had access to tools for manipulating apk-formatted files. We used the information about the as-built architectures to validate the results of the data analysis of the as-designed architectures. In stage 3, we used Grounded Theory (GT) to analyze all evidence we got from previous stages to derive an emergent theoretical model for designing SCMA. Finally, in stage 4 we applied a set of tasks to evaluate and refine our model.

Further details of the design and execution of the multi-case study and reverse-engineering are described in our complementary material site[1], which provides a full replication package of this study. In the next sections of this chapter, we describe how the multi-case study and the reverse-engineering (stages 1 and 2) were conducted. Chapter 4 and 5 will describe in more detail our qualitative analysis through the grounded theory building and evaluation process respectively (stages 3 and 4).

3.1 CASE SELECTION

The multi-case study was initiated in the middle of 2017 and conducted in two parts: first, a single embedded case study (CS) was conducted with a development team, which we named *T1*, following the CS guidelines described in Runeson and Höst (2008), and then, this study was replicated with teams *T2, T3,* and *T4* in the following year (see Fig. 3.2).

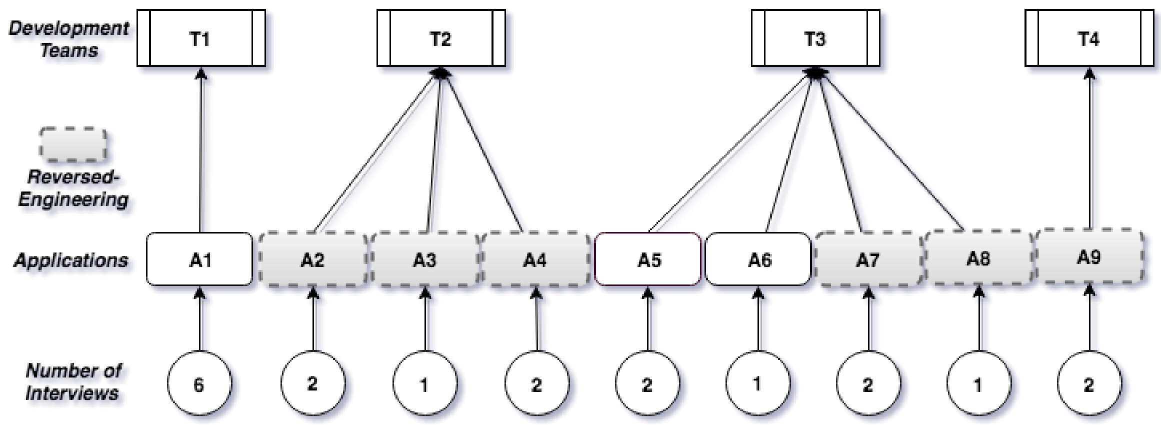


Figure 3.2 Characterization of the 19 interviews about 9 mobile applications developed by 4 different development teams

The main criteria we used to select the apps in both parts of the study were the following: mobile applications intended for the smart city domain. For the purposes of this study we created our own characterization of smart city mobile application (SCMA) based on scientific references, highlighting the necessary criteria on which our selection was based:

- Enables solutions to improve citizens' quality of life (DAMERI, 2013);

[1]Supplementary material site is available at https://rose2s.github.io/EMSE2019

- Promises multiple benefits to safety and security, environment and transportation, energy management, educational facilities, tourism, and citizens' health (KHATOUN; ZEADALLY, 2016);
- Facilitates citizens' interaction to solve urban problems, often on the fly (ESPOSTE et al., 2019).

Besides being a SCMA, other selection criteria included having ready access to the development teams and having access to the source code. First, we made contact with practitioners from the Fraunhofer Project Center for Software Systems and Engineering (FRAUNHOFER, 2014), whose focus is the development of innovative software solutions. For the teams *T2*-*T4* we kept the same selection criteria as for *T1*, but we broadened the search to include development teams around Brazil. After collecting some general information such as the domain, the status of the project, and team background, we sent an email to selected project managers telling them about our study proposal and asking for their participation. Once we got the manager's agreement, we asked them to assist us in identifying development team members to be interviewed. Since our study required specific information to be gathered, participants needed to have played specific roles and have at least one year of experience on the project.

All four teams selected work with mobile applications which support citizens and entities related to public services, such as public safety, tourism, transportation, education, and smart buildings; through training and developing human resources in the context of smart cities; and through disseminating knowledge, experiences, and results in scientific publications.

Most teams we selected, except for *T1* which had a SE team composed of 6 stakeholders, had either 1 or 2 developers. We interviewed all stakeholders even those who had already left the project by the time we conducted this study. A characterization of each subject is detailed in Table 3.1, including an interviewee ID (I1 to I19), their roles in the project, years of experience, the application with which they were involved (A1 to A9), and their respective development teams (T1 to T4).

3.2 CHARACTERIZATION OF SOFTWARE SELECTED FOR STUDY

In this section, we present a short description of each application we selected grouped by their respective development teams.

T1: `System Engineering Project Center`

- `A1` is an effective emergency and crisis management solution developed by a European-Brazilian partnership to ensure that the people visiting large-scale events will feel safe[2]. An experimental prototype was tested at the 2014 FIFA World Cup.

T2: `Development Group for Smart Cities`

- `A2` is a mobile traffic application for bicycle navigation and information sharing among cyclists. It provides a crowd-sensing service to report and share special

[2]A1 is private but indirect information is available at http://www.rescuer-project.org

Table 3.1 Role and experience of the respondents according to their groups

#	Role	Experience (years)	Application	Development Team
I1	Manager	5	A1	T1
I2	Req. Analyst	20+	A1	T1
I3	Mobile Developer	8	A1	T1
I4	Web Developer	3	A1	T1
I5	Architect	8	A1	T1
I6	Researcher	1	A1	T1
I7	Req. Analyst, Architect Developer	1	A2	T2
I8	Developer Tester	2	A2	T2
I9	Req. Analyst, Architect Developer, Tester	2	A3	T2
I10	Developer, Tester	1	A4	T2
I11	Developer, Tester	4	A4	T2
I12	Manager	20+	A5, A6	T3
I13	Req. Analyst, Architect, Developer, Tester	2	A5	T3
I14	Developer	2	A6	T3
I15	Manager	4	A7, A8	T3
I16	Developer	4	A7	T3
I17	Developer	1	A8	T3
I18	Manager, Req. Analyst	20+	A9	T4
I19	Developer, Tester	10+	A9	T4

locations and events on the city roads, informs about weather conditions, and issues voice alerts near dangerous locations in real time.

- `A3` is a smart collection app that facilitates the collection of cooking oil residue through an intelligent system that informs collecting companies about the level of oil in real time at the eco-points distributed throughout the city and helps turn the cooking oil collected into fuel oil.

- `A4` is a smart application that streamlines the search for vacant spots within a parking lot with a vacancy detection system[3].

T3: `Project for Smart Cities`

- `A5` is intended to speed up registration of emergency incidents in the University. It was developed in partnership with the public safety department, which is the

[3]A2, A3, and A4 are no longer available but the APK files are at our supplementary material site

main customer, to allow the registration and monitoring of the status of emergency incidents for its users. The idea arose from the need to assist security guards of all campuses of the Federal University of Rio Grande do Norte, because of suspicious actions and incidents that had occurred and were reported by students.

- `A6` is integrated with A5 and it is aimed to assist security guards of the university at the moment of an incident. It is intended to speed up the arrival of police forces at emergency incidents in the geographical area of the University[4].

- `A7` is a citizen problem reporter app that allows the general public to report non-emergency problems concerning public services such as trash, pothole, and flooding[5]. Through the app, local government personnel can monitor, verify and assign reports to responsible agencies for resolution.

- `A8` is a smart city app that is intended to enhance tourists' travel experience[6]. The app encompasses a mobile tourist guide application, a tourism information system, and a business intelligence infrastructure. This app was sponsored by the city hall and headed by the department of tourism. The objective is to assist the tourists free of charge in an easy and practical way and to enhance the experience of those who want to know the city of Natal better.

T4: `Information Technology Management`

- `A9` is a communication channel to report security incidents and is intended to increase the quality of prevention and safety-related actions on university campuses at the University of Sao Paulo[7].

3.3 QUALITY OF THE SELECTED APPLICATIONS

We asked the managers of each development team about the acceptance of the selected applications by the intended audience and the general answer was that citizens have supported the apps by downloading and using it and giving feedback to improve it. For those apps that have not achieved many users, one of the team's goals after releasing the product has always been to encourage users to adopt it.

To exemplify the success of the app A9, Ferreira et al. (2017) presented statistical data collected by a case study showing its effectiveness. A9 was released in 2016, and by the date of this work, it already had around 12,000 installs in iOS and Android versions, which shows a good acceptance from the audience. Important findings of the study include: (1) after the app introduction, the security guard response time reduced from 8-10 minutes to 5-7 minutes; and (2) the *emergency call* and *watch over me* buttons were the types of use most often reported by the users.

[4]A5 and A6 are private and not available for download

[5]A7 is no longer available but the APK file is at our supplementary material site

[6]A8 is available for download at goo.gl/8wBxVb

[7]A9 is available for download at goo.gl/1njBJz

These buttons have been widely used by users to keep themselves safe. When pressing the "*watch over me*" button, the user can alert the security dispatch center that he is going to be walking around a suspicious place. If within the specified period of time, the user shakes his phone, it sends an alert to the dispatch center, and a vehicle is immediately sent to the user's location. When the *emergency call* button is touched, the local campus security is called and immediate action can be taken including sending a vehicle to the user's location.

Amorim et al. (2017) described the evaluation process used for the app A1 with experts on emergency response. The results showed that the participants thought the app was well designed, easy to understand, easy to learn, and easy to use.

3.4 DATA COLLECTION

As shown in Figure 3.1, we used a variety of activities to collect data so as to achieve maximize the benefits of the multiple available sources of evidence.

3.4.1 Stage 1: Multi-Case Study

- **Pre-Questionnaire -** A background questionnaire was sent to each participant before the interview to gather personal information and their main roles on the project (recall Table 3.1).

- **Archival Record -** The type of archival records that were used varied depending upon what the project had documented. *T1* was the only development team that had formal documents so that we analyzed artifacts, such as requirement specifications, design documents, mobile solutions, a conceptual model of mobile user interaction, portability and variability management. The other teams' artifacts were based on quarterly reports and informal diagrams. We analyzed all the documents to which we had access. Additionally, each team had at least one published work about the design and/or implementation of their mobile application, from which we could capture some information.

- **Interview -** The interview sessions were based on a semi-structured model (BRERETON et al., 2008). Aiming to investigate the design process from the perspective of the SE team, we divided the interview into three points of interest following the guidelines defined in (YIN, 2013). These three points were captured on cards used to present during the interviews: *card 1* - software requirements, *card 2* - mobile development and *card 3* - software architecture. All cards were used with all participants, even if it was not directly related to their role on the project. The division of the cards helped us to guide the interview based on the most important role played by the interviewee, e.g., the participant who mainly worked on the mobile aspect of the app first answered questions on Card 2. We paused briefly between each of the cards telling the interviewees we would be moving to another card. This way, we kept the participant engaged, and answering the main questions before getting tired. Each interview lasted on average of 40 minutes. In total, a set of six participants from team T1 were interviewed, five from T2, six from T3, and two from T4, resulting in nineteen interviews and roughly thirteen hours of audio-recorded and transcribed.

3.4.2 Stage 2: Reverse-Engineering

- **Architecture Reverse-Engineering -** The goal of this part of the study was to replicate the original study described by Bagheri et al. (2016) so as to investigate software architecture principles in our set of apps as another data source for our analysis.[8] We reverse-engineered the software architecture from either the source code of the applications that were public and available in Google Play Store or from its APK file. The applications *A2, A3, A4, A7, A8, and A9* are represented by the gray boxes with dashed lines in Figure 3.2.
- **Feedback Questionnaire -** We used a short questionnaire to validate the reverse-engineered architecture, its structure, and the architectural styles used from the point of view of the architects/developers.

According to Runeson and Höst (2008), it is useful to pilot-test the data collection instrument to anticipate problems in data analysis; therefore, before doing any interviews, a pilot was conducted with a software architect of a smart city project. This was a good approach that helped refine our interview questions. All the interviews were recorded with the consent of the interviewees, then the audio files were transcribed almost literally and integrally using Trint[9], an on-line transcription application.

When further data collection and analysis leads to a point of diminishing results, the category is said to have reached theoretical saturation and the researcher can stop collecting data (HODA; NOBLE; MARSHALL, 2012). In this research, we stopped collecting and analyzing data after interviewing nine mobile applications from four different development teams (recall Fig. 3.2), when interviews stopped providing new insights into the existing categories. This was a clear indication we had reached a saturation point, i.e., no more new information would be gained from additional subjects (CORBIN; STRAUSS, 2008).

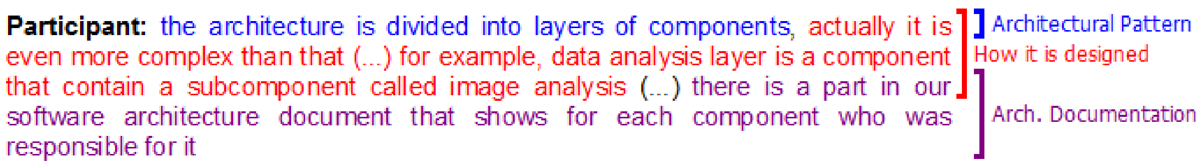
3.5 DATA ANALYSIS

GT is a suitable research technique for understanding and explaining a phenomenon (STOL; RALPH; FITZGERALD, 2016). In our case, the phenomenon to be explained is the process of designing architectures for SCMA. Using GT enables us to generalize findings from case studies, a common situation where statistical generalization is not sensible (YIN, 2013). Rather than only using GT techniques, we explicitly claim to use GT by building a theory based on the data as an effective way to obtain general knowledge about the development of SCMA and for understanding the development process (CORBIN; STRAUSS, 2008).

[8]Since it is not a trivial task, we created a tutorial about this study, and made it available in our complementary material.

[9]Trint is available at https://trint.com

We employed coding, a technique for performing data analysis in GT (recall section 2.4). For each transcript, two researchers employed (independently) *open coding*, where text phrases that represent the domain were used as labels or codes and associated with portions of all nineteen transcriptions and archival data. Two software packages that facilitate coding and other types of qualitative analysis were used: Atlas.ti[10] and QDA Miner[11]. Since coding is a subjective task and it is based on the researcher's experience, the resulting codes of each author for each transcription were compared, merged and refined until a consensus was reached.



Participant: the architecture is divided into layers of components, actually it is even more complex than that (...) for example, data analysis layer is a component that contain a subcomponent called image analysis (...) there is a part in our software architecture document that shows for each component who was responsible for it

Figure 3.3 An example of open coding extracted from QDA Miner tool

As shown in Figure 3.3, in the quote[12], a researcher attached pieces of text from the interview transcripts, in blue, red and purple colors respectively, to one of the chosen codes "Architectural Pattern","How it is designed", and "Architecture Documentation". After the researchers discussed all the generated codes, the codes were grouped according to their properties, composing concepts that represent categories. Since *Open coding* is an iterative process, the codes were refined and new categories were identified until the last document had been analyzed and the saturation point had been achieved.

In the second phase, *axial coding*, categories were associated with the set of codes previously identified corresponding to the entire design process of the smart mobile applications. Using the same example (Fig. 3.3), the three pieces of text were added into the following categories:

1. Architectural patterns for SCMA
2. How software architecture is designed
3. Architecture Documentation

Finally, in *selective coding* we identified the core categories that most describe the design process of the selected SCMA by following Sjøberg's framework (2008). The core categories are defined in the table of constructs (see table 4.2) which make up the theory building process detailed in Chapter 4.

[10]Atlas.ti is available at https://atlasti.com

[11]QDA Miner is available at https://provalisresearch.com

[12]Since interviews were performed in the participant's native language, the quotes are around texts that were translated into English.

3.6 VALIDITY PROCEDURE

We carefully built the studies to reduce bias in data collection and data analysis by using data triangulation. All six interviews for *T1* were face-to-face and three researchers (the author of this dissertation and two masters students) participated in all six interviews. Moreover, the transcriptions were validated by the participants, allowing them to give feedback on interview audios and transcripts.

To triangulate the information gotten through documentation, we made summaries in which one researcher did a summary and the other two reviewed it. Since the development teams are located in four different states of Brazil, interviews for *T2*-*T4* were done via Skype, using its call recorder tool.

3.7 CHAPTER SUMMARY

In this chapter, we outlined the research analysis process used in this work. This study can be divided into four stages. The first two stages are studies conducted to collect empirical evidence on how mobile applications developed for smart cities were designed from the perspective of the software engineering team, and the last two stages represent the process of building and validating a theoretical model based on Grounded Theory. The next chapter presents our Grounded Theory building process and the steps to generate an emergent model.

Chapter 4

A GROUND THEORY ABOUT THE DEVELOPMENT OF SMART CITY MOBILE APPLICATIONS

A major facet of our research is developing a theory about the creation of mobile apps in the Smart Cities context. Although Section 2.4 presented several approaches for applying GT in Software Engineering, we used Sjøberg's framework (SJØBERG et al., 2008) to represent and describe a theory from empirical data collected in this research project. This framework was selected because it made describing the theory creation process more straightforward and easier to understand.

Table 4.1 shows the major constructs of this framework and maps it onto our scenario. The steps model the situation where an actor applies a technology to perform development activities on a software system.

Table 4.1 Major constructs in Sjøberg's framework for SE theories

Archetype Class	**Subclasses**
Actor	Software Engineering (SE) team
Technology	Design processes for mobile technologies
Activity	Creation of the Architecture Design
Software System	Smart City Mobile Applications

Before describing a theory, these high-level concepts must be instantiated for the problem at hand. According to Sjøberg's framework (SJØBERG et al., 2008), a theory is created by building the following elements: theory constructs, propositions, explanation, and scope. The theory is based on, and the instantiated constructs are created from, the completed coding process (recall Section 3.5). Each element of the framework is briefly described in Section 4.1 and then the detailed information for our study is presented in Section 4.2.

4.1 CHARACTERIZATION OF A THEORY

Theory constructs

Theory constructs are the core categories identified previously in the coding process that helps to explain a phenomena. For example, a construct termed “architecture” might be identified after interviews produced references to high-level design, system design, and system structure.

Theory proposition

A theory proposition is a relationship between theory constructs, which describes how core categories interact with each other. For example, a proposition might relate the architecture to non-functional requirements (NFR) by stating the proposition “an architecture must satisfy the NFRs as well as the functional requirements”.

Scope

The scope of a theory is the universe for which the theory is expected to be an accurate explanation. For example, a theory might explain that it is not applicable to game apps, but does apply to apps that communicate with a central server.

Explanation

The explanation is based on all the previous steps of the theory building process. It describes “why” the resulting theory is what it is. For example, user training on a product is seen as related to acceptance by the users of the product. In one of the studied apps, user training was related to users being successful at directing help to incidents.

4.2 CHARACTERIZATION OF OUR THEORY

In this section, we present our theory created from the empirical data. We describe each of the artifacts using the constructs from Sjøberg’s framework (SJØBERG et al., 2008).

Constructs

Table 4.2 describes the constructs identified during our study. We identified 21 core categories directly related to our central category, which is SCMA.

Table 4.2 List of constructs identified in the study

C1	*Architecture design*	A high-level structural design of a software system.
C2	*Architectural styles*	A set of design decisions that identify the kinds of components and connectors that may be used to compose a system or subsystem.
C3	*Design decisions*	A description of the set of rationales, design rules, and design constraints for a given architecture (JANSEN, 2008).
C4	*Technical debt*	A concept in SE that reflects the extra development work that arises when code that is easy to implement in the short run is used instead of applying the best overall solution (JANSSEN; JANSSEN, n.d.).
C5	*Smart city context*	A characterization of a city's infrastructure for collecting and using data in management and policy decisions.
C6	*Domain experts*	A person with special knowledge in the context in the domain of smart cities (e.g., a firefighter is an expert in emergency management).
C7	*Technical skills*	Abilities needed for software programming.
C8	*Documentation*	Artifacts that explain the software (requirement specification).
C9	*Functional requirement*	A requirement that defines what a system is supposed to do.
C10	*Non-functional requirement*	A requirement that specifies criteria that can be used to judge the quality of operation of a system, rather than specific behaviors.
C11	*Technologies*	Technical tools that support software development.
C12	*API*	A software intermediary that describes how software units talk to each other.
C13	*Framework*	It is a coherent unit of reuse, both by use-relationships and by extension through sub-classing.
C14	*Software development*	The process of specifying, designing, developing, and testing involved in mobile applications.
C15	*Challenges*	Challenges faced when designing and developing a SCMA.
C16	*Development approach*	There are three primary approaches to building mobile apps: web, hybrid and native.
C17	*Resources management*	The effective use of mobile resources.
C18	*Software testing*	An investigation conducted to provide information about the quality of the software product.
C19	*Testing tools*	Software intended to help software engineers to find bugs in mobile apps.
C20	*Testing issues*	Issues related to testing mobile software (e.g., testing real scenarios).
C21	*User training*	Process of training a staff who will work supporting the citizen through the app.

Propositions

The constructs from Table 4.2 were examined and relationships among them were considered. All pairwise combinations of constructs were analyzed. When a pair of constructs were found to be related, a proposition was constructed that captured the relationship. These relationships were based on information from several sources including discussions with the study participants, the professional and research literature, and the experience of the researcher. The initial statements of the propositions were reviewed by the study's participants and researchers before being accepted as final. Table 4.3 describes the final propositions produced by our study.

Table 4.3 List of propositions identified in the study

P1	Domain experts positively impact the definition of requirements for a SCMA.
P2	The lack of documentation creates technical debt.
P3	Functional requirements for a SCMA come from a smart cities context, which is citizen-oriented
P4	Most non-functional requirements are not taken into consideration by SE team when making design decisions.
P5	The satisfaction of non-functional requirements depends upon the design decisions made.
P6	The level of technical skills impacts the development of SCMA.
P7	The lack of architecture decisions creates technical debt.
P8	The architecture design of SCMA is API-centric.
P9	The use of architectural styles positively impacts the design of SCMA.
P10	Effective mobile resource management positively impacts the architecture design of SCMA.
P11	The smart cities context brings complexity to mobile applications.
P12	The development or adoption of smart city frameworks reduce the complexity of the development of SCMA.
P13	The adopted development approach brings specific development challenges for SCMA.
P14	Smart technologies positively impact the development speed of SCMA.
P15	The lack of testing tools for mobile applications negatively impacts the testing process of SCMA.
P16	Training users positively impact the acceptance of SCMA products.
P17	Testing SCMA involves the need to simulate real-world scenarios.

Scope

The scope of the theory is smart city mobile applications. Our theoretical model is illustrated in Figure 4.1, where it is constrained to "SE team", "create architecture design" activity, and "smart city mobile applications" of a software system.

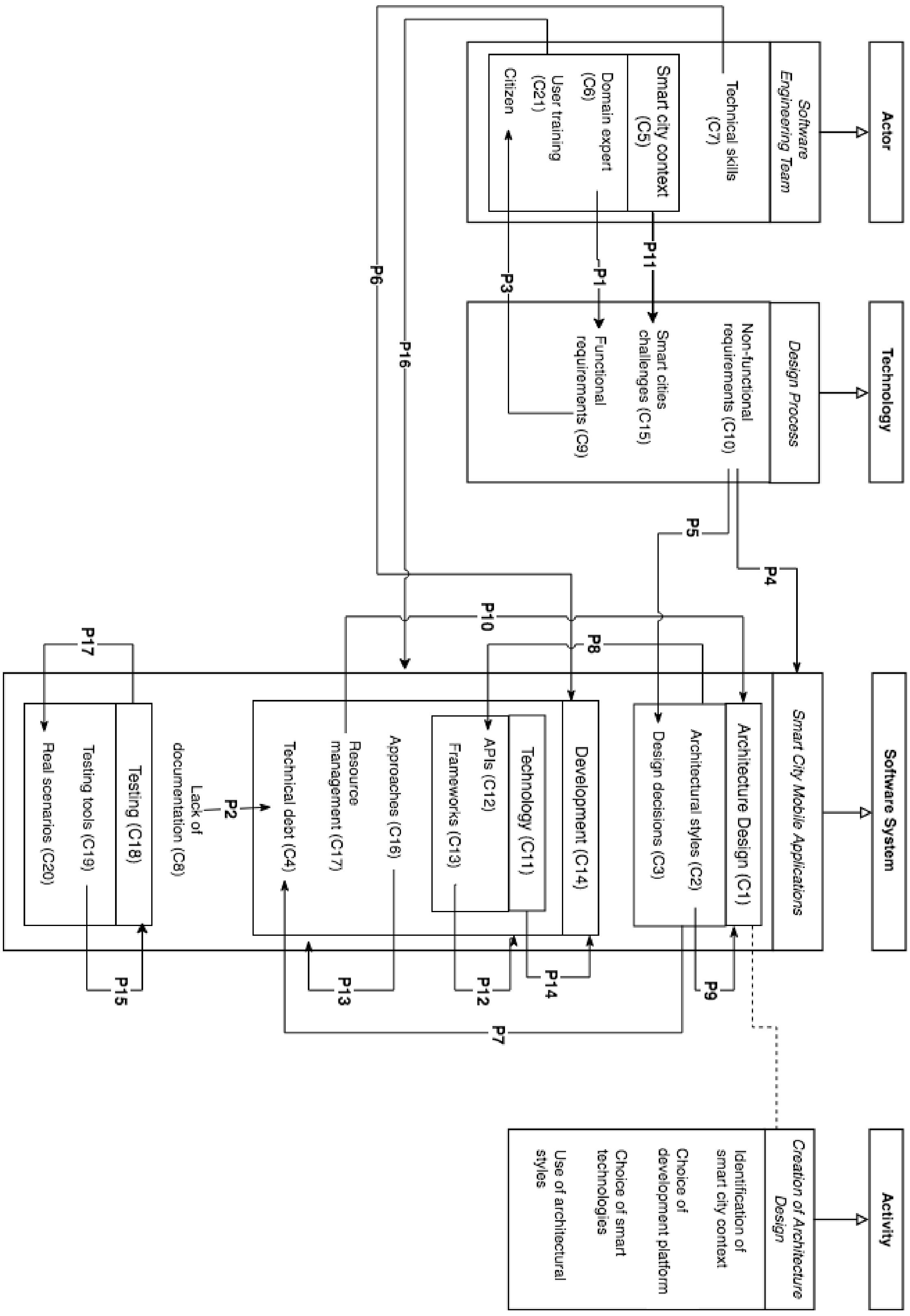

Actor
Software Engineering Team
Technical skills (C7)
Smart city context (C5)
Domain expert (C6)
User training (C21)
Citizen
Technology
Design Process
Non-functional requirements (C10)
Smart cities challenges (C15)
Functional requirements (C9)
Software System
Smart City Mobile Applications
Architecture Design (C1)
Architectural styles (C2)
Design decisions (C3)
Development (C14)
Technology (C11)
APIs (C12)
Frameworks (C13)
Approaches (C16)
Resource management (C17)
Technical debt (C4)
Lack of documentation (C8)
Testing (C18)
Testing tools (C19)
Real scenarios (C20)
Activity
Creation of Architecture Design
Identification of smart city context
Choice of development platform
Choice of smart technologies
Use of architectural styles
P1
P2
P3
P4
P5
P6
P7
P8
P9
P10
P11
P12
P13
P14
P15
P16
P17

Figure 4.1 Theory representation of how software engineering teams design mobile apps for smart cities

Explanation

The theory cuts across the 4 major divisions in the theoretical framework. Figure 4.1 describes our theoretical model by showing the inter-relationships among the constructs and propositions. The lines in the figure labeled *P(n)* show the specific connections between constructs and the arrows show the direction of implication between the constructs. These relationships were identified from interview data provided by the app developers.

The figure shows that SE teams building smart cities mobile applications have focused primarily on speeding up development under the pressure of time to market thus neglecting important software architecture principles in the application development process. Although they do not follow any guide to design their application, architects and developers have reduced the development complexity by adopting architecture specification frameworks, creating faster and more portable apps by choosing a specific development approach, reducing time to market by adopting agile methodologies to mobile platform development, and satisfying some NFRs by using architectural patterns and styles in their apps.

On the other hand, the software design effort has suffered from inadequate testing processes due to the difficulty in simulating real scenarios required in the smart cities context, and the lack of documentation of the process, which have caused technical debt to be accrued.

4.3 DISCUSSION OF THE GROUNDED THEORY

By cross analyzing the core categories in the coding process, we identified some relationships that helped to hypothesize a theory about how SCMA are developed and what characteristics in the software development process influence the architecture design of these applications. In general terms, we found that the construction of the architecture for a SCMA relies on several factors such as the priorities among quality attributes, completeness of the non-functional requirements (NFR), project domain, technologies, a development approach that was adopted, and others.

We will discuss the generated theory from four main directions: Quality attributes and the resulting NFRs for mobile applications, smart cities context, architectural patterns and styles, and candidate anti-patterns of architecture construction. To clarify, the proposition numbering *P(n)* is a result of the coding process in the grounded theory method and reflects the table of propositions (recall Table 4.3).

4.3.1 Quality attributes and the resulting NFRs for Mobile Applications

Development teams building mobile apps take into account some of the quality attributes (QAs) inherent in the context of smart cities in the construction of the architecture, but the degree of mobility needed for mobile apps affects the priorities of the QAs. In Table 4.4, we catalog the five QAs most related to mobile apps as reported by the participants in the interviews.

We noticed that all participants whose role in the project was either architect or developer reported all the QAs listed in Table 4.4 for the following reasons: respondents

Table 4.4 How mobile application aspects are impacted by the QAs

	Battery Life	**Mobile Resource Management**	**Development Approach**	**Development Challenge**
Performance	✓	✓	✓	✓
Portability		✓	✓	✓
Privacy		✓		
Reusability		✓	✓	✓
Reliability		✓	✓	✓

claim that even apps for smart cities seem to be very simple and lightweight, but it is the large number of interactions among complex entities in smart cities that adds complexity to mobile applications (**P11**). For them, the back-end of these apps is very complex because they have to monitor extra sensors, analyze inputs from the crowd, integrate many components, communicate through APIs, and so on.

One QA reported by the participants was ***Performance***, one of the most relevant to mobile applications (WASSERMAN, 2010; BIØRN-HANSEN et al., 2019). The continuous use of GPS (*battery life*) used to get the user's information leads to high battery consumption. In order to mitigate this, the SE team made use of asynchronous tasks - computation that runs tasks on background threads (GOOGLE, 2018b) and limited the period in which GPS events are dispatched. For instance, *"(...) the user sets the time period in which they want to be tracked while walking in any dangerous place at the university"* (respondent #I18). Cross-platform development, a type of *development approach* in which developers generate the same app for multiple platforms from a single set of code, has an adverse effect on the performance of the app (NAGAPPAN; SHIHAB, 2016).

Building native applications for each mobile platform can be very expensive and so we expected ***Portability*** to be within the list of the most relevant QAs. Indeed, the interviews showed that developers of mobile applications have taken advantage of frameworks to develop hybrid applications as (*mobile resources*). For instance, *"We are thinking of adopting a hybrid approach to generate the native code for both platforms (Android and IOS) because as (the previous developer) has left the project, continuing maintenance with both platforms will be very costly"* (respondent #I16).

On the other hand, experts who master all programming knowledge required for all platforms are hard to find. Projects have invested in cross-platform development to attract users from different platforms, but cross-platform development brings with it development issues related to performance, unexpected behavior of the application because of feature interactions, hardware limitations, and problems with different versions of the platform. Our theory explains that the level of development challenges in SCMA depends upon the development approach adopted (**P13**), but a deeper study is needed specifically on the use of multi-platform approaches for SCMA to provide strong evidence of these challenges and solutions.

Reliability is a concern since predominant features for smart applications are receiving/analyzing/reporting data from several users automatically as soon as the data arrives. The apps need to ensure the reliability of their data (*Development challenges*), especially for emerging occurrences by analyzing fake inputs from the crowd and developing alternative solutions: *"I implemented a listener that keeps monitoring the data flows in case of unavailability of the Internet, wrong data types, and late data for potential mitigation as quickly as possible"* (respondent #I9).

The cross-platform *development approach* also affects this QA since the development is heavily influenced by the quality of the functionality available in third-party libraries.

A trade-off was identified between ***Reliability*** and ***Privacy*** when developing an SCMA and the architect needs to prioritize one over the other based on the objectives of the app. We identified both situations in our study. Basically, the satisfaction of non-functional requirements depends upon the design decisions made (**P5**). SCMA, just like any real-world application, has conflicting non-functional requirements and the architect will have to make trade-offs among them.

On one hand, the architect only uses the data that is extremely relevant to the system and maintains user privacy: *"Since it is a crowd-sourcing app, we can not save information from any user. Actually, it would be good to have this information as to avoid pranks (in the system), but we decided that every GPS position would be anonymous (...) privacy in this context of smart cities is fundamental to preserve the user's own privacy"* (respondent #I5).

On the other hand, the architect decides to reveal the user identify so that the system's reliability is enhanced: *"When an unknown person uses the app, they are informed about what information will be used. (...) we have few pranks because the user is aware that they will be identified by our monitoring center"* (respondent #I19).

Finally, ***Reusability*** is a prevalent QA for mobile applications. Developers reuse libraries and components within an application, as well as across platforms (e.g., Android and IOS). Our findings give support to the work of (MOJICA et al., 2014), in which they compared the extent of reuse on hundreds of thousands of Android apps and revealed that overall 84% of mobile apps are completely reused by another mobile app. According to respondents, the two biggest reasons for them to reuse code were to increase the productivity of mobile app projects and to integrate SCMA with smart city frameworks. For example, *"The architecture of the app (A3) was somewhat based on the architecture of (A2), which relied basically on Fiware[1] platform"* (respondent #I16). At the same time, SE teams have faced development challenges regarding reuse, such as refactoring, either before reusing or afterward due to the large effort to integrate a reused class.

4.3.2 Smart cities context

The smart cities context brings challenges to all phases of the development process of SCMA, and we discuss here some of the main challenges reported by the SE team.

`Requirements`: Extracting and understanding requirements for mobile applications is not a trivial task, and a number of studies have focused on requirement extraction for mobile applications (NAGAPPAN; SHIHAB, 2016). While SE teams for SCMA also

[1]Fiware is available at https://www.fiware.org

extract topics from user reviews to revise requirements, our study also identified that the elicitation of requirements of SCMA depends heavily on the experts (**P1**) in the application domain who, based on citizen's needs (**P3**), act as partners justifying the need for the application, distilling the features, explaining important aspects related to this domain, and validating it.

"In a report about fire, firefighters taught us that the color of the smoke is important, (..) they explained that there is a lighter and a darker gray and (identifying this difference) helped us to understand the type of fire to be treated" (respondent #I2).

`Design`: The smart city context also brings challenges to the design of SCMA. The complexity due to the number of interactions among complex entities in smart cities led SE teams to adopt smart city frameworks (**P12**). *Fiware* was the framework adopted for the majority of the applications. It is a generic, extensible platform able to cope with the essential requirements in smart cities, and which has been used in Europe successfully in several cases. Development teams have created a wide range of smart city applications using the platform (FIWARE Foundation, 2018). Many benefits were listed by the participants who adopted this platform, including that (1) it becomes the core of the app's architecture since it establishes communication among all the components, (2) it offers a greater facility to integrate components and applications, and (3) provides a set of Application Programming Interfaces (APIs) that facilitate the designing of such applications. *Fiware* provides several architectural styles that address specific NFRs. *"One of the advantages that we had using the Fiware platform was that (it) had components that dealt with security already within the context of the platform"* (respondent #I3).

`Development`: Mobile applications, especially crowd-sourcing apps - apps that obtain information from a large number of people voluntarily undertaking a task of mutual benefit (NANDAN; PURSCHE; ZHE, 2014) - pose challenges to software development. Respondents reported issues related to network coverage. Since such apps require connection to the Internet and in some areas the network coverage is poor, developers might need to implement alternative solutions to deal with the availability of the app. *"We had a very big challenge which was the WiFi signal that worked very well inside the mall but in the parking lot (it's six floors of parking) was very bad. How were we going to map the routes if there is no signal?"* (respondent #I11). The use of Google/Apple maps API, beacons, and triangulation were solutions studied by the participants. Moreover, real-time data in this context implies that information is only true at the time data arrives and if it arrives at the right time. For example, in the context of the application *A1*, where the crowd sends reports about emergency situations in real time, delayed information may lead to the problem not being solved. Therefore, aiming to mitigate problems related to software development of SCMA and hence speeding up its development, SE teams make use of smart technologies (**P14**).

`Testing`: Finally, testing SCMA involves several factors. Respondents reported that it is exhausting, takes a lot of time, and there is inadequate support by testing tools for mobile applications (**P15**). The lack of testing tools for mobile apps, a gap confirmed by other researchers (FRANCESE et al., 2017), becomes even more serious in the context of smart cities where testing also involves the need to simulate real-world scenarios (**P17**). It is difficult to construct test scenarios with the same challenges as the real operational environment.

"The problem is not the development of the app itself, the big problem in these tests are the people (who take action) in response to some emergency reported by the app" (respondent #I18).

This quote raised another testing challenge which is training users who will use the app to support a citizen who will report a problem using the app (**P16**). For instance, *"There was a lot of training with the university security forces because they were resistant to changing their way of working, and most of them were already retiring and were not adapted to mobile technology."* (respondent #I14).

4.3.3 Architectural patterns and styles

As previously discussed in Section 2.1, mobile applications follow the architectural principle of building blocks, which was derived from traditional software architecture practice (BAGHERI et al., 2016). In this context, our study reveals that although architects and developers of SCMA have not adopted any specific architectural pattern and most of the time the application architecture is not planned, they are likely to use good development practices, such as modular programming, layered architecture, and component-based software engineering. This is due to the need to fit their code into frameworks, which are based on architectural styles. Syer et al. (2011) found that Android apps rely heavily on the Android platform and its architectural styles.

The preference for one style over another depends on how the architecture is designed, but in general, they are API-Centric (**P8**). *"The apps of our group (T2) are a bit similar because there are a lot of API calls in them."* (respondent #I10).

SCMA used some architectural styles that are very common in smart city applications, and the use of such styles have positively impacted the design of these applications (**P9**).

One of the most predominant architectural styles was **Client-Server**. It may be because SCMA have clear separation between clients and servers. Picking the application *A2* as an example, the citizens using the mobile app to report any obstacles in bike paths and to receive information about weather conditions are on the client side while *Fiware* is on the server side. Using the Client-Server style positively impacts the architecture in the following ways (BOSCH, 2000):

- Scalability and Portability. Each component can evolve separately without affecting others;
- Maintainability. Stable protocols lead to well-defined and reusable components;
- Security. Security layers can be added to evaluate external events.

Publish-subscribe, a type of implicit invocation style, was also identified in SCMA. In this style, components communicate through asynchronous messages allowing better integration among them. Indeed, the publisher object, when performing internal processing of a routine, checks for objects that have subscribed to that event and notifies them immediately. *"We use publish-subscribe to automatically update the data when some data* *is updated on the server, the app receives a notification, we treat this notification and perform the internal update of the application."* (respondent #I17). We could observe from the interviews that architects/developers utilize this style through the use of frameworks. Examples reported by the participants were Fiware and Firebase[2]. This style strongly impacts the *reliability* of the system since the implicit invocation style is centralized, and so it more easily deals with unexpected events, thereby improving reliability; as well as *maintainability* due to the independence of the components.

Lastly, the **message broker** style was also used to ensure a more efficient communication among the components. *"The choice of communication by broker had excellent results, the performance was exceptional."* (respondent #I4). The message broker is responsible for receiving a message from one component and making it available to others, and so it was used for validation, transformation, and routing of messages. Two of the teams (*T1, T2*) that used this style relied respectively on *RabbitMQ*[3] - a open-source software message broker software, and a *Fiware* component called *Orion Context Broker*.

4.3.4 Candidate anti-pattern on architecture construction

We noticed some situations within the SE teams that negatively impact not only the design of the application, but the entire development process. The first evidence of the candidate anti-pattern[4] that emerged from our theory is that the lack of documentation creates technical debt (**P2**). Contrary to traditional software development, which is more focused on specification documents, developers of mobile applications are more likely to adopt agile practices (FRANCESE et al., 2017). However, reviewing the archival data we mostly saw high-level technical reports (except for *T1* which is a research and innovation project), that do not specify any feature nor how the design decision were made (**P4**). The problem appears when it comes to testing the application and it is not possible to track the requirements to specific tests or the person who holds the tacit knowledge leaves the project: *"I was developing this part, but as I ended up leaving the project, this ended up being discontinued."* (respondent #I7).

The other evidence of a negative impact is that failing to implement architecture decisions creates technical debt (**P7**). To get products to market faster, mobile application development teams usually have a short period in which to deliver the product (FRANCESE et al., 2017). Many developers prefer to meet time-to-market requirements at the expense of robustness: *"We did not create an architecture before we developed the application because of the short time we had."* (respondent #I13). Mobile applications developed for smart cities are no different. Since the demand comes from citizens' needs to be protected (**P3**), delays can be life-critical. All of the apps included in this study were directly related to life-critical situations such as personal protection or emergency response. Therefore, the faster "solution" that SE teams find is neglecting the architecture design. The consequence of this common reaction is the creation of technical debt

[2]Firebase is available at https://firebase.google.com

[3]RabbitMQ is available at https://www.rabbitmq.com

[4]We use "candidate" here to reflect the Patterns community's belief that there be multiple published instances of a design before it is referred to as a pattern.

on the project. That is, there is an implied cost of additional rework caused by choosing an easier alternative instead of using a correct approach that would take longer.

4.4 A DEEPER LOOK AT THE ARCHITECTURE OF SELECTED ANDROID APPS

We begin with a multi-level description of the Android apps' architecture based on a previous study (BAGHERI et al., 2016) and then we present an in-depth description of one specific app.

4.4.1 Multi-level analysis

From a qualitative data analysis of nineteen interviews, we can began to understand which characteristics describe and influence the architecture design of SCMA in terms of design decisions that architects make, architectural styles they use, and limitations of mobile devices that may affect the prioritization of non-functional requirements, among other aspects discussed in the previous sections. However, this study also raised a question about the structure of the architecture for mobile apps: *What does the android style, which multiple study respondents mentioned, look like?* Therefore, we decided to take a deeper look at the architecture of the set of selected apps, which were either available in the Google Play Store or we had access to its APK file.

Our architectural analysis was intended to investigate the effectiveness of the architecture design performed by the teams using agile methods. To ensure accuracy we needed to compare the as-designed and as-built architectures of each app. Data collected via interviews with the development teams provided the as-designed information. Reverse engineering from each app's source code provided the as-built architecture. This cross validation allowed us to investigate the extent to which the resulting app actually utilized specific Android styles, components, and connectors.

The results of these analyses are similar to those found in a study by Bagheri et al. (2016), in the sense of using a wide variety of components and connectors, which contribute to the success of Android apps. However, most of the components were derived from a small set of components provided by the Android framework. Table 4.5 shows the average number of component instances and connector usages in each app. While BAGHERI et al. found that the ranking of component types in frequency from highest to lowest is: *Activity, Content Provider, Service*, and *Broadcast Receiver*; we found *Activity* and *Service* are the most used components in our small sample of apps from the Google Play repository. Another difference is that we found the number of ports in some of our apps was well above the average for Android apps. This might indicate that the mobile applications analyzed in this study have more data flows than the average Android app.

The high number of components and connectors results from Android apps using more than one architectural style. We confirmed the use of four out of five styles identified in Bagheri's study. Table 4.6 depicts which ones were identified in our set of apps.

We noticed that all six app architectures have used the *publish-subscribe* style instead of the *message-based explicit-invocation* style for sending a group communication

Table 4.5 Number of components and their provided ports of the reverse engineered apps

App-Team	**Components**				**Provided**
	Activity	**Service**	**Provider**	**Receiver**	**Ports**
A2-T2	9	1	0	0	18
A3T-2	7	2	1	2	9
A4-T2	6	2	2	1	6
A7-T3	4	8	1	5	14
A8-T3	13	1	0	0	114
A9-T4	17	1	0	0	55

Table 4.6 Architectural styles identified in the reverse engineered apps

	message-based explicit-invocation	**message-based implicit- invocation**	**publish-subscribe**	**shared state**
A2-T2	✓		✓	
A3-T2	✓		✓	
A4-T2	✓		✓	✓
A7-T3	✓		✓	
A8-T3	✓		✓	
A9-T4	✓	✓	✓	

(except for *A9T4* that used both), which makes the architecture more efficient and elegant (BAGHERI et al., 2016). This result confirms the findings of the interviews which suggested the use of the *publish-subscribe* style made the communication between components more efficient.

4.4.2 Example Reconstructed Architecture

In this section, we present a case study using one app to illustrate the concepts investigated in this research. We reverse-engineered the architecture of the mobile app **A8** using the architecture description language ACME[5]. Then we give an in-depth description of the structure of **A8** and the reader will see how the architectural principles identified in this research can be applied to a specific example.

Figure 4.2[6] shows the reverse-engineered app architecture of **A8** at a very coarse grained level. This level of representation helps to see the typical Android architecture and supports information captured from the interviews.

The system is largely divided into service components and activity components. The architecture shows a high degree of connectivity as discussed earlier. From the feedback interview with the developer of **A8**, we could differentiate which components were implemented by developers, and which components were provided by the Android framework. For example, the components *LocationService, UpadateRatesIntentService, My-*

[5]ACME Studio is available at http://acme.able.cs.cmu.edu/AcmeStudio

[6]Figure 4.2 was created using the ACME Studio tool. A full tutorial of the architecture reverse-engineering and recovered architectures are available in our complementary material.

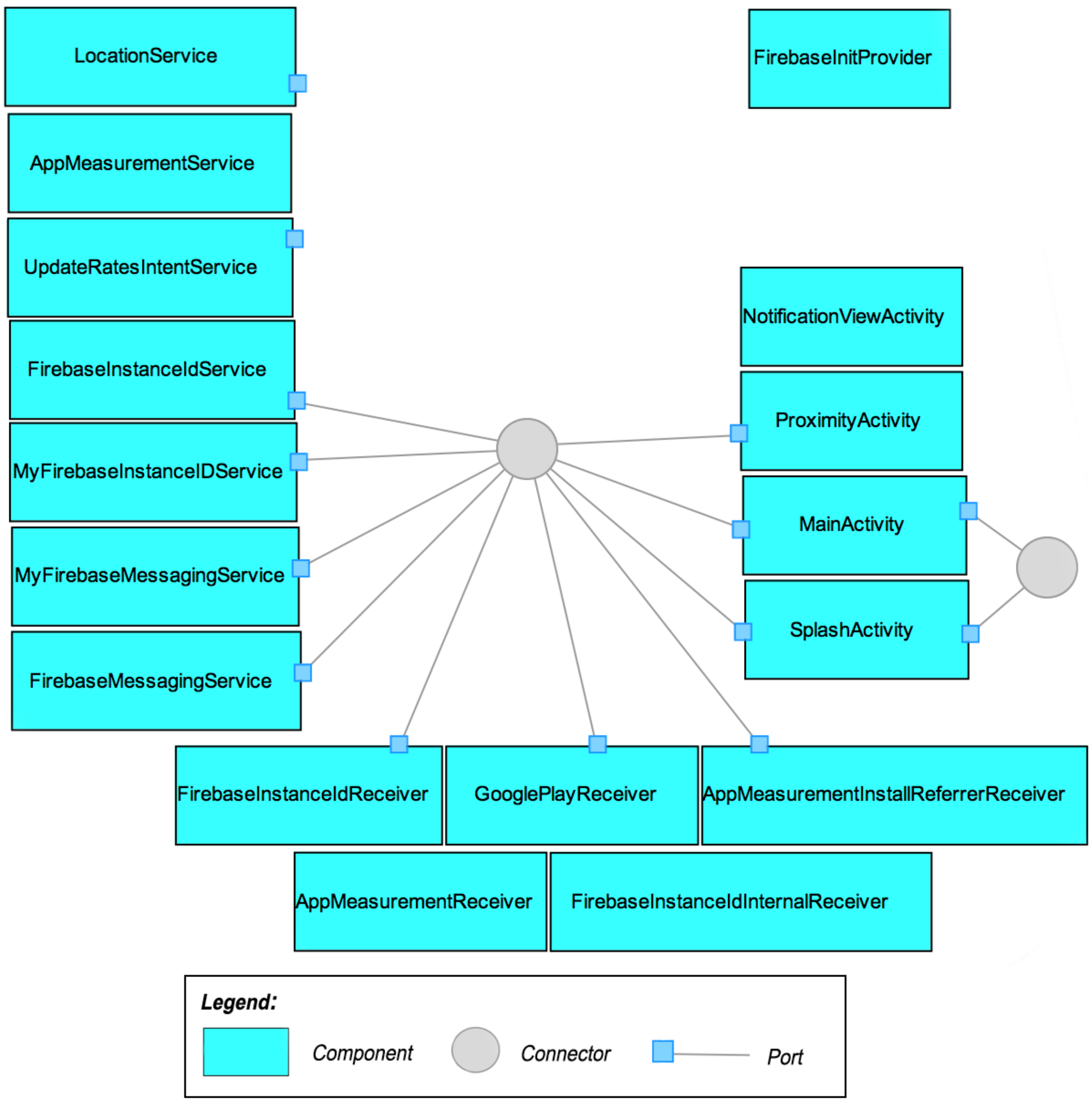


Figure 4.2 Recovered architecture of the application *A8-T3* using the architecture description language ACME

FirebaseMessagingService, MyFirebaseInstanceIDServer, ProximityActivity, NotificationViewActivity were implemented by the developers. However, *appMeasurementService* and all components of the type *Receiver*, in the bottom part of the figure, are components provided by the Android framework.

We used the reverse-engineered architecture to confirm information from the designer about the connections as actually implemented. For example, *"It [the recovered architecture] is correct. The connection between the components MainActivity and SplashActivity is because the SplashActivity calls the MainActivity component directly."* (respondent #I17, developer, app A8).

We also identified that *FirebaseInitProvider, FirebaseInstanceIdService, and FirebaseMessaginService* are components of the Firebase framework, which was discussed in Section 4.3.3 as a framework frequently used for the respondents to implement the architectural style *publish-subscribe*. Therefore, we were able to identify and verify the use of, this architectural style through this framework.

Overall, the recovered architectures have a high degree of connectivity among numerous components (at the activity component level) and do not explicitly show the exact styles that were adopted in the app, but the styles used could be identified from the representation. Study participants found it very helpful to see how their app is structured, how components are interacting with others and felt they would make better decisions on the architecture design of their applications using this information.

4.5 THREATS TO VALIDITY

In this section we discuss several potential threats to the validity of our results with their corresponding mitigating factors according to Runeson and Höst (2008), which distinguish four criteria for validity:

Construct Validity

Our study is mainly threatened by a list of architectural patterns and styles, which are available in our complementary material[7], we provided the list to the participants to be used as reference. This data could have biased their answers. However, the list was made by considering surveys of the software architecture literature (TAYLOR; MEDVIDOVIC; DASHOFY, 2009; ROMAN; PICCO; MURPHY, 2000; GARLAN, 2000; SHARMA; KUMAR; AGARWAL, 2015; ZHANG et al., 2016). Moreover, we conducted a pilot study to adjust the time required to perform the interview, the quality of our questions, and thus mitigate any construct threats.

- Internal Validity

Misinterpretations of the interviews is an internal validity issue we need to consider. The data collection could have been biased by the researcher's opinion. Regarding that, a validity procedure for the multi-case study was already discussed in Section 3.6. Also,

[7]https://rose2s.github.io/EMSE2019

the recovered architecture obtained from the reverse-engineering of Android apps was validated with each respective developer using a feedback questionnaire (recall Fig. 3.1). The profiles of the interviewees were very diverse, each one played a different role on the project and the majority had more than 4 years of experience in the team. This diversity reduced the probability of misinterpretations. Finally, besides the discussion on the criteria to select the systems (recall Section 3.1), budgetary restrictions also were a constraint on which apps to study and how to collect the data.

- External Validity

The main threat to external validity is the non-generalizability of the study to the whole domain of mobile applications being developed for smart cities. However, to the best of our knowledge, the best place to look for SCMA is in research and innovation groups. By selecting nine mobile applications from four different locations we certainly found applications that varied by size, methodology, objective, team, and design process. Although innovation groups do not represent a cross-section of the app industry, the applications were all related to public services and most of the development teams had partnerships with cities.

- Conclusion Validity

The sensitivity of GT is a threat since this method is very subjective and depends on how the researcher sees the problem. To mitigate this threat, another student from the same research group performed the entire analysis process separately from the coding process until the identification of the constructs and the depiction of propositions; then every step was discussed, refined, brought to a consensus, and results were merged to shape the theory.

Moreover, according to Adolph, Hall and Kruchten (2011), confirmability of the study is a good criteria to evaluate the theory for rigor. It argues that conclusions depend on subjects and conditions of the study rather than the researcher. Then, by involving the study participants in several phases of the research, potential biases or influences present in the researchers conducting our study were eliminated. Lastly, we provide a supplementary material that will help other researchers to replicate this study and test our theory under changes in the study components.

4.6 CHAPTER SUMMARY

In this chapter, we presented the central result of this research effort. The theory described in this chapter was developed on top of data collected from people with first hand knowledge of the project and the resulting software. The use of a grounded theory approach to model development took advantage of these data sources to identify a set of fundamental constructs and then to capture relationships among the constructs in the form of propositions. A detailed explanation of each proposition in Figure 4.1 is discussed within the text in Section 4.3. The model provides an initial resource for explaining the choices made by teams developing SCMA and for reasoning about scenarios regarding the implications of changing the development process. In the following chapters, we provide a description of the evaluation of the model.

Chapter
5

THE EVALUATION OF THE INITIAL THEORY

In this chapter, we describe the evaluation process of the grounded theory. To the best of our knowledge, papers that use GT usually define their own evaluation method for their theory. Some researchers have even applied only single evaluation tasks (HODA; NOBLE; MARSHALL, 2012; SOUSA et al., 2018). However, for the purpose of making our evaluation process more robust, we applied a set of 5 tasks based on techniques from the GT literature (ADOLPH; HALL; KRUCHTEN, 2011; STOL; RALPH; FITZGERALD, 2016) and the researcher's experience. The 5 steps are defined in Figure 5.1.

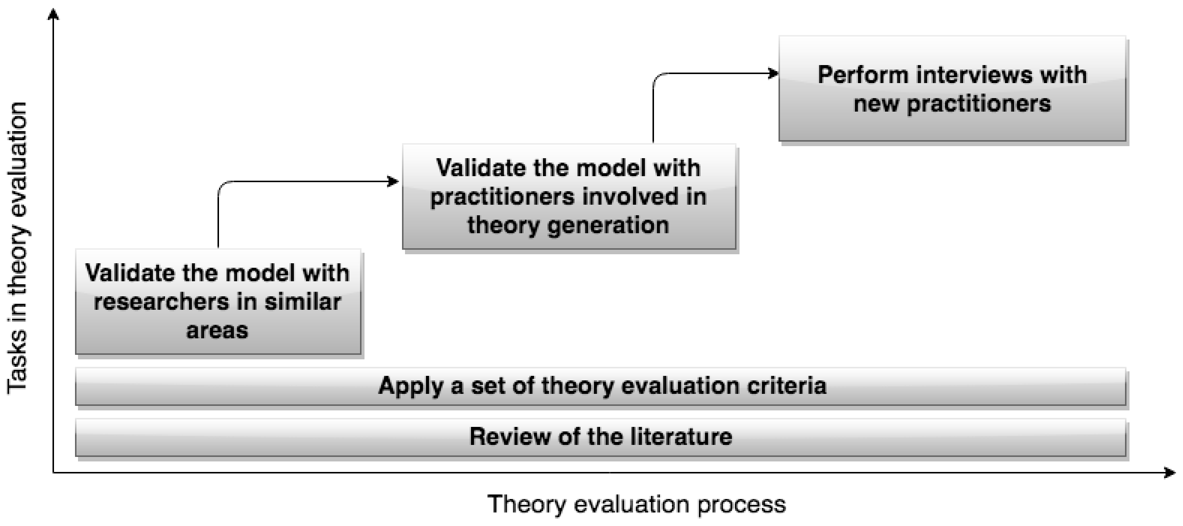

Tasks in theory evaluation
Perform interviews with new practitioners
Validate the model with practitioners involved in theory generation
Validate the model with researchers in similar areas
Apply a set of theory evaluation criteria
Review of the literature
Theory evaluation process

Figure 5.1 Theory evaluation process design

The two tasks shown at the bottom of Figure 5.1 were applied periodically during the evaluation process. The three remaining activities were performed sequentially so that each task provided insights that contributed to the following activities. In Section 5.1, we describe each task and its respective results when applied to our theory are presented in Section 5.2.

5.1 THEORY EVALUATION TASKS

In this section, we describe each of the five tasks in the theory evaluation process.

Review of the literature

The literature review used in this study was conducted in two parts. The first one was an exploratory research study within the current knowledge landscape of smart city mobile applications. The second phase was intended to determine how well the generated theory fits with the existing literature on the subject as recommended by Adolph, Hall and Kruchten (2011) and Hoda, Noble and Marshall (2012). To that end, it was focused on relating the research findings to the literature through the integration of ideas.

Apply a set of theory evaluation criteria

Sjøberg et al. (2008) specified a set of criteria for evaluating grounded theories. Next, we list these criteria and give a very brief evaluation of our theory relative to each criterion:

- **Testability -** The degree to which a theory is constructed such that empirical refutation is possible.
- **Empirical Support -** The degree to which a theory is supported by empirical studies that confirm its validity.
- **Explanatory Power -** The degree to which a theory accounts for and predicts all known observations within its scope.
- **Parsimony -** The degree to which a theory is economically constructed with a minimum of concepts and propositions.
- **Generality -** The breadth of the scope of a theory and the degree to which the theory is independent of specific settings.
- **Utility -** The degree to which a theory supports the relevant areas of the software industry.

According to Adolph, Hall and Kruchten (2011), there are other criteria we need to take into account when evaluating a theory for rigor. They addressed the grounded theory rigor issue with two questions:

1. Is the story expressed in the theory a true story and not a fabrication?
2. Is the theory a good story, one people will find interesting, adds to the known body knowledge and is useful for informing policy?

These two questions address aspects of a theory that can be effectively compared to the following more general software engineering qualities:

- **Transferability -** How far can the findings/conclusions be transferred to other contexts and how do they help derive useful theories?
- **Confirmability -** Conclusions depend on subjects and conditions of the study rather than the researcher.
- **Dependability/Auditability -** The study process is consistent and reasonably stable over time and among researchers.
- **Credibility -** The research findings are credible and consistent.

Validate the model with researchers in similar areas

One of the criteria employed to validate rigor in qualitative grounded theory studies is fittingness, also known in the literature as resonance (CHARMAZ, 2006). In order to verify that our theory makes sense to others, we decided to present our emergent model to other software engineering researchers. In addition, the contributions of this work were evaluated by experts through a paper submission to an EMSE Special Issue on "Software Engineering for Mobile Applications".

Validate the model with practitioners involved in theory generation

According to Stol, Ralph and Fitzgerald (2016), resonance is one of the criterion when evaluating grounded theory studies. It addresses questions such as: does our grounded theory make sense to our participants? Similarly, Sousa et al. (2018) stated that an important step in theory evaluation is validating the model with practitioners involved in theory generation. We decided to perform an agree/disagree survey with the practitioners involved in the theory generation as to appropriately evaluate this criterion.

Just to recap, all the teams selected to participate in this study focus on innovative software solutions by supporting entities related to public services, such as public safety, tourism, transportation, education, or smart buildings. A detailed characterization of each subject was presented in Table 3.1 and a description of each app was also detailed in Section 3.2.

Since we already had interrupted their work at least twice for this study, this time we were not able to interview them, instead we used an online survey to get quick feedback from each of the team members.

The survey basically consisted of the 17 propositions (recall Table 4.3), followed by a short description of each, and the following options from which they had to select: (1) agree, (2) disagree or (3) neither agree nor disagree (shorted in this work to NAND) and the option of writing a short justification for their answers.

From 19 stakeholders interviewed, 13 answered our request (almost 70%), although 2 of the responding stakeholders emailed us back justifying the reason why they would not be answering the survey. We had a total of 11 respondents. Respondent #I6 decided to not respond the survey because her/his role was researcher, not active participant, on A1-T1, and the respondent #I14 had left the project by the time we sent the survey.

Perform interviews with new practitioners

Finally, following the example presented in Hoda, Noble and Marshall (2012) for evaluating a grounded theory by performing interviews with new practitioners, we selected another company to perform new interviews. The main selection criterion was having the same as the previous ones, but having a greater potential market given the size of the team and the timing with respect to the targeted market.

We performed two interviews following Runeson and Höst's guidelines (RUNESON; HÖST, 2008), with a duration of 30 minutes each, based on the same protocol as the previous interviews. Our conduct of these new interviews was informed by the feedback from the previous tasks detailed in this chapter. The questions followed the same idea of the survey but the order of the questions were based on the groups of priorities: critical, moderate and desired propositions.

- Characterization of the subjects

A characterization of the new subjects, I20 and I21, is detailed in Table 5.1. This includes their interviewee IDs, I20 and I21, their roles in the project, years of experience, the application with which they were involved (A10 and A11), and their respective development team (T5).

Table 5.1 Role and experience of the respondents

#	Role	Experience (years)	Application	Development Team
I20	Solution Architect	10+	A10	T5
I21	Software Architect	10+	A11	T5

- Characterization of the apps

Next, we present a short description of applications A10 and A11.

T5: `A public institution of the government of Bahia`

- `A10` is a digital customer service system for the Government of Bahia. Its main purpose is to consolidate the use of all government digital services in one place[1].
- `A11` offers a quality service in finding the best profiles to be outsourced for the Government of Bahia. It has a smart selection criteria with professionals who access the app, through the application of several tests[2].

5.2 DISCUSSION OF THE EVALUATION TASKS

In this section, we discuss the main findings of the five tasks in the theory evaluation process.

[1]A10 is available for download at shorturl.at/fVWXZ
[2]A11 is available for download at shorturl.at/gsI57

5.2.1 Review of the literature

Although the development of mobile applications for the smart city domain is a more restrictive subject than the general notion of mobile apps across all domains, by evaluating our results in the light of previous work in the literature we can highlight the following:

– Ivan et al. (2009) stated that citizen-oriented applications must be orientated towards citizen satisfaction, delivering high quality public services.

– We have seen in this work the need to better manage the mobile resources for SCMA. Due to the fact the battery is a scarce resource for those apps, several studies have proposed ways to measure and to save energy used for mobile apps (NAGAPPAN; SHIHAB, 2016).

– Nagappan and Shihab (2016) reported that more work is needed on automated testing of mobile apps, specially for cross-platform apps. This gap was also reported by other researchers (FRANCESE et al., 2017). Similarly, our work confirmed that the lack of automated testing tools for mobile apps is a concern that needs more investigation.

– Mobile development teams have often adopted cross-platform development frameworks (FRANCESE et al., 2017; BIØRN-HANSEN et al., 2019).

5.2.2 Apply a set of theory evaluation criteria

Next, we give a very brief evaluation of our theory relative to each criterion described in Section 5.1:

- **Testability -** Our theory has an acceptable level of testability since empirical refutation of its propositions is possible by replicating the study.

- **Empirical Support -** Considering the brief discussion in Section 5.2.1, we consider that the propositions of our theory are reasonably supported by other empirical studies. Even though they have a slightly different focus than the study on which our theory is based, some of the findings of this dissertation agreed with the findings of previous studies. Conducting additional studies is part of our planned future work.

- **Explanatory Power -** We consider the explanatory power of the theory to be adequate given the narrow scope of the study. Part of our future work is to expand the scope of the theory.

- **Parsimony -** We consider the parsimony of the theory to be moderate given that our original efforts focused more on roughing out a theory and less on the efficiency with which it is expressed.

- **Generality -** The current scope of this theory is narrow, therefore we consider the generality of the theory as low. Our planned future work will address this by broadening the scope.
- **Utility -** While the scope of the theory is narrow the study was very focused and hit its intended audience. We consider the utility of the theory to be high.
- **Transferability -** Our theory is transferable within the context of our targeted domain of smart cities.
- **Confirmability -** By involving the study participants in several phases of the research, any biases or influences present in the researchers conducting our study were eliminated.
- **Dependability/Auditability -** Although no other researchers have replicated our work yet, there was a high level of dependability among the results of the various study activities.
- **Credibility -** Our theory has a high degree of credibility due to presenting various results of the research back to the study participants and adjusting the theory to reflect their input.

5.2.3 Validate the model with researchers in similar areas

The model discussed in chapter 4 (recall figure 4.1) was presented to the Reuse in Software Engineering (RiSE) research group at Federal University of Bahia and also two professors who work with Software Architecture and Testing in Software Engineering. As a result, we got a lot of good feedback on our model representation allowing us to reshape it, and include some insights on how to make it more robust. In addition, the professors made a valuable contribution by linking the topics of this dissertation with their students' work. This enables us to find additional studies that support our findings.

To exemplify this, during the presentation of the proposition P15, which initially was about the *lack of testing tools for mobile apps*, a professor whose area is Testing in Software Engineering pointed out several papers that report on available testing tools. The professor also agreed that the lack of *automated testing tools* is actually the biggest problem. Later on, while performing the evaluation of the theory by a survey with the practitioners involved in theory generation (section 5.2.4), we confirmed this deficiency with the participants themselves and refined that proposition for our model.

Besides getting feedback from research in similar areas we got excellent in-depth expert feedback from several researchers through a paper submission to an EMSE Special Issue on "Software Engineering for Mobile Applications". The reviewers' feedback provided input to our theory evaluation making the work more complete and useful for the research community.

Although they did not ask about the model propositions themselves, the reviewers raised an important question concerning the quality of the selected apps and asked for more details. The question concerned missing information about the acceptance of the apps by the customers, their success with respect to their goals, etc. The validity of the research results is connected with the quality of apps, in the sense that an unsuccessful app should be treated as a counterexample, showing practices and perhaps architectures that have to be avoided. To address the request of this reviewer, we created a subsection to discuss the quality of the selected applications (recall 3.3) and provide more information about the apps and their acceptance by the audience. Section 5.4 provides some recommendations for a SE team based on the good and bad practices we found in our study.

Another concern involved whether the scope was sufficiently well defined. The smart city mobile applications (SCMA) were selected based on a characterization of "smart city app." This could impact the quality of the model. To mitigate this issue we included in Section 3.1 a more detailed characterization of SCMA highlighting the specific criteria on which our selections were based.

Reviewers also asked about the sensitivity of the theory: *How sensitive is this theory? Would the results of the study be the same if the study components were exchanged for other similar components?* For example, if a different set of apps were chosen. Adolph, Hall and Kruchten (2011) argue that conclusions depend on subjects and conditions of the study and confirmability is an important criteria to evaluate the theory. By involving the study participants in several phases of the research, potential biases or influences present in the researchers conducting the study were eliminated. In the next sections (5.2.4 and 5.2.5), we describe the findings of validating the model with practitioners involved in theory generation and with new practitioners to apply the confirmability criteria.

5.2.4 Validate the model with practitioners involved in theory generation

As can be seen in Figure 5.2, we divided the results of this task into three groups of priorities: critical - the propositions with the highest percentage of disagreement between participants (between 20% and 50%), moderate - the propositions with some disagreement but also some agreement (between 50% and 70%) and desired - the propositions with the highest percentage of agreement among participants (above 80%).

We detail the results of the questionnaire by providing supporting statements by the interviewees who agreed with the proposition described P(n); NAND statements, which stands for neither agree nor disagree with the P(n) and disagreement statements of those who disagreed with each P(n).

- Substantial Disagreement (20% of agreement)

- **P4.** *Most non-functional requirements are not taken into consideration by the SE team when making design decisions.*

  `Supporting statement:` "Unfortunately, due to the pressure to deliver something functional quickly and lack of resources, most of the nonfunctional requirements are left to the end of the project or even, are not implemented" (#I12).

  `NAND statement:` "Some non-functional requirements are essential for mobile applications and therefore are prioritized (e.g. performance and scalability). Others

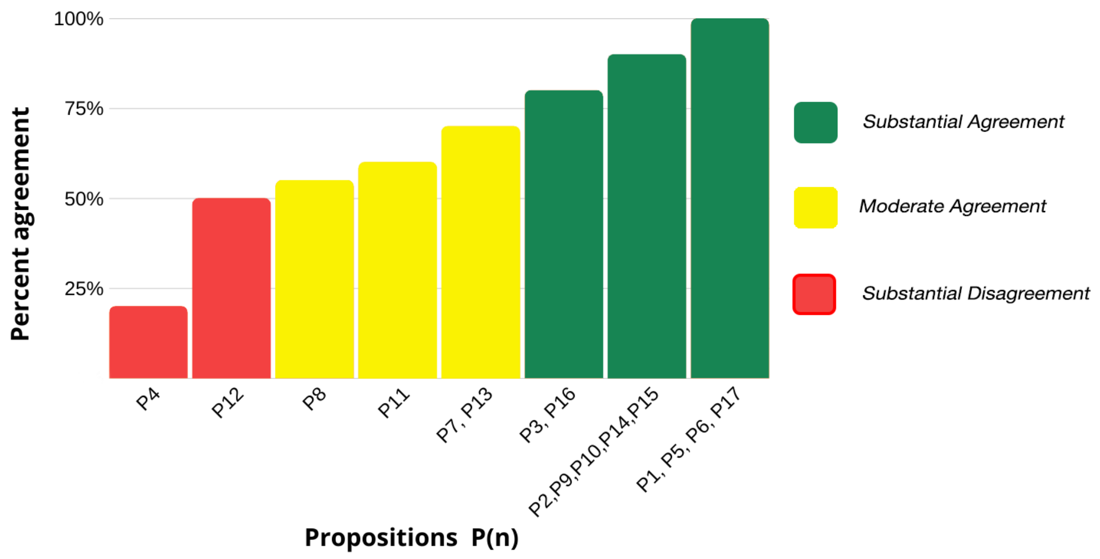


Figure 5.2 Percentage agreement gathered from the questionnaire about the propositions

end up in the background" (#I4).

`Disagreement statement:` "At the architectural level, there is clear concern about the non-functional requirements of the system" (#I5).

We concluded that NFRs were taken into account for the SE team if they were considered crucial to the app although some NFR are only used in the planning phase. In short, only those NFRs that affect the run time performance are considered. Long term properties such as maintainability are paid much less attention.

- **P12.** *The development or adoption of smart city frameworks reduces the complexity of the development of SCMA.*

  `Supporting Statement:` "I agree that the adoption of frameworks and libraries speed the development of applications and decrease complexity since it is encapsulated in the components of the framework itself and not in the application, which only calls them" (#I17).

  `NAND statement:` "This will depend on other factors to measure whether or not it actually reduces the complexity of development such as the learning curve for the framework in question, the community/company that maintains the framework, and the degree of technical knowledge of the team" (#I15).

  `Disagreement statement:` "The choice for a framework can bring advantages, such as code reuse. However, there is no guarantee that certain non-functional requirements will be guaranteed by the choice of the framework. On the other hand, a choice may even impact performance, for example" (#I5).

  We concluded some smart city frameworks for smart city applications may reduce the complexity of the development due to the ability of code reuse, reduced coupling, etc, but it will depend on many factors and because of that we can not guarantee a direct correlation. P12 needs to be refined.

- Moderate Agreement (between 50% and 70% of agreement)

- **P8.** *The architecture design of SCMA is API-centric.*

  `Supporting statement:` “Because SCMAs usually need to interact with platforms and applications, the use of APIs is a way to enable most of the architectural styles” (#I15).

  `NAND statement:` “Our solution was not based on APIs although we used them a lot” (#I5).

  We concluded that most of the interviews relied on the use of APIs to design their SMCA, except for #I5 since they built their own solution for app A1. P8 needs to be somewhat refined.

- **P7.** *The lack of architecture decision-making creates technical debt.*

  `Supporting statement:` “Any decision that is made to benefit one factor will be detrimental to another. For example, prioritizing development time will hamper the quality of the architecture, which will actually create technical debt” (#I12).

  `Disagreement statement:` “Architectural decisions will always exist, although they are not explicit sometimes. What can generate technical debt is the fact that developers do not follow the decisions made” (#I5).

  This is where the as-designed and the as-built system split off from each other. Given that the as-designed is the result of careful analysis, failure to use it results in a system with reduced quality and hence increased tech debt. Therefore, failing to implement architectural decisions will create technical debt.

- **P11.** *The smart cities context brings more complexity to mobile applications.*

  `Supporting statement:` “SCMAs are relatively simple, but the whole background needed to provide simplicity of the domain can get very complex to meet all the actors, variables and infrastructures required” (#I12).

  `NAND statement:` “This complexity depends on the context of the SCMA. For example, in many cases, interaction with another system can be very complex and in other cases, it can be very straightforward” (#I17).

  We concluded that the complexity comes with the degree of difficulty of the requirement and this domain usually needs to integrate with other components and sensors, use gateways to communicate between platforms, use a lot of third party technologies. Therefore, they together add complexity to the smart city domain. However, it is not exclusive to SCMA, there are other domains to which this proposition also applies, such as safety-critical systems.

- **P13.** *The level of development challenges in SCMA depends upon the development approach adopted.*

  `Supporting statement:` "The challenge will be measured according to the chosen approach, and it is up to the team to decide which approach to take" (#I12).

  `NAND statement:` "This may happen depending on the mobile application framework chosen to develop the app" (#I14).

  `Disagreement statement:` "The level of development challenges might be related to the complexity of the problem to be solved" (#I5).

  We noticed that the hybrid approach issue brought different experiences depending on the mobile application framework and the SE team needs to list the pros and cons of each approach before choosing one because it will strongly impact the quality of the application. Based on the answers, we decided to change P13 to the following: *the adopted development approach and its associated tools bring specific development challenges for SCMA.*

- Substantial Agreement (above 80% of agreement)

- **P3.** *Functional requirements for SCMA come from a smart cities context, which is citizen-oriented.*

  `Supporting statement:` "Citizens are the *owners* of the solutions; The citizen's need is often the starting point for the development of such applications" (#I15).

  `Supporting statement:` "The application must meet the needs of the citizen so their functional requirements should attend them directly. Besides, the specification of these requirements should be assisted by domain experts" (#I2).

  Since there were no opposite opinions regarding P3, we concluded it does not need to be refined.

- **P16.** *Training users positively impacts the product acceptance of SCMA.*

  `Supporting statement:` "We faced some resistance to get the agents and end user's acceptance of our application so that if we had invested more in training this issue would have been less" (#I15).

  `NAND Statement:` "When a SCMA is also a crowd-sensing app, there is no way to train the crowd but it may need another application for internal users to monitor/ analyze what the crowd sent as input. They definitely must be trained to know how to analyze the reports that arrive and how to help the citizens more quickly and effectively" (#I2).

  We noticed that the majority of the SCMAs have a team of internal users that use the mobile app or a part of it (dashboard, desktop app, a different view of the app) to assist the citizens and this group definitively needs to be trained.

- **P2.** *The lack of documentation creates technical debt.*

`Supporting statement:` “The time taken to contextualize/train a new stakeholder can be greatly reduced if there is at least a technical documentation, specially in small teams where there is a member turnover culture” (#I10, #I15).

`Supporting statement:` “The lack of documentation increases the number of meetings to discuss the scope of the project and generates momentary demands that may not be necessary for the project” (#I7).

`NAND statement:` “Our project had a lot of documentation but perhaps the stakeholders were not aware and did not have access to a large number of documents” (#I5).

We concluded that a project needs to have at least a technical documentation as to deliver a successful product with good quality within a reasonable time.

- **P14.** *Smart technologies positively impact the development speed of SCMA.*

  `Supporting statement:` “There is no need to re-develop the technologies adopted if they bring benefit we should use them” (#5).

  `Supporting statement:` “Having these technologies that aid SCMA development, we can focus more on developing business rules, and it positively impacts the speed of development” (#I5).

  Although new technologies will always request an initial time investment to learn them, smart technologies have grown their use lot in the applications developed for smart cities. We concluded that this proposition does not need to be refined.

- **P15.** *The lack of testing tools for mobile applications negatively impacts the testing of SCMA.*

  `Supporting statement:` “We still have a huge gap in finding tools and techniques focused on testing for SCMA. Today, in my opinion, it is the main bottleneck in the SCMA development process” (#I15).

  `Supporting statement:` “In many cases testing is still a stressful activity since the process is not fully automated” (#I17).

  `NAND statement:` “The problem I faced was the effort required to implement automated testing” (#I5).

  These results confirmed what we discussed in section 5.2.3 by validating the model with experts. We concluded that the lack of automated testing tools is seen as a wide problem for SCMA and it can be considered a good research opportunity for further investigation.

- **P1.** *Domain experts positively impact the definition of requirements for SCMA.*

  `Supporting statement:` “They have important information gained from their experience that helps the requirements to fit better the real needs of the domain” (#I5).

`Supporting statement:` "The definition of the requirements for SCMA is fully tied to its domain and so it is essential to have a domain expert to act as the bridge between domain and the application" (#I17).

The proposition P1 got total approval of the respondents and so it does not need to be refined.

- **P5.** *The satisfaction of non-functional requirements depends upon the design decisions made*

  `Supporting statement:` "There are always trade-offs for architectural decision making, which also applies to SCMA" (#I5).

  `Supporting statement:` "Any non-functional requirement is based on the cost-benefit ratio, that is, when implemented, another will be impacted, so it must be evaluated to decide whether or not to develop a specific non-functional requirement" (#I15).

  The satisfaction of non-functional requirements is indeed a joint work under the participation of requirements engineer, software architect, and development team. The proposition P5 got total approval of the respondents and so it does not need to be refined.

- **P6.** *The level of technical skills impact the design/development of SCMA.*

  `Supporting statement:` "Inexperience with the technologies that a SCMA requires produces a system more prone to bugs" (#I5).

  `Supporting statement:` "One of the main lacks in this context is the absence of a framework intended to support the development of SCMA" (#I12).

  The proposition P6 got total approval of the respondents and so it does not need to be refined.

- **P9.** *The use of architectural styles positively impacts the design of SCMA.*

  `Supporting statement:` "Architectural styles facilitate code maintenance and legibility" (#I7).

  The proposition P9 got total approval of the respondents and so it does not need to be refined.

- **P10.** *Effective mobile resource management positively impacts the architecture design of SCMA.*

  `Supporting statement:` "SCMA or any app that consumes a lot of resources, managing them is essential" (#I2).

  `NAND statement:` "Although it is desirable, the effective use of mobile resources does not have as much impact on the apps of non-continuous use" (#I16).

  Despite the above statement provided by the interviewee #I16, whose app had not a continuous use, the majority of SCMA requires a lot of mobile resources and this needs to be well managed; otherwise, it will impact negatively the product success.

- **P17.** *Testing SCMA involves the need to simulate real-world scenarios.*

  `Supporting statement:` "Testing for SCMA is the main bottleneck for SCMA, and simulation of real-world scenarios is one of the major challenges of these tests. We can not predict, for example, how people will behave in a life-threatening scenario" (#I2, #I5).

  `Supporting statement:` "These applications will be used in the complex context that brings the possibility of various types of errors, it is important that the application is robust and prepared for real scenarios" (#I17).

  The proposition P17 got total approval of the respondents and so it does not need to be refined.

5.2.5 Perform interviews with new practitioners

From the analysis of two interviews (#I20, #21) performed in December 2018 with 2 software architects of a public institution of the government of Bahia, it was possible to confirm our conclusions described in the last section and to provide some recommendations for the SE team (Section 5.4). The main findings are described next.

The respondent #I20 reported that the critical NFRs for their project (app A10) were performance and security and both were part of the architecture decision and were taken into consideration, but some other NFRs, which they considered to have lower priority were ignored because of the same reasons discussed earlier in this chapter (P4). Nevertheless, the application A21 failed on prioritizing critical RNFs and noticed the consequences of neglecting them after launching the app. The respondent #I21 reported that one of the main concerns with innovative mobile apps is speeding up development under the pressure of time to market. "*The first one to launch an idea in the market wins!*".

The respondent #21 confirmed our findings about P7 in the previous section. Because they failed to implement architectural decisions, technical debt was generated in the project. "*We still have unresolved bugs in the system because we do not have time to come back to fix them*".

They did not adopt a framework for smart cities (P13) but they made use of several communication libraries to connect to other platforms, and APIs (P8). One successful example reported by #I20 was the use of the *Layer7 API Gateway*[3], a high-performance gateway to connect the data and applications across any combination of cloud, container or on-premises environments. It is a consolidated market solution applied to increase speed, quality, and security but also to decrease the complexity of the smart solution (P14). "*If there is a known solution, we do not create one, we reuse it*" (#I20).

Similarly, the development of the app A21 made use of smart technologies to better manage the mobile resource (P16), mainly the battery and the GPS. Since this app keeps looking for the closest available professional to attend a request, the use of the GPS is continuous. The solution used a Google API and changed the architecture to not sending

[3]Layer7 API Gateway is available at https://www.broadcom.com/products/software/api-management/layer7-api-gateways

the GPS signal all the time but sending it only when the user has exceeded a radius of X meters (this number may vary), and calculating the best route.

The native development approach was chosen to develop the app A10 because performance was critical and so they preferred to develop it native for both platforms (Android and iOS) even knowing that the financial cost to retain professional engineers to maintain both platforms was much higher, they do not regret that decision. According to #I20, the hybrid approach has a lot of limitations and has caused a lot of problems in the past related to compatibility and performance issues (P13). On the other hand, the hybrid development approach was chosen to develop the app A11 using the Ionic[4], a cross-platform mobile app development. Although it is a widely know framework, previous interviewees besides #I21 do not recommend it. The reason mentioned includes performance and the life cycle of the app. According to him, it will always be shorter than a native app because ionic libraries release updates faster than android libraries and the developer needs to keep updating the versions of the code, the code is unstable until it breaks. When SCMAs are developed for fields such as e-governance, the SE team has no autonomy, and the project is managed by people in different roles who have divergent interests most of the time. Besides that, these customers usually have different opinions about what should be included in the SCMA (P11). Aiming to manage the conflicting requirements raised in this context, the software architect #I20 raised the need for a demand prioritization committee in the project. His SE team applied the prioritization matrix approach to provide to project stakeholders with a resource to balance between requirements, the priorities, and impacts, and to increase the chances of a successful project by promoting consensus between stakeholders.

According to #I20, tests are neglected most of the time and this is a cultural problem where they shifted testing to the end. He confessed that failed when they did not give relevance to testing earlier in the planning phase and so got a negative impact later in the development process. Testing SCMA is even more crucial because of the need for testing real-world scenarios (P15-P17). For that, the respondent #I20 recommended the Firebase test lab for Android and iOS[5], which is a Google platform that provides cloud-based infrastructure for testing Android and iOS apps.

Finally, adopting agile methodology in software development was a good alternative found to handle technical debt in the project (P2). By ensuring the value of the following agile manifesto: *working software over comprehensive documentation*[6], both development teams learned, from failures, to document as little as possible and express the requirements as user stories and eventually prototypes.

5.3 FINAL PROPOSITIONS FOR THE GROUNDED THEORY

After evaluating the theory by performing the five tasks detailed in this chapter, we created a list of refined propositions and highlighted the changes in Table 5.2. A total of nine propositions were refined after applying the five evaluation tasks.

[4]https://ionicframework.com

[5]Firebase Test lab is available at https://firebase.google.com/docs/test-lab

[6]Agile manifesto is available at https://agilemanifesto.org

Table 5.2 List of propositions that were modified after the theory evaluation process

P2	The lack of `at least technical` documentation creates technical debt.
P4	`Some` non-functional requirements are not taken into consideration by the SE team when making design decisions.
P7	`Failing to implement` architectural decision creates technical debt.
P8	The architecture design of SCMA is `mostly` API-centric.
P11	The smart cities context `adds` complexity to a mobile app's `backend`.
P12	The development or adoption of smart city frameworks `may` reduce the complexity of the development of SCMA.
P13	The adopted development approach and `its associated tools bring specific` development challenges for SCMA.
P15	The lack of `known automated` testing tools for mobile applications negatively impacts the testing of SCMA.
P16	Training `internal` users positively impacts the acceptance of SCMA products.

5.4 RECOMMENDATIONS FOR THE SE TEAM

Based on the propositions concerning not only architects' actions but also those of requirements analysts, developers, and testers, we provide the following additional support for SE teams that are designing SCMAs:

1. *Adopt agile methodologies.* Since teams have a culture of member turnover, documentation must be extremely relevant to be of value. In the scenario of SCMA, where the pressure to deliver a product is critical but the customer's need is also very important, agile methodologies are a good choice because they produce a constant delivery stream of value to the customer and allow developers to reorganize priorities as needed. Priority matrices are strongly recommended to track the changing priorities among quality attributes (QA).

2. *Bring all stakeholders to the design decision meetings.* A non-functional requirement (NFR) is prioritized over another based on the cost-benefit ratio, and a decision about anyone attribute may impact other QAS and the associated NFRs. Therefore, design decisions need to be evaluated by the entire team including the domain experts as to have a successful result.

3. *Invest in training on new technologies.* If possible, after choosing new technologies and tools, align the team through a short training oriented to the use of this smart technology. It would decrease the time needed to learn new technologies and will allow you to use the existing smart technologies that are aimed for SCMAs.

4. *Adopt a SCMA framework when you are really going to use it.* This topic arose some disagreement because there were architects that adopted robust frameworks and so, took a lot of effort to learn it, but only used a few components from the framework. The use of SCMA frameworks, such as Fiware was strongly recommended. However,

only when the solution is fully integrated into the framework and if you use several components it provides rather than just take advantage of a few components.

5. *Separate the effort required to plan and perform testing.* Testing SCMA is an exhausting activity because the process is not always fully automated. Development teams have the "culture" of shifting testing to the last moment and it entails several problems to the application.

6. *Invest time in user training.* The respondents were very positive about the impact of user training on mobile applications developed for smart cities.

7. *Try Firebase Test Lab for testing the apps.* Our respondents strongly recommended the use of the test lab to provide a realistic environment for testing SCMA.

8. *Identify and evaluate those architectures widely used in SCMA.* Choosing the architecture that maximizes similarities in the face of operating system differences can greatly influence the effort required for cross-platform development and maintenance. Respondents mentioned abandoning cross-platform strategies due to cost considerations.

9. *Examine the architecture styles used to realize those SCMA architectures.* A recurring problem among the respondents was the lack of knowledge about architectural styles, which may have prevented them from having a more robust and intelligent design. Individual styles can be swapped in and out of a larger architecture to achieve specific values for NFRs.

10. *Native development approaches are more welcome by participants than hybrid approaches.* It was stated by respondents that a hybrid approach, such as Ionic, impacts the performance of the app. The life cycle of the hybrid-developed app is usually lower compared to a native-developed app due to the necessity to perform updates frequently. On the other side, approaches that natively render applications, such as React Native[7] was recommended by some of the participants.

5.5 THREATS TO VALIDITY

There are some threats to the validity of our research and some of those threats arise from the choices about how to evaluate the research results and process. In this section, we discuss how elements of our evaluation approach contribute to our research having three types of validity.

- Construct Validity

To have construct validity a research must be objective and unbiased and the evaluation method must be similarly objective and unbiased. The extensive use of evaluation criteria from the published literature and interactions with the actual engineers involved

[7]React Native is available at http://www.reactnative.com

in developing the apps being studied ensures that this work has construct validity due to eliminating researcher bias. Interviewing developers from two additional projects who were unaware of the results of the first interviews eliminated participant bias. Previewing the interview questions with an experienced developer was intended to ensure that the questions being asked were related to the study being investigated.

- Internal Validity

To have internal validity a research must ensure that the objective and unbiased procedures are strictly followed. Our evaluation followed techniques published in well respected venues for evaluating a grounded theory as to determine whether inferences, in this case in the form of propositions, are unbiased. The development of a grounded theory applied in this dissertation was not intended to alter the behavior of the developers interviewed and thus the inferences arising from this research are in the form of propositions in an initial model. The review of these propositions was carried out by experienced researchers other than the principal investigator.

- External Validity

To have external validity a research must be applicable to areas outside the scope of the study and the evaluation techniques must ensure this. The results of this work were focused on mobile apps providing emergency services. The grounded theory was developed by analyzing the architectural structures and development method with very little reference to the emergency service domain. This work may not be applicable to just any app but it should apply to apps used in a mobile context.

5.6 CHAPTER SUMMARY

This chapter described our theory evaluation process in order to validate whether the constructs and propositions of the theory are clear and precise; whether the theory can be (dis) confirmed; and whether the theory scope is clearly specified and the implications of widening the scope. The evaluation was composed of five tasks: (1) review the literature, (2) apply a set of theory evaluation criteria, (3) validate the model with researchers in similar areas, (4) validate the model with practitioners involved in theory generation, and (5) perform interviews with new practitioners. As a result, we provided a list of propositions that were modified after the theory evaluation process in Section 5.3 and 10 recommendations for the SE team in Section 5.4. In the next chapter, we present the main conclusions of this work, summarize the research contributions and discuss potential future work.

Chapter 6

CONCLUSION AND FUTURE WORK

Smart city applications have become an integral part of human activity in cities with the appropriate infrastructure and mobile devices have been key to this evolution. While mobile computing is becoming ubiquitous, the creation of architectural designs for mobile applications is still not a trivial process. Mobile applications have specific requirements and the development process needs to be adapted to address the increased use of frameworks and to accommodate the criteria for being accepted into the proprietary app store of the equipment manufacturer (IMAGINOVATION, 2017). Although there are new studies investigating the different facets of the world of mobile applications (ZHAO; MEDVIDOVIC, 2019), there is very little knowledge about how software engineering (SE) teams actually proceed to design their apps in practice.

6.1 SUMMARY OF RESEARCH CONTRIBUTIONS

This work made several contributions to the SE research community, but we focused on three: (1) empirical data from a multi-case study, (2) a set of reverse-engineered architectures, and (3) a theoretical development model of developing SCMA. Each contribution was described in detail earlier in this dissertation so each is summarized below.

6.1.1 Empirical Data from a Multi-case Study

This research was partially motivated by the knowledge gap surrounding the development of smart city mobile applications (SCMA). Using a multi-case study approach was intended to produce a larger knowledge base of empirical data more quickly than a single case study approach. The management of the data collection and analysis process was carefully documented. Researchers will be able to easily adapt the experimental procedures freeing them to concentrate on recruiting projects to participate in research studies.

6.1.2 Reverse-Engineered Architectures

This research focused on the architectures underlying the apps chosen for the study. Project documents and interviews with staff provided an as-designed view of the architecture. As a basis for comparison, reverse engineering tools, such as COVERT and ACME were used to produce an as-built view of the architecture. These views were produced for six SCMA and then used to describe relationships between certain architectural styles and the properties of the resulting apps. The analyses will be of use in future research projects.

6.1.3 Theoretical Development Model

This research organized and analyzed the empirical data using a grounded theory technique. This technique produced a model of developing SCMA. The model, which contains 21 constructs and 17 propositions, can be used to answer "what-if" questions about changes to the development process. This model provides the basis for an evolving understanding of SCMA development. As new studies produce additional data, future researchers will be able to incorporate their data into the model, revise and strengthen the model. As future studies investigate mobile apps in domains different from emergency management the model will become more robust.

6.2 RESEARCH PRODUCTS

This research generated several types of tangible products that support the significant contributions to the research community. These contributions have been discussed in detail in this dissertation and summarized in this chapter, but have also resulted in three research products.

Empirical data - The raw data, analyzed data and all experimental procedures have been captured on our research website.

Journal paper - A paper addressing some intermediate results of this work was published in the journal Empirical Software Engineering (EMSE):

- *Farias, Roselane Silva*, de Souza, Renata Maria, McGregor, John D., and de Almeida, Eduardo Santana. Designing smart city mobile applications: An initial grounded theory. *Empirical Software Engineering*, May 2019. ISSN 1573-7616.

Replication package To promote openness within the research community and facilitate further work, an extensive replication package was created and made publicly available. The complementary material site is available on github[1]. The package includes copies of survey forms, questions and other materials created for use in the original research.

Impact on studied projects - An indirect and very difficult to quantify, contribution is the knowledge gained by the study participants. Through the interviews and follow-up questions, the participants have had the opportunity for introspection about

[1]Supplementary material site is available at https://rose2s.github.io/EMSE2019

their work. It is likely that the intervention of the researchers has influenced the future behavior of the study participants.

6.3 RELATED WORK

This study is within the intersection of mobile software, software architecture, and smart city issues. To the best of our knowledge, there is no other study that addresses all three areas so we have divided the discussion into the following categories of research relevant to our work: empirical studies on mobile applications, software architectures, and smart city architectures. Next, we provide an overview of the most relevant previous work from each of these areas and outline the differences between them and this work.

6.3.1 Mobile applications

Mobile applications are an increasingly important part of our daily life, and the research community has been invested in this area focusing mainly on issues in mobile app development (WASSERMAN, 2010; JOORABCHI; MESBAH; KRUCHTEN, 2013; KATZ, 2014; NANDAN; PURSCHE; ZHE, 2014; NAGAPPAN; SHIHAB, 2016; FRANCESE et al., 2017). Many challenges found in traditional software development will also impact mobile development (NAGAPPAN; SHIHAB, 2016). These studies of mobile applications have identified common issues faced by both areas, and those challenges specific to mobile devices. As apps have become more complex and life critical their development started to require consideration of factors beyond ordinary telephony features, those features related to making or receiving a telephone call. Our work interviewing SE teams confirmed most of the challenges described in these papers, but also revealed challenges specific to the smart city domain.

To exemplify, Wasserman (2010) described a set of issues related to software development for mobile devices. Among the topics addressed in his paper are potential interaction with other applications, sensor handling, heterogeneity, security, user interface, restricted resources, and power consumption. In his study, he found that as mobile applications have become more complex, it is essential to apply software engineering processes to assure high-quality mobile applications. He concluded that mobile developers are likely to use recommended sets of principles, but rarely use any formal development processes. Similarly, our work revealed a set of non-functional requirements that smart city mobile applications (SCMA) have taken into account, and those, which have been neglected.

Katz (2014) identified eight *security issues* that affect mobile applications' architecture because of their constraints. The three issues pertinent to our work include:

- Not using encryption or using weak encryption - The workflow in the app will be different depending upon the level of encryption being used. This will affect the architecture as well as the performance of the software.

- Not implementing secure communications to servers - There are many patterns of secure communication. The pattern selected will impact the shape of the architecture.

- Patching your app too slowly - Mobile apps are exposed to a wide range of sites that may infect the app. The architecture must be designed to allow for fast and easy modifiability.

These issues are pertinent to our work in that they impact the architecture of the mobile apps, each of which constitutes the client side of client-server systems where the client is mobile. Beyond the basic client server architecture, the complexity of these apps and the criticality of their constraints illustrate that the architectural styles used in these apps are worthy of study. The discussion in Katz (2014) addresses architecture issues but only within the confines of the security issues. Our work considers architecture styles that guide architecture creation for complete systems. These styles will be investigated using several quality attributes including security. More recently, FRANCESE et al.'s work (2017) reported conclusions about mobile development, based on a qualitative investigation, which were also observed in our study:

- app development is mostly done by junior developers. In fact, our study interviewed 10 junior developers (less than 5 years of industrial experience) as compared to 2 senior developers;
- agile methodologies and cross-platform development frameworks are adopted by many teams. We could observe that even though they follow no development process, they make use of agile methodologies, such as daily meetings, and adopt frameworks for architecture specification;
- support for testing is considered inadequate. We have identified several pieces of evidence of inadequate support, which is an identified gap and serves as a potential trend upon which to base research;
- fragmentation of software and hardware is a concern. Indeed developers have complained about the need to have different skills, and
- app development is considered different from the development of web/desktop applications. We confirmed this finding by describing the design process of SCMA, which is challenging and very different from desktop system development.

6.3.2 Software architecture

Software architecture (SA) plays a crucial role in the software development process. Papers on this area have addressed a variety of topics such as non-functional requirements (NFR), decision-making, architectural styles, and others. Our work addressed all these aspects, which are essential for the design process of SCMA.

Non-functional requirements (NFR) directly affect the software architecture, especially architectural decision-making. Regarding this subject, Ameller et al. (2012) presented an empirical study about how software architects deal with NFRs in practice. Their main conclusions include that the two most important types of NFRs for architects relate to the QAs *performance* and *usability*, and that NFRs are not often documented.

These results were also observed by our empirical study on mobile applications. However, limiting our scope to SCMA and interviewing different stakeholders we could note that the role of the stakeholder and the project's domain may influence the perception of importance of the NFRs. This could be observed in our study but not in theirs because all our participants played the architect role and there was essentially a single domain so that our analysis was very focused.

Regarding software architectural styles, Sharma, Kumar and Agarwal (2015) addressed the importance of choosing an architectural style appropriately using a survey. They specify the advantages and disadvantages of each architectural style and clarify that it would be helpful for developers to select an appropriate style according to their project's requirements. In that regard, our work investigated how SE teams build the architecture of SCMA, and listed several requirements inherent to the domain of smart cities and mobile applications that have impacted the design of these systems and choice of one style over another.

Software architecture and mobility To support the rapid and cost-effective design of smart applications on mobile devices, Medvidovic and Edwards (2010) provided a roadmap of the intersection of the areas of software architecture and mobility. They highlighted representative existing solutions and identified several remaining research challenges. They concluded that existing architectural principles need to be adapted and novel architectural paradigms devised to support mobile computing. In a way, our work addresses this opportunity since our theory describes how architects/developers have dealt with architecture design in practice.

Liu et al. (2011) conducted an empirical study from a developer's perspective on two hundred apps from the App Store to provide a focused overview of the status and trends of iOS mobile health apps and an analysis of related technology, architecture, and user interface design issues. As a result, one of the implications for developers is the compatibility with external sensors. This finding remains true since our study also observed that developers have different problems depending on the platform chosen (Android and iOS) and cross-platform development become a known issue for mobile software development (NAGAPPAN; SHIHAB, 2016).

Bagheri et al. (2016) described the main software architectural principles in contemporary mobile software by mining the reverse-engineered architectures of hundreds of Android apps. They listed five mobile computing drivers for contemporary mobile software and traced them back to the software architecture literature. Although this work is not related to smart cities, it was important for us to understand how software architecture has been applied to mobile devices. However, our work looks beyond the communication portion of the application to the domain of application. We investigated those architectural styles in actual use in the emerging domain of apps for mobile devices.

6.3.3 Smart city architecture

Regarding the smart city domain, this research was based on two work: a literature review (TOMAS et al., 2013) and a survey on smart city architectures (SILVA et al., 2013). These studies have concluded that no architecture fully satisfies all essential requirements in the context of smart cities. However, they reported important aspects of this context.

Tomas et al. (2013) conducted a literature review on smart city architectures. Their work found eleven relevant architectural approaches, and then identified a set of issues that these approaches aim to solve and some architectural patterns employed. Their main findings were regarding the main challenges and issues of smart city architectures as well as the most used architectural patterns in this context. Although their results were very complete and verified for several smart city software, they were not concerned about the architectural style, which was covered by our work.

Later, Silva et al. (2013) conducted a survey discussing the same topic. They highlighted a set of essential NFRs for smart city applications, and we used their list of architecture requirements as a guide to conduct our interviews. For each NFR, we asked the SE teams how it was specified, designed, implemented and tested.

Santana et al. (2017) provided an updated review of the literature and a reference architecture for a software platform for mobile apps. The architecture for this platform is not specifically about mobile applications but includes the means to integrate mobile applications into a network of stationary devices. Our work built on this information to go more deeply into the mobile aspects of these systems by examining the concrete application architectures used rather than the more abstract reference architecture reported in their work.

6.4 FUTURE WORK

Due to the relatively brief time allocated for a master's degree, this work should be viewed as an initial step in assembling the evidence that supports the SCMA design process. The research reported here is the foundation for additional investigation into the design of SCMA. There are several possible avenues for future research. The opportunities for future research include:

- **Study Replications -** Aiming to enhance the possibility of generalization, more case studies should be conducted in similar conditions. The next step is to perform studies in an industry environment with larger and more diverse mobile applications. This will give us the possibility of more variation in the data to analyze and the ability to make a cross-case analysis by performing comparisons among applications, and thus supporting a more robust theoretical model.

- **Deeper Architectural Analysis -** We intend to investigate in more depth whether a specific architectural style impacts the architecture construction, comparing it with the set of smart city non-functional requirements, by reverse-engineering the software architecture of a large number of mobile applications and subjecting those results to rigorous analysis.

- **More refined theory evaluation -** We intend to expand our theory evaluation not only by refining the existing propositions, as we did in this work but also by adding new propositions to enhance the theoretical model.

- **A reference architecture for SCMA -** We intend to expand this work by providing a reference architecture specific for SCMA including architectural styles most used to structure SCMA and architecture tactics used to design SCMA.

6.5 CONCLUDING REMARKS

We conducted a multi-case study with four SE teams who developed a total of nine mobile applications for smart cities. Using a variety of techniques we collected empirical data from the artifacts and personnel of these projects. The data were analyzed using grounded theory techniques and the resulting information was used to derive a theory, grounded in the evidence we collected, describing the characteristics in the software development process and factors that influence how SE teams create the architectural design of SCMA. In addition to the collection of first-hand data from developers, an initial version of the theory and derived model was used to collect input from researchers in similar areas. This additional data was used to modify and extend the theory.

This contribution to the body of knowledge concerning designing SCMA is an initial step in developing an architecture design technique for SCMA. To exemplify, our theoretical model predicts that developers will rely on a set of smart city frameworks to reduce the complexity of designing mobile applications in this domain and confirms previous studies about the impact of development approaches in mobile apps. The model does not yet detail what those frameworks should be nor how they should interact to avoid increasing the complexity of the applications. The empirical grounded theory approach used in this work should provide an excellent basis for attacking the next level of issues constraining the evolution of the model.

BIBLIOGRAPHY

ADOLPH, S.; HALL, W.; KRUCHTEN, P. Using grounded theory to study the experience of software development. *Empirical Software Engineering*, v. 16, n. 4, p. 487–513, Aug 2011. ISSN 1573-7616.

AIRAKSINEN, M. et al. Smart city performance measurement framework citykeys. In: *2017 International Conference on Engineering, Technology and Innovation (ICE/ITMC)*, 2017. p. 718–723.

AMELLER, D. et al. How do software architects consider non-functional requirements: An exploratory study. In: *2012 20th IEEE International Requirements Engineering Conference (RE)*, 2012. p. 41–50. ISSN 1090-705X.

AMORIM, A. M. et al. Quality attributes analysis in a crowdsourcing-based emergency management system. In: *ICEIS - Proceedings of the 19th International Conference on Enterprise Information Systems, Volume 2, Porto, Portugal*, 2017. p. 501–509.

BAGHERI, H. et al. Software architectural principles in contemporary mobile software: from conception to practice. *Journal of Systems and Software*, Elsevier, v. 119, p. 31–44, 2016.

BIØRN-HANSEN, A. et al. An empirical study of cross-platform mobile development in industry. *Wireless Communications and Mobile Computing*, 2019.

BOSCH, J. *Design and Use of Software Architectures: Adopting and Evolving a Product-line Approach*. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 2000. ISBN 0-201-67494-7.

BRERETON, P. et al. Using a protocol template for case study planning. In: *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering*. Swindon, UK: BCS Learning & Development Ltd., 2008. (EASE'08), p. 41–48.

CHARMAZ, K. *Constructing Grounded Theory: A Practical Guide Through Qualitative Analysis*: SAGE Publications, 2006. ISBN 9780761973539.

COCCHIA, A. *Smart and Digital City: A Systematic Literature Review*: Springer International Publishing, 2014. 13-43 p. ISBN 978-3-319-06160-3.

CORBIN, J. M.; STRAUSS, A. Grounded theory research: Procedures, canons, and evaluative criteria. *Qualitative sociology*, Springer, v. 13, n. 1, p. 3–21, 1990.

CORBIN, J. M.; STRAUSS, A. *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory.*: SAGE Publications., 2008.

DAMERI, R. Searching for smart city definition: a comprehensive proposal. *INTERNATIONAL JOURNAL OF COMPUTERS & TECHNOLOGY*, v. 11, n. 5, p. 2544–2551, Oct. 2013.

DINIZ, H. B. M. et al. A reference architecture for mobile crowdsensing platforms. In: *Proceedings of the 18th International Conference on Enterprise Information Systems*: SCITEPRESS - Science and Technology Publications, Lda, 2016. (ICEIS 2016), p. 600–607. ISBN 978-989-758-187-8.

ESPOSTE, A. de M. D. et al. Design and evaluation of a scalable smart city software platform with large-scale simulations. *Future Generation Computer Systems*, v. 93, p. 427 – 441, 2019. ISSN 0167-739X.

FERREIRA, J. E. et al. Smart services: A case study on smarter public safety by a mobile app for university of são paulo. In: *2017 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computed, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation*, 2017. p. 1–5.

FIWARE Foundation. *Fiware - Developers*. 2018. Available from Internet: ⟨https://www.fiware.org/developers/⟩.

FRANCESE, R. et al. Mobile app development and management: Results from a qualitative investigation. In: *Proceedings of the 4th International Conference on Mobile Software Engineering and Systems*. Piscataway, NJ, USA: IEEE Press, 2017. (MOBILESoft '17), p. 133–143. ISBN 978-1-5386-2669-6.

FRAUNHOFER. *Fraunhofer Project Center for Software Systems and Engineering*. 2014. Available from Internet: ⟨http://www.fpc.ufba.br/index.php/en/about-us⟩.

GARLAN, D. Software architecture: A roadmap. In: *Proceedings of the Conference on The Future of Software Engineering*. New York, NY, USA: ACM, 2000. (ICSE '00), p. 91–101. ISBN 1-58113-253-0.

GLASER, B. *Doing Grounded Theory: Issues and Discussions*. Sociology Press, 1998. ISBN 9781884156113. Available from Internet: ⟨https://books.google.com.br/books?id=XStmQgAACAAJ⟩.

GLASER, B.; STRAUSS, A. *The Discovery of Grounded Theory: Strategies for Qualitative Research*: Aldine, 1967. (Observations (Chicago, Ill.)). ISBN 9780202302607.

GOOGLE. *Application Fundamentals - Android Developer Documentation*. 2018. Available from Internet: ⟨https://developer.android.com/guide/components/fundamentals⟩.

GOOGLE. *AsyncTask - Android Developer Documentation*. 2018. Available from Internet: ⟨https://developer.android.com/reference/android/os/AsyncTask⟩.

HODA, R.; NOBLE, J.; MARSHALL, S. Developing a grounded theory to explain the practices of self-organizing agile teams. *Empirical Software Engineering*, v. 17, n. 6, p. 609–639, Dec 2012. ISSN 1573-7616.

HODA, R.; NOBLE, J.; MARSHALL, S. Self-organizing roles on agile software development teams. *IEEE Transactions on Software Engineering*, v. 39, n. 3, p. 422–444, March 2013. ISSN 0098-5589.

IMAGINOVATION. *16 Reasons the App Store Rejects Mobile Apps & How to Avoid Them*. 2017. Available from Internet: ⟨https://medium.com/@Imaginovation/16-reasons-the-app-store-rejects-mobile-apps-how-to-avoid-them-63f73fa33a3a⟩.

IVAN, I. et al. The modern development cycle of citizen oriented applications. *Studies in Informatics and Control*, v. 18, p. 263–270, 09 2009.

JANSEN, A. G. J. *Architectural design decisions*. Thesis (Doctoral) — University of Groningen, 2008.

JANSSEN, D.; JANSSEN, C. *Techopedia - The IT Education Site*. n.d. Available from Internet: ⟨https://www.techopedia.com/definition/27913/technical-debt⟩.

JOORABCHI, M. E.; MESBAH, A.; KRUCHTEN, P. Real challenges in mobile app development. In: *2013 ACM / IEEE International Symposium on Empirical Software Engineering and Measurement*, 2013. p. 15–24. ISSN 1949-3770.

KATZ, K. *Eight security issues to prepare for in mobile app development*. 2014. Available from Internet: ⟨https://www.kony.com/resources/blog/eight-security-issues-prepare-mobile-app-development⟩.

KHATOUN, R.; ZEADALLY, S. Smart cities: Concepts, architectures, research opportunities. *Commun. ACM*, ACM, New York, NY, USA, v. 59, n. 8, p. 46–57, jul. 2016. ISSN 0001-0782.

KITCHIN, R. Making sense of smart cities: addressing present shortcomings. *Cambridge Journal of Regions, Economy and Society*, v. 8, n. 1, p. 131–136, 2015.

LIU, C. et al. Status and trends of mobile-health applications for ios devices: A developer's perspective. *Journal of Systems and Software*, v. 84, n. 11, p. 2022 – 2033, 2011. ISSN 0164-1212. Mobile Applications: Status and Trends. Available from Internet: ⟨http://www.sciencedirect.com/science/article/pii/S0164121211001610⟩.

MEDVIDOVIC, N.; EDWARDS, G. Software architecture and mobility: A roadmap. *Journal of Systems and Software*, v. 83, n. 6, p. 885 – 898, 2010. ISSN 0164-1212. Software Architecture and Mobility. Available from Internet: ⟨http://www.sciencedirect.com/science/article/pii/S0164121209002854⟩.

MEHDI, M. et al. Referenceable mobile crowdsensing architecture: A healthcare use case. *Procedia Computer Science*, v. 134, p. 445–451, 2018. ISSN 1877-0509. The 15th International Conference on Mobile Systems and Pervasive Computing (MobiSPC 2018) / The 13th International Conference on Future Networks and Communications (FNC-2018) / Affiliated Workshops.

MOJICA, I. J. et al. A large-scale empirical study on software reuse in mobile apps. *IEEE Software*, v. 31, n. 2, p. 78–86, Mar 2014. ISSN 0740-7459.

MORSE, J. M. Procedures and practice of mixed method design: Maintaining control, rigor, and complexity. *The Sage Handbook of Mixed Methods Research in Social & Behavioral Research*, p. 189–208, 2003.

NAGAPPAN, M.; SHIHAB, E. Future trends in software engineering research for mobile apps. In: *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, 2016. v. 5, p. 21–32.

NANDAN, N.; PURSCHE, A.; ZHE, X. Challenges in crowdsourcing real-time information for public transportation. In: *2014 IEEE 15th International Conference on Mobile Data Management*, 2014. v. 2, p. 67–72. ISSN 1551-6245.

NATIONS, D. o. E. U.; AFFAIRS, S. *World Population Prospects 2019: Highlights*, 2019. 46 p. Available from Internet: ⟨https://esa.un.org⟩.

NEWCOMBE, T. *The Rise of the Sensor-Based Smart City.* 2014. Available from Internet: ⟨http://www.govtech.com/data/The-Rise-of-the-Sensor-Based-City.html⟩.

OPENSIGNAL. *Global State of Mobile Networks.* 2016. Available from Internet: ⟨https://www.opensignal.com/reports/2016/08/global-state-of-the-mobile-network⟩.

ROMAN, G.-C.; PICCO, G. P.; MURPHY, A. L. Software engineering for mobility: A roadmap. In: *Proceedings of the Conference on The Future of Software Engineering.* New York, NY, USA: ACM, 2000. (ICSE '00), p. 241–258. ISBN 1-58113-253-0.

RUNESON, P.; HÖST, M. Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, v. 14, n. 2, Dec 2008. ISSN 1573-7616.

SAIFI, R. *The 2017 Mobile App Market: Statistics, Trends, and Analysis.* 2017. Available from Internet: ⟨https://www.business2community.com/mobile-apps/2017-mobile-app-market-statistics-trends-analysis-01750346\\#7Fi1f2awFZ0xIFTh⟩.

SANTANA, E. F. Z. et al. Software platforms for smart cities: Concepts, requirements, challenges, and a unified reference architecture. *ACM Comput. Surv.*, ACM, New York, NY, USA, v. 50, n. 6, p. 78:1–78:37, nov. 2017. ISSN 0360-0300.

SEAMAN, C. B. Qualitative methods in empirical studies of software engineering. *IEEE Transactions on Software Engineering*, v. 25, n. 4, p. 557–572, Jul 1999. ISSN 0098-5589.

SHARMA, A.; KUMAR, M.; AGARWAL, S. A complete survey on software architectural styles and patterns. *Procedia Computer Science*, v. 70, p. 16 – 28, 2015. ISSN 1877-0509. Proceedings of the 4th International Conference on Eco-friendly Computing and Communication Systems.

SILVA, W. M. da et al. Smart cities software architectures: a survey. In: ACM. *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, 2013. p. 1722–1727.

SJØBERG, D. I. K. et al. Building theories in software engineering. In: SHULL, F.; SINGER, J.; SJØBERG, D. I. K. (Ed.). *Guide to Advanced Empirical Software Engineering.* London: Springer London, 2008. cap. 12, p. 312–336. ISBN 978-1-84800-044-5.

SOUSA, L. et al. Identifying design problems in the source code: A grounded theory. In: *Proceedings of the 40th International Conference on Software Engineering.* New York, NY, USA: ACM, 2018. (ICSE '18), p. 921–931. ISBN 978-1-4503-5638-1.

STATISTA. *Number of available applications in the Google Play Store from December 2009 to June 2018.* Android, Google, 2018. Available from Internet: ⟨https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/⟩.

STATISTA. *Number of mobile app downloads worldwide in 2017, 2018 and 2022 (in billions).* 2018. Available from Internet: ⟨https://www.statista.com/statistics/271644/worldwide-free-and-paid-mobile-app-store-downloads/⟩.

STOL, K.-J.; RALPH, P.; FITZGERALD, B. Grounded theory in software engineering research: A critical review and guidelines. In: *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)*, 2016. p. 120–131.

SYER, M. D. et al. Exploring the development of micro-apps: A case study on the blackberry and android platforms. In: *2011 IEEE 11th International Working Conference on Source Code Analysis and Manipulation*, 2011. p. 55–64.

TAYLOR, K.; SILVER, L. Smartphone ownership is growing rapidly around the world, but not always equally. Pew Research Center, 2019.

TAYLOR, R. N.; MEDVIDOVIC, N.; DASHOFY, E. M. *Software architecture: foundations, theory, and practice*: Wiley, 1 edition, 2009.

TOMAS, G. H. et al. Smart cities architectures - a systematic review. In: HAMMOUDI, S. et al. (Ed.). *ICEIS (2)*: SciTePress, 2013. p. 410–417.

WASSERMAN, A. I. Software engineering issues for mobile application development. In: ACM. *Proceedings of the FSE/SDP workshop on Future of software engineering research*, 2010. p. 397–400.

WENGE, R. et al. Smart city architecture: A technology guide for implementation and design challenges. *China Communications*, IEEE, v. 11, n. 3, p. 56–69, 2014.

YIN, R. K. *Case study research: Design and methods*: Sage publications, 2013.

ZHANG, W. et al. An empirical study on big video data processing: Architectural styles, issues, and challenges. In: *2016 International Conference on Identification, Information and Knowledge in the Internet of Things (IIKI)*, 2016. p. 110–115.

ZHAO, Y.; MEDVIDOVIC, N. A microservice architecture for online mobile app optimization. In: *Proceedings of the 6th International Conference on Mobile Software Engineering and Systems*. Piscataway, NJ, USA: IEEE Press, 2019. (MOBILESoft '19), p. 45–49.

Appendix

A

CASE STUDY ARTIFACTS

In the stage phase 1 of this study, we conducted a multi case study using background questionnaires, archival records of the applications and 19 interviews with stakeholders of SE teams. This appendix presents the following artifacts: background questionnaire; interview planning; confidentiality agreement and interview consent term; and interview protocol.

A.1 BACKGROUND QUESTIONNAIRE

Background Questionnaire

This form aims to gather information about the participants. No information will be disclosed.

* Required

1. **Email address** *

   ____

2. **Full name** *

   ____

3. **Skype:** *

   ____

4. **Educational Background** *
   Check all that apply.

   ☐ Bachelor’s degree
   ☐ Master’s degree
   ☐ Doctorate degree
   ☐ Professional degree
   ☐ Other: ______________________

5. **In which role(s) have you worked before starting on this project?** *
   Check all that apply.

   ☐ Requirements Analyst
   ☐ Project Manager
   ☐ Software Engineer/Developer
   ☐ Tester
   ☐ Software Architect
   ☐ Other: ______________________

6. **How much experience do you have in the role(s) selected above?** *

Check all that apply.

- ☐ < 1 year
- ☐ between 1 and 3 years
- ☐ between 3 and 5 years
- ☐ > 5 years

7. **With which applications are you involved?**

____

8. **What is your role in this application?** *

Check all that apply.

- ☐ Software Engineer/Developer
- ☐ Tester
- ☐ Project Manager
- ☐ Requirements Analyst
- ☐ Software Architect
- ☐ Other: ______________________________

9. **How long have you been working on this project?** *

Mark only one oval.

- ◯ < 1 year
- ◯ < 2 years
- ◯ < 3 years
- ◯ More than 3 years
- ◯ Other: ______________________________

10. **Have you ever worked with mobile applications?** *

Mark only one oval.

- ◯ Yes
- ◯ No

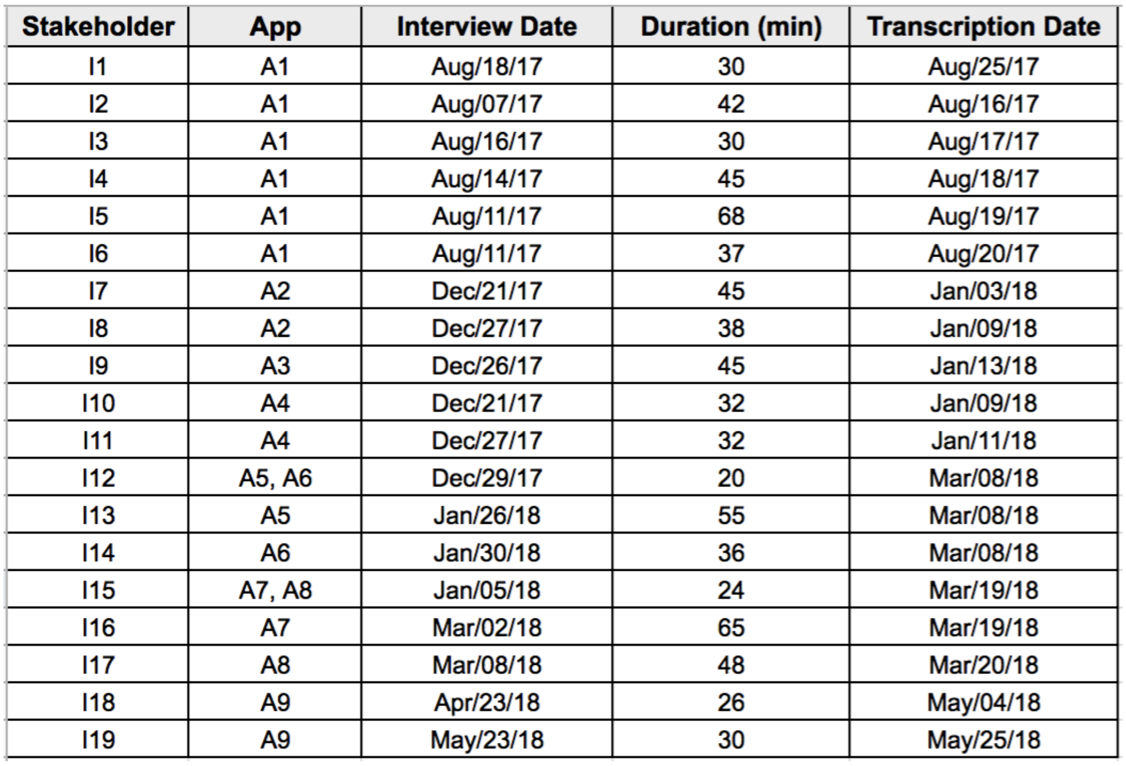
11. **Have you ever worked with applications in the smart city domain?** *

Mark only one oval.

- ⬭ Yes
- ⬭ No

12. **Is there any additional information that you would like to share with us?**

A.2 INTERVIEW PLANNING



Stakeholder	App	Interview Date	Duration (min)	Transcription Date
I1	A1	Aug/18/17	30	Aug/25/17
I2	A1	Aug/07/17	42	Aug/16/17
I3	A1	Aug/16/17	30	Aug/17/17
I4	A1	Aug/14/17	45	Aug/18/17
I5	A1	Aug/11/17	68	Aug/19/17
I6	A1	Aug/11/17	37	Aug/20/17
I7	A2	Dec/21/17	45	Jan/03/18
I8	A2	Dec/27/17	38	Jan/09/18
I9	A3	Dec/26/17	45	Jan/13/18
I10	A4	Dec/21/17	32	Jan/09/18
I11	A4	Dec/27/17	32	Jan/11/18
I12	A5, A6	Dec/29/17	20	Mar/08/18
I13	A5	Jan/26/18	55	Mar/08/18
I14	A6	Jan/30/18	36	Mar/08/18
I15	A7, A8	Jan/05/18	24	Mar/19/18
I16	A7	Mar/02/18	65	Mar/19/18
I17	A8	Mar/08/18	48	Mar/20/18
I18	A9	Apr/23/18	26	May/04/18
I19	A9	May/23/18	30	May/25/18

Figure A.1 Interview Planning.

A.3 CONFIDENTIALITY AGREEMENT AND INTERVIEW CONSENT TERM

Since all the interviews were performed in Portuguese, the confidentiality agreement and the consent term were written in Portuguese. They basically describes the purpose of the interview and it is a guarantee for the companies the researchers will keep confidential any information about the project and every archival data their have access during the study.

Termo de Confidencialidade

Estudo de Caso sobre o design de aplicações móveis no contexto de cidade inteligentes

Este termo se refere ao estudo de caso sobre o design de aplicações móveis no contexto de cidade inteligentes conduzido no projeto ______________________ conduzido através do programa de pós graduação em Ciência da Computação (PGCOMP) da Universidade Federal da Bahia (UFBA) sob orientação do Prof. PhD. Eduardo Santana de Almeida.

Declaramos estar ciente de que os documentos obtidos por meio deste estudo serão mantidos sob confidencialidade, **não divulgando nenhuma informação sensível do projeto.** Da mesma forma, nos comprometemos a manter sigilo das técnicas e documentos apresentados e que fazem parte do estudo. Em caso de eventuais publicações, divulgaremos apenas informações com alto nível de abstração, e não serão divulgadas diagramas ou imagens inseridas nos documentos do projeto.

Pesquisadora Responsável:

____

Roselane Santana Silva

PROFESSOR RESPONSÁVEL
Prof. PhD. Eduardo Santana de Almeida
Programa de Pós-Graduação em Ciência da Computação/UFBA

Salvador, Ba, Brasil

Termo de Consentimento

Estudo de Caso sobre o design de aplicações móveis no contexto de cidade inteligentes

OBJETIVO DO ESTUDO
Este estudo visa realizar uma investigação sobre os princípios arquiteturais em aplicações móveis no contexto de cidades inteligentes.

DECLARAÇÃO
Eu declaro concordar em participar do estudo de caso conduzido por Roselane Santana Silva do Programa de Pós-Graduação em Ciência da Computação (PGCOMP) da Universidade Federal da Bahia (UFBA), sob orientação do Professor PhD. Eduardo Santana de Almeida.

PROCEDIMENTO
Para participar deste estudo colaborarei em: (1) fornecer informações sobre minha experiência no projeto; (2) responder as questões feitas na entrevista pelas pesquisadoras com base nas práticas atuais; (3) permitir a verificação dos artefatos de software gerados pela equipe de desenvolvimento de software; e (4) responder um questionário final com as minhas impressões.

CONFIDENCIALIDADE
Eu estou ciente de que meu nome não será divulgado em hipótese alguma e não será utilizado em nenhum momento durante a apresentação dos resultados. Também estou ciente de que os dados obtidos por meio deste estudo serão mantidos sob confidencialidade. Da mesma forma, me comprometo a não comunicar os meus resultados enquanto o estudo não for concluído, bem como manter sigilo das técnicas e documentos apresentados e que fazem parte do estudo.

BENEFÍCIOS E LIBERDADE DE DESISTÊNCIA
Eu entendo que, os benefícios que receberei deste estudo são limitados ao aprendizado do material que é distribuído e ensinado. Também entendo que sou livre para realizar perguntas a qualquer momento, solicitar que qualquer informação relacionada a minha pessoa não seja incluída no estudo ou comunicar minha desistência de participação, sem qualquer penalidade. Por fim, declaro que participo de livre e espontânea vontade com o único intuito de contribuir para o avanço e desenvolvimento da Engenharia de Software.

Assinatura do Entrevistado

Salvador, Bahia, Brasil

A.4 INTERVIEW PROTOCOL

First of all, let me thank you for being willing to participate in this interview. In this study, we are trying to understand how the modeling/design process of mobile applications works in the context of smart cities. We are interviewing a range of software engineering teams, in order to uncover how this process works, what tools they use, what design decisions they make, what architectural styles and frameworks they adopt, what challenges exist in that domain, and so on.

The interview will last between 30 to 50 minutes and it is divided into three parts: quality attributes/non-functional requirements, aspects of mobile applications, and software architecture. By and large, we want to have a nice conversation with you about these topics. Do you have any question before we start?

Card 1: Quality Attributes/Non-Functional Requirements

1. What was your main role in the application?
2. Could you talk about the software development team members of the project?
3. How long did this project last?
4. Where did the idea of the app come from? Did you think about the smart city context initially?
5. How were the requirements defined? Was there any documentation?
6. Did you analyze the main challenges of the project from a development perspective? If so, how did you mitigate them?
7. Could you cite any problem encountered and how did you solve it?
8. *Show a list of quality attributes (performance, availability, security), and ask the following questions for each one:*
   (a) Why <`performance`> was/was not crucial?;
   (b) How did you deal with <`performance`> issues?;
   (c) How was it specified? How was it modeled in the architecture? How was it implemented? What aspects of testing were applied to ensure <`performance`>?

Card 2: Software Architecture

1. How was the system architecture designed? Was it based on other architecture/frameworks?
2. Have you participated in any design decision in Architecture? If so, do you remember any design decision your team made? why?

3. How were the quality attributes mapped in the architecture? How were they modeled? [Give some examples]

4. Has any architectural pattern been adopted? [*Give some examples*]

5. Has any architectural style been adopted? [*Give some examples*]

Card 3: Mobile Development

1. What were the main challenges faced in the development of the mobile application?

2. What development approach (native, hybrid) was used to develop the application? Why?

3. Have you used any framework during design/development phase? Which ones?

4. What smartphone features (GPS, camera, audio) does the application use and how? How did it impact the architecture?

5. Have you thought about maximizing battery life and managing these resources? How?

6. Has any test method for mobile applications been used?

Any other final thoughts? Thank you for your time.

Appendix B

REVERSE-ENGINEERING ARTIFACTS

In the stage 2 of this study, we reverse-engineered the software architecture of six apps that were either available in Google Play Store or we had access to its APK file. This appendix presents the following: a tutorial of the reversed-engineered architecture, feedback questionnaire, and images of the recovered architectures.

B.1 TUTORIAL OF THE REVERSED-ENGINEERED ARCHITECTURE

We provide a tutorial of our Reverse-Engineering study so that other researchers can replicate this study.

1. APK Downloader
   (a) Get the example.apk file using the app name or Google Play URL at APK-Downloader website
2. COVERT
   (a) Use COVERT tool to generate the intermediate code (*example.xml*):
      i. Download COVERT tool[1]
      ii. Add your *example.apk* into Covert/<`appFolder`>
      iii. Run the command:
         ```
         ./covert.sh <appFolder>
         ```
      iv. Output: *example.xml* (located in Covert/appFolder/analysis/model/example.xml)
3. ACME-Generator
   (a) Add your *example.xml* from step 1 into ACME-Generator folder[2]

[1]Available at https://www.ics.uci.edu/∼seal/projects/covert/index.html
[2]Available at https://github.com/arsadeghi/ACME-Generator

(b) Add your *example.apk* into ACME-Generator folder

(c) Run the command:

```
sh run.py ./apps/example.apk ./resources
```

(d) Output: *example.acme*

4. ACME STUDIO

   (a) Download ACME STUDIO[3]

   (b) Open your *example.acme* in ACME STUDIO

      i. Create a new project

      ii. Add your file *example.acme* into ACME workspace

      iii. Refresh and open the project in acme to see the architecture of your app.

For more details please check Covert Tool User Manual[4], which was developed by the Software Engineering and Analysis Lab (SEAL) of University of California, Irvine.

B.2 FEEDBACK QUESTIONNAIRE

Some developers allowed us to interview them again to get their feedback was used to validate the recovered architectures from the architecture reverse engineering of their apps, but most of them were preferred the questionnaire online, which we describe next.

- Architecture Reverse Engineering
  - Do you recognize these components?
  - Which of these components have you implemented?
  - Which of these components belong to the Android framework?
  - Is there any component that was crucial in the implementation, but this tool did not recover?
  - Is there any component that was not recovered?
  - Is it possible to identify any architectural style that was used (client server, publish subscribe) or any technology (Fiware, Firebase, Context Broker)?

Next is the questionnaire we sent to the developer of application named “Campus USP”.

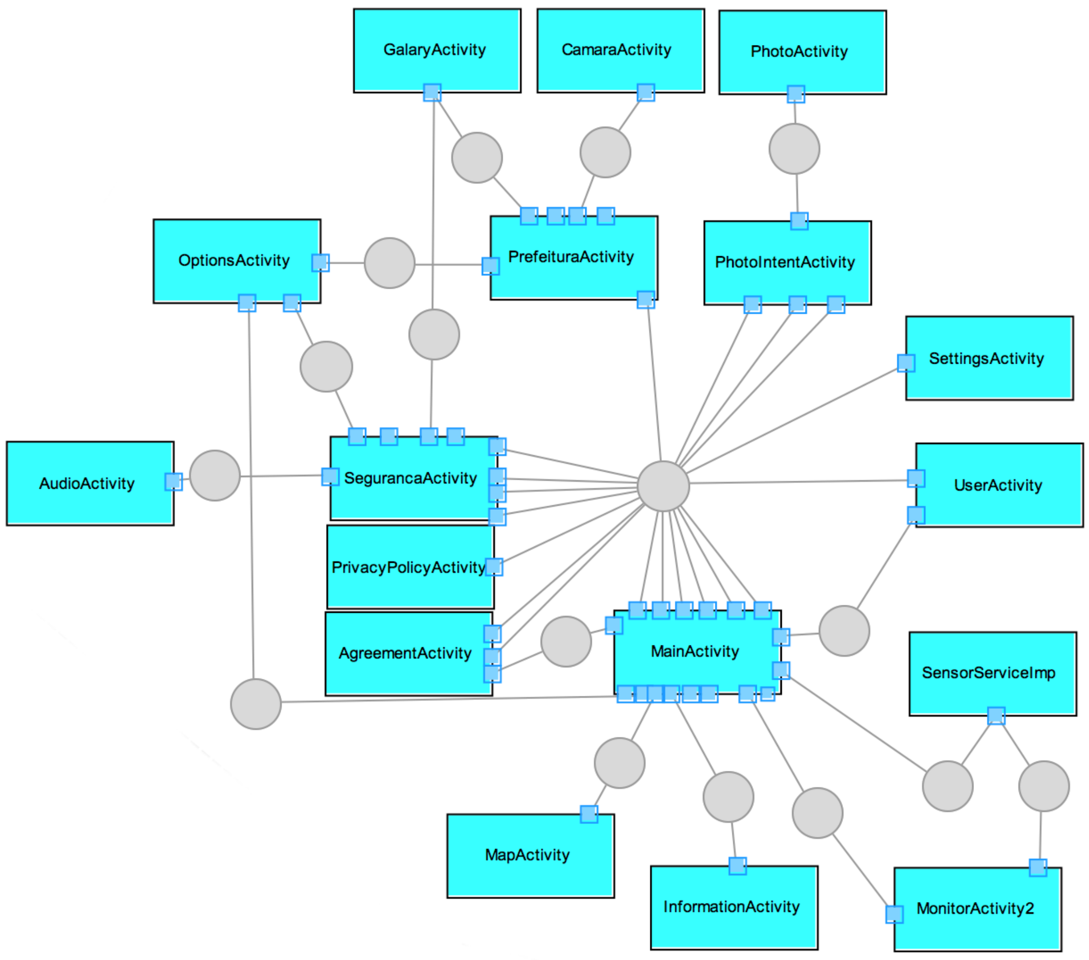
[3]Available at http://www.cs.cmu.edu/∼acme/AcmeStudio/
[4]Available at https://www.ics.uci.edu/∼seal/projects/covert/COVERT%20User%20Manual_2.0.pdf

Arquitetura do app "Campus USP"

Realizamos um estudo empírico para recuperar a arquitetura do aplicativo Campus USP, e gostaríamos de saber sua opinião sobre essa arquitetura gerada

* Required

Arquitetura - Campus USP



1. **Você reconhece esses componentes? Justifique.** *

2. **Quais desses componentes foram implementados por vocês?** *

Mark only one oval.

- GalaryActivity
- CamaraActivity
- PhotoActivity
- OptionsActivity
- PrefeituraActivity
- PhotoIntentActivity
- AudioActivity
- SegurancaActivity
- PrivacyPolicyActivity
- AgreementActivity
- MainActivity
- SetingsActivity
- UserActivity
- MapActivity
- InformationActivity
- MonitorActivity2
- SensorServiceImp

3. **Existe algum componente que foi fundamental na implementação, mas a ferramenta não identificou? Qual?** *

4. **Quais desses componentes são do próprio framework do Android?** *

5. **Olhando a arquitetura, é possível identificar o estilo arquitetural que foi utilizado (cliente servidor, publish subscribe, etc) ou alguma tecnologia utilizada (Fiware, Context Broker)? Justifique.** *

B.3 THE RECOVERED ARCHITECTURES

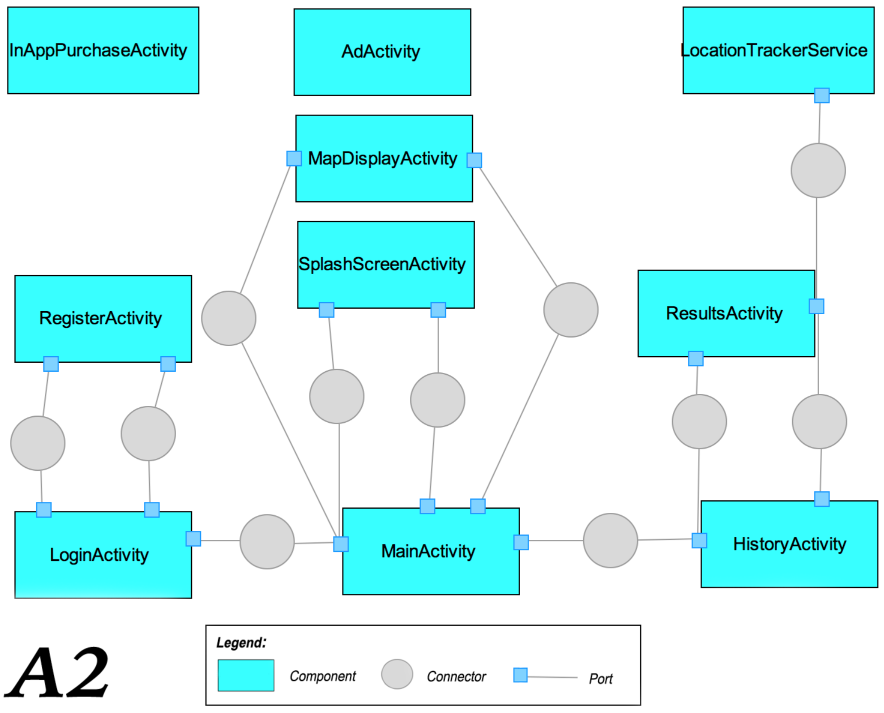


Figure B.1 Architecture of Application **A2** extracted from ACME

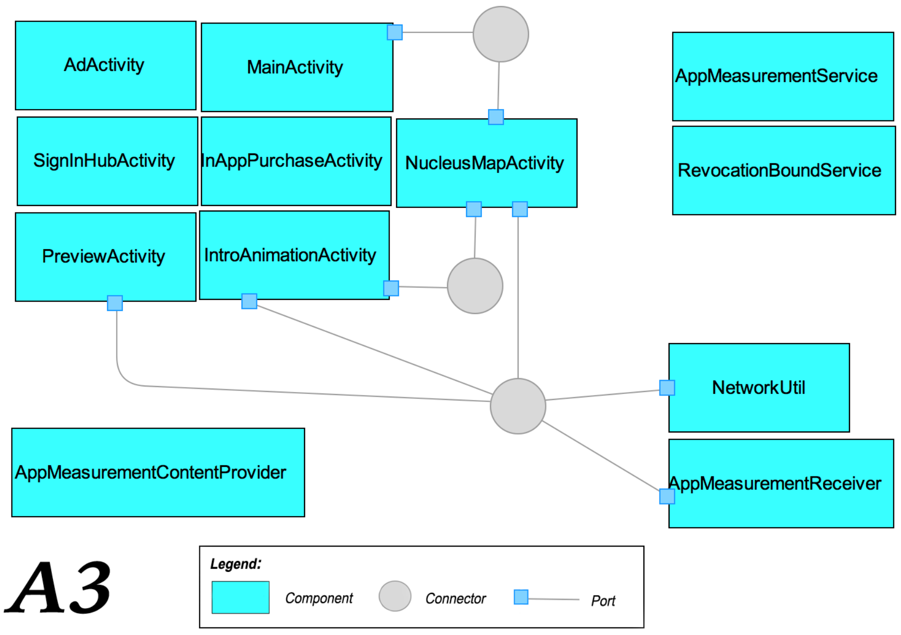


Figure B.2 Architecture of Application **A3** extracted from ACME

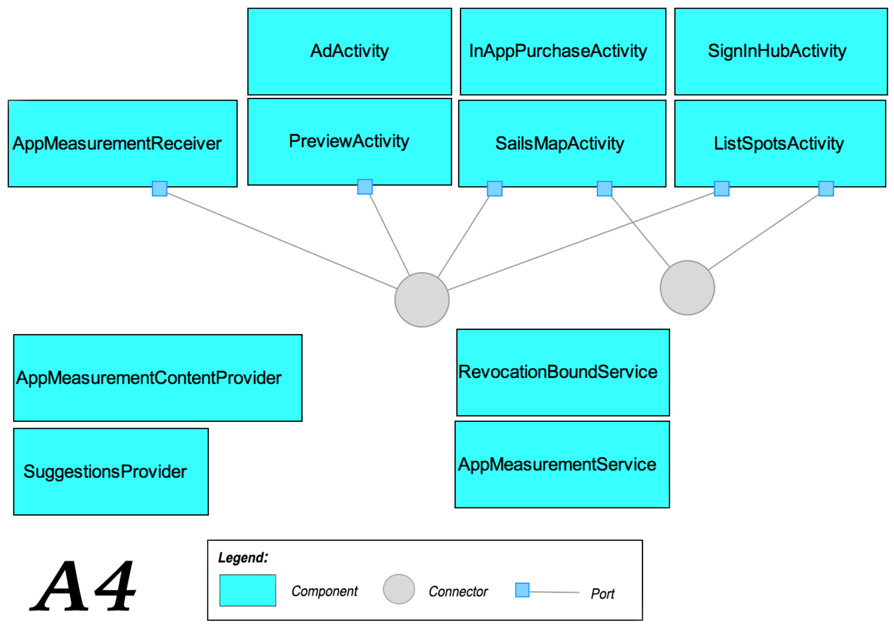


Figure B.3 Architecture of Application **A4** extracted from ACME

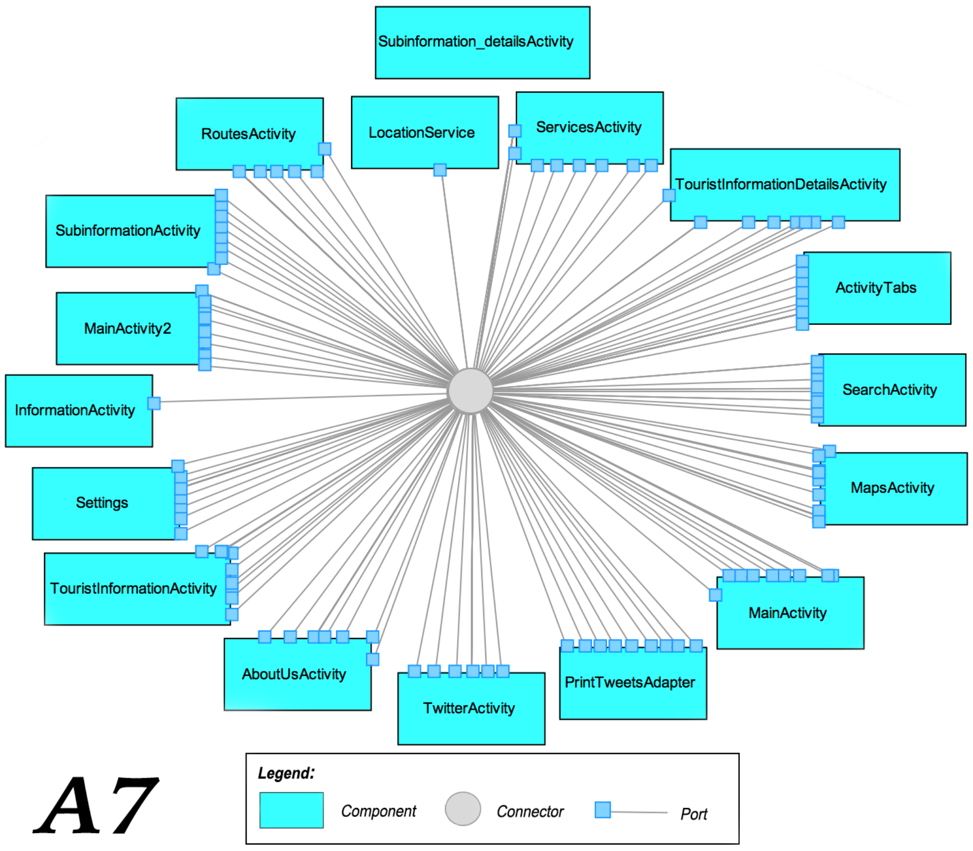


Figure B.4 Architecture of Application **A7** extracted from ACME

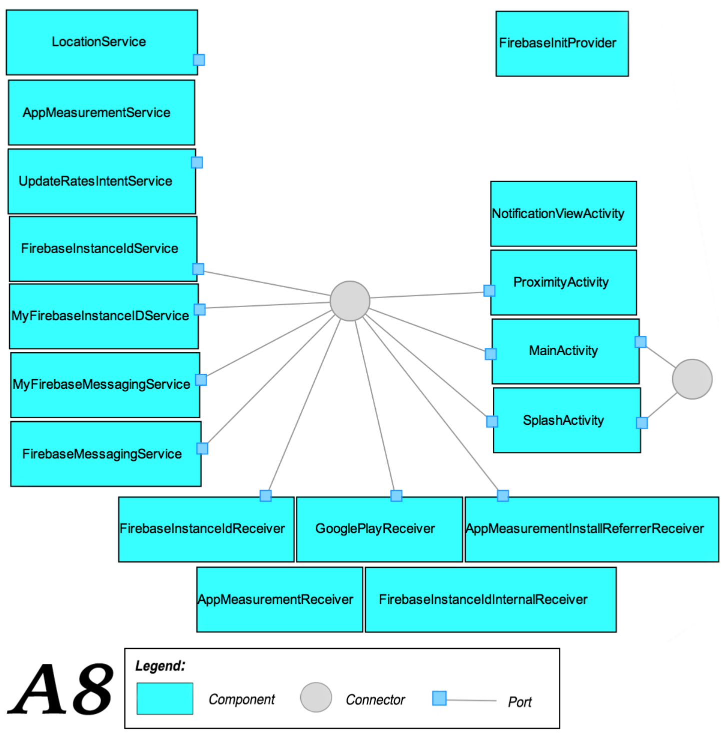


Figure B.5 Architecture of Application **A8** extracted from ACME

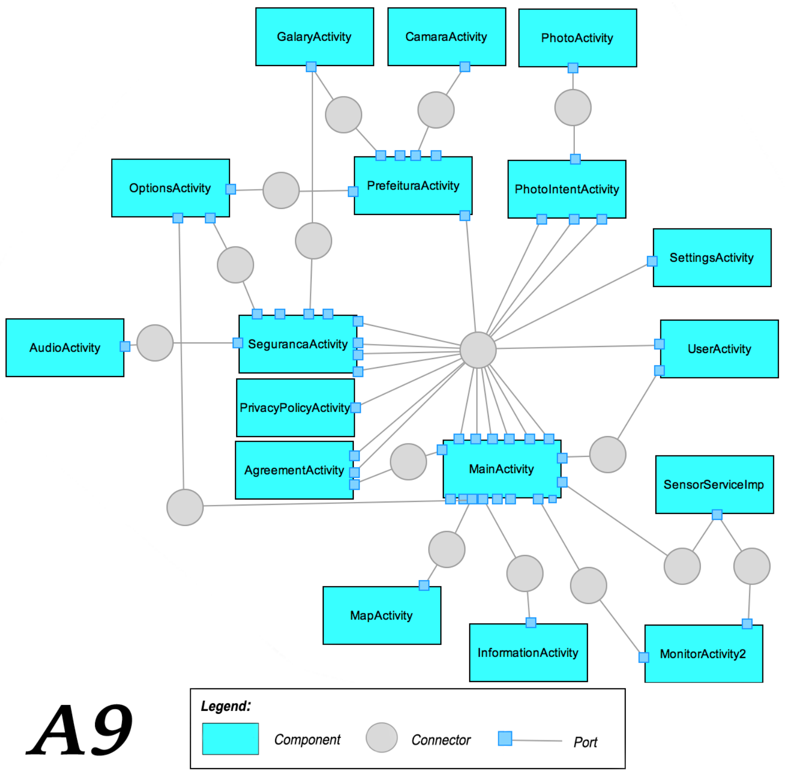


Figure B.6 Architecture of Application **A9** extracted from ACME

Appendix

C

GROUNDED THEORY ARTIFACTS

For the data analysis process we analyzed the evidence from the case study using Grounded Theory (GT) procedures to derive a theory. Next we provide the following artifacts: the axial coding diagram, and the theory evaluation questionnaire.

C.1 AXIAL CODING DIAGRAM

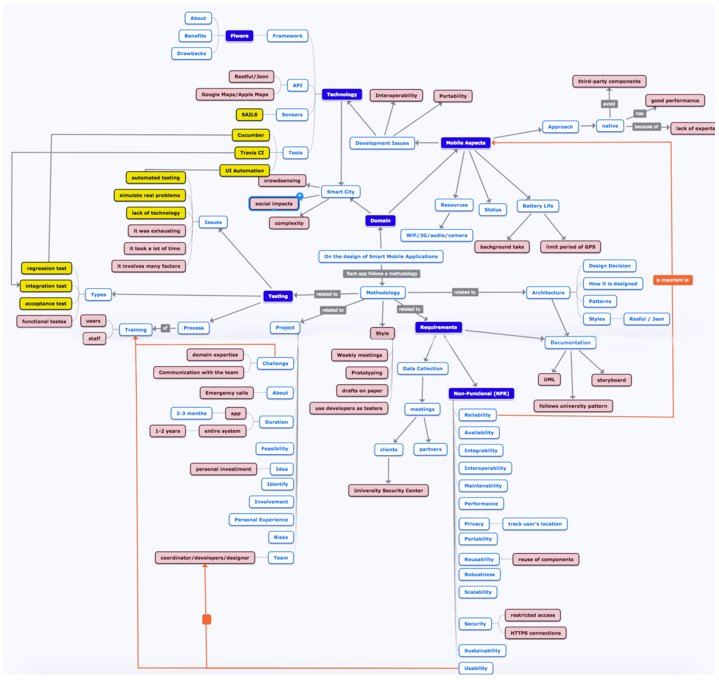


Figure C.1 Axial coding diagram (the complete diagram is available online at the supplementary material site)

C.2 THEORY EVALUATION QUESTIONNAIRE

Questionário de validação do modelo

Para obter mais informações da primeira parte deste estudo, acesse o nosso material suplementar: https://rose2s.github.io/EMSE2019

Este questionário visa validar o modelo teórico sobre o design de aplicações móveis no contexto de cidades inteligentes na perspectiva dos entrevistados.

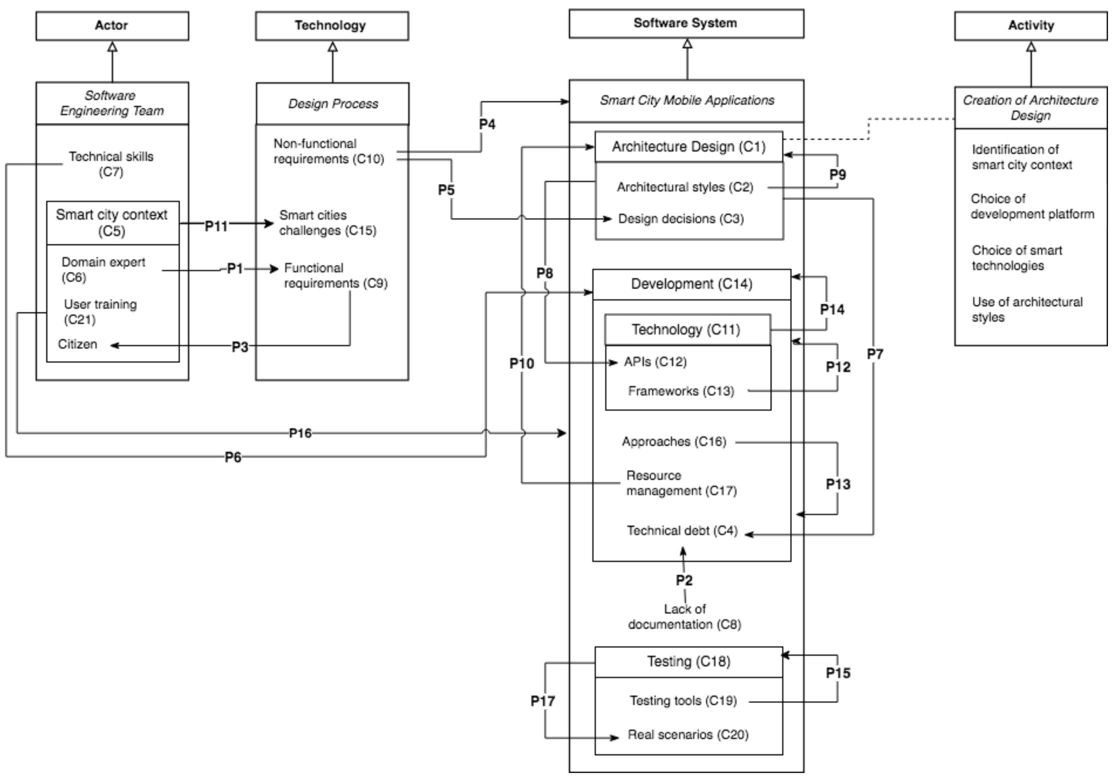
Obs: A palavra SCMA se refere a Smart City Mobile Applications

* Required

1. **Email address** *

2. **Nome e Instituição** *

Representação do modelo teórico


Actor
Software Engineering Team
Technical skills (C7)
Smart city context (C5)
Domain expert (C6)
User training (C21)
Citizen
Technology
Design Process
Non-functional requirements (C10)
Smart cities challenges (C15)
Functional requirements (C9)
Software System
Smart City Mobile Applications
Architecture Design (C1)
Architectural styles (C2)
Design decisions (C3)
Development (C14)
Technology (C11)
APIs (C12)
Frameworks (C13)
Approaches (C16)
Resource management (C17)
Technical debt (C4)
Lack of documentation (C8)
Testing (C18)
Testing tools (C19)
Real scenarios (C20)
Activity
Creation of Architecture Design
Identification of smart city context
Choice of development platform
Choice of smart technologies
Use of architectural styles
P1
P2
P3
P4
P5
P6
P7
P8
P9
P10
P11
P12
P13
P14
P15
P16
P17

3. **P1. Domain experts positively impact the definition of requirements for SCMA (Smart City Mobile App). P1. Especialistas de domínio (ex: bombeiro, advogado, e servidores) impactam positivamente na definição de requisitos para SCMA).** *

A elicitação de requisitos dessas aplicações depende fortemente dos especialistas de domínio da aplicação que, com base nas necessidades dos cidadãos, atuam como parceiros explicando aspectos importantes relacionados a esse domínio e validando.
Mark only one oval.

- Concordo
- Nem concordo nem discordo
- Discordo
- Não tenho informação suficiente para responder esta questão
- Other: ____________________

4. **Justifique P1**

____

5. **P2. The lack of documentation creates technical debt. (P2. A falta de documentação gera débito técnico).** *

Problemas como o desenvolvimento mais lento que o normal, o aumento da densidade de defeitos causado pelo acúmulo de débito técnico ou problemas/dificuldades ao testar o aplicativo por não ter os requisitos ou por a pessoa que detém o conhecimento da app ter saído do projeto.
Mark only one oval.

- Concordo
- Nem concordo nem discordo
- Discordo
- Não tenho informação suficiente para responder esta questão
- Other: ____________________

6. **Justifique P2**

____

____

____

____

____

7. **P3. Functional requirements for SCMA come from a smart cities context, which is citizen-oriented. (P3. Requisitos funcionais para SCMA surgem do próprio contexto de cidades inteligentes, que é orientado ao cidadão).** *

Esse apps são visam melhorar a vida do cidadão em áreas como utilidade pública, turismo, segurança pública, e portanto os requisitos são baseados nesse contexto.
Mark only one oval.

- Concordo
- Nem concordo nem discordo
- Discordo
- Não tenho informação suficiente para responder esta questão
- Other: ______________________

8. **Justifique P3**

____
____
____
____
____

9. **P4. Most non-functional requirements are not taken into consideration by the team when making design decisions. (P4. A maioria dos requisitos não funcionais não são levados em consideração pela equipe de desenvolvimento ao tomar decisões de design).** *

A equipe não se preocupa com requisitos não funcionais devido a limitações de projeto como tempo curto, recursos, etc.. A documentação principal é baseada em relatórios técnicos de alto nível, os quais não especificam as decisões de design tomadas.
Mark only one oval.

- Concordo
- Nem concordo nem discordo
- Discordo
- Não tenho informação suficiente para responder esta questão
- Other: ______________________

10. **Justifique P4**

11. **P5. The satisfaction of non-functional requirements depends upon the design decisions made. (P5. A satisfação de requisitos não funcionais depende das decisões de design tomadas).** *

SCMA, assim como qualquer app do mundo real, tem requisitos não funcionais conflitantes e o designer terá que fazer um trade-off entre eles. Por exemplo, trade-off entre privacidade do usuário e confiabilidade dos dados.
Mark only one oval.

- Concordo
- Nem concordo nem discordo
- Discordo
- Não tenho informação suficiente para responder esta questão
- Other:

12. **Justifique P5**

13. **P6. The level of technical skills impact the design/development of SCMA. (P6. O grau de conhecimentos técnicos afeta o desenvolvimento de SCMA).** *

A falta de domínio de tecnologias móveis e frameworks para o design e desenvolvimento de apps de smart city ocasionam no atraso do desenvolvimento.
Mark only one oval.

- Concordo
- Nem concordo nem discordo
- Discordo
- Não tenho informação suficiente para responder esta questão
- Other: ____________________

14. **Justifique P6**

____
____
____
____
____

15. **P7. The lack of architecture decision-making creates technical debt. (P7. A falta de decisão de design gera débito técnico).** *

Devido curto prazos de projetos, a solução mais rápida que os times de desenvolvimento encontram é negligenciar o design da arquitetura. A conseqüência disso é a criação de débito técnico no projeto, um custo implícito de retrabalho causado pela escolha de uma alternativa mais fácil, em vez de usar uma abordagem correta que levaria mais tempo.
Mark only one oval.

- Concordo
- Nem concordo nem discordo
- Discordo
- Não tenho informação suficiente para responder esta questão
- Other: ____________________

16. **Justifique P7**

17. **P8. The architecture design of SCMA is API-centric. (P8. O design arquitetural de SCMA é centrado em APIs).** *

A preferência por um estilo arquitetural sobre outro depende de como a arquitetura é projetada, mas em geral são centrados em chamadas de APIs.

Mark only one oval.

- Concordo
- Nem concordo nem discordo
- Discordo
- Não tenho informação suficiente para responder esta questão
- Other: ______

18. **Justifique P8**

19. **P9. The use of architectural styles positively impacts the design of SCMA. (P9. O uso de estilos arquiteturais impactam positivamente no design de SCMA).** *

SCMA usam estilos arquiteturais muito comuns em aplicativos de cidade inteligentes (cliente-servidor, publish-subscribe, message broker), e o uso de tais estilos impactam positivamente no design dessas aplicativos.
Mark only one oval.

- Concordo
- Nem concordo nem discordo
- Discordo
- Não tenho informação suficiente para responder esta questão
- Other: ____________________

20. **Justifique P9**

____
____
____
____
____

21. **P10. Effective mobile resource management positively impacts the architecture design of SCMA. (P10. O gerenciamento efetivo dos recursos móveis impacta positivamente no design de SCMA).** *

O gerenciamento efetivo da bateria, a escolha da plataforma de desenvolvimento (nativa, híbrida) e outros recursos impactam diretamente nos requisitos não funcionais de performance, segurança, portabilidade, privacidade, etc, impactando assim no design dessas aplicações.
Mark only one oval.

- Concordo
- Nem concordo nem discordo
- Discordo
- Não tenho informação suficiente para responder esta questão
- Other: ____________________

22. **Justifique P10**

23. **P11. The smart cities context brings more complexity to mobile applications. (P11. O contexto de cidades inteligentes traz mais complexidade para aplicativos móveis).** *

Mesmo que SCMA pareça ser um domínio de aplicações simples, há um grande número de interações entre entidades em cidades inteligentes que adiciona complexidade as aplicações, e o back-end desses app geralmente é muito complexo, pois tem que lidar com sensores, analisar inputs vindas da multidão (crowdsourcing), integrar muitos componentes internos, e chamada de APIs.

Mark only one oval.

- Concordo
- Nem concordo nem discordo
- Discordo
- Não tenho informação suficiente para responder esta questão
- Other:

24. **Justifique P11**

25. **P12. The development or adoption of smart city frameworks reduces the complexity of the development of SCMA. (P12. O desenvolvimento ou adoção de frameworks para apps de cidades inteligentes reduz a complexidade do desenvolvimento de SCMA).**
*

A equipe de desenvolvimento optam por utilizar frameworks complexos de smart cities como o Fiware (uma plataforma genérica para especificação de arquitetura em cidades inteligentes) ou desenvolver seu próprio framework de acordo com suas necessidades visando benefícios como (1) maior facilidade para integrar componentes e aplicativos e (2) fornecer um conjunto de APIs que facilitam o design, e (3) estabelecer comunicação entre todos os componentes.
Mark only one oval.

- Concordo
- Nem concordo nem discordo
- Discordo
- Não tenho informação suficiente para responder esta questão
- Other: ______________________________

26. **Justifique P12**

____
____
____
____
____

27. **P13. The level of development challenges in SCMA depends upon the development approach adopted. (P13. O grau de desafios de desenvolvimento em SCMA depende da abordagem de desenvolvimento escolhida).** *

Desenvolvimento multi-plataforma gera o mesmo aplicativo para várias plataformas, mas afeta o desempenho do aplicativo. Ao mesmo tempo, a abordagem nativa se torna muito custosa pois além de demorar mais, é necessário ter desenvolvedores experts em mais de uma linguagem de programação. Com isso, muitos desenvolvedores têm usado frameworks para utilização de desenvolvimento híbrido, mas os mesmos já relatam outros problemas devido a essa abordagem.
Mark only one oval.

- Concordo
- Nem concordo nem discordo
- Discordo
- Não tenho informação suficiente para responder esta questão
- Other: ____________________

28. **Justifique P13**

____
____
____
____
____

29. **P14. Smart technologies positively impact the development speed of SCMA. (P14. Tecnologias inteligentes impactam positivamente a velocidade do desenvolvimento de SCMA).** *

O uso de APIs de mapas, beacons e triangulação são tecnologias amplamente utilizadas em aplicações crowdsensing. Além disso, aplicações de smart city em tempo real (exemplo, apps de emergência), a fim de mitigar problemas relacionados a latência de informações, aceleraram o seu desenvolvimento através do uso de tecnologias.
Mark only one oval.

- Concordo
- Nem concordo nem discordo
- Discordo
- Não tenho informação suficiente para responder esta questão
- Other: ____________________

30. **Justifique P14**

31. **P15.The lack of testing tools for mobile applications negatively impacts the testing of SCMA. (P15. A falta de ferramentas de teste para aplicativos móveis afeta negativamente o teste de SCMA).** *

Entrevistados relataram que teste para SCMA é desgastante, leva muito tempo e não há suporte adequado por meio de ferramentas de teste.

Mark only one oval.

- Concordo
- Nem concordo nem discordo
- Discordo
- Não tenho informação suficiente para responder esta questão
- Other: ______

32. **Justifique P15**

33. **P16. Training users positively impacts the product acceptance of SCMA. (P16. O treinamento de usuários impactam positivamente na aceitação do produto de SCMA).** *

Um grande desafio em testes de SCMA são os profissionais que usam o aplicativo para apoiar um cidadão que relatou um problema no aplicativo. Por exemplo, um segurança que recebe uma ocorrência de emergência através do aplicativo e usa essas informações para solucionar o incidente.
Mark only one oval.

- Concordo
- Nem concordo nem discordo
- Discordo
- Não tenho informação suficiente para responder esta questão
- Other: ______________________

34. **Justifique P16**

____
____
____
____
____

35. **P17. Testing SCMA involves the need to simulate real-world scenarios. (P17. Testar SCMA envolve a necessidade de simular cenários do mundo real.).** *

A falta de ferramentas de teste para aplicativos móveis torna-se ainda mais sério no contexto de cidades inteligentes, onde o teste também envolve a necessidade de simular cenários do mundo real.
Mark only one oval.

- Concordo
- Nem concordo nem discordo
- Discordo
- Não tenho informação suficiente para responder esta questão
- Other: ______________________