Extracting Technical Domain Knowledge to Improve Software Architecture

Faculty of Science, Engineering and Technology
Swinburne University of Technology
Melbourne, Australia

Submitted for the degree of Doctor of Philosophy

Scott Barnett

2017
Abstract

This thesis advances the state of the art in terms of our understanding of the influence the technical implementation details have upon software architecture. We propose the concept of the technical domain to capture the implementation concerns of a specific domain as well as viable solutions known as domain tactics. Rather than relying on experience or rediscovering domain specific architectural knowledge, we capture this information in a technical domain model. This model encourages reuse of domain specific patterns, is suitable for tradeoff analysis at the requirements stage, and acts as a teaching aid for new developers to the domain.

Furthermore, we present a method for defining a technical domain and mining architectural knowledge. Using this method developers can enhance their understanding of a technical domain. To evaluate our approach we developed a prototype toolchain, RAPPT, to realise an implementation based on the principles of the technical domain in the context of mobile apps. RAPPT is a model based tool that generates architectural scaffolding that developers then extend to finish off their mobile apps.

Finally, we evaluated our approach and RAPPT with a user evaluation, commercial and open source case studies, and expert interviews. All of the evaluations focused on the technical domain of mobile apps with initial work for the technical domains of smart homes and AI solutions available in the Appendix. We found that our approach captures architectural knowledge that can benefit novice developers and a toolchain that can be used in the development of commercial apps. However, the domain specific architectural knowledge needs to be presented in a way for professionals to quickly find the information they need and should include working code snippets as examples.
Dedicated to all of my teachers, family and friends.
Acknowledgements

I would like to acknowledge the support, encouragement and guidance provided by my advisers Prof. Rajesh Vasa, Prof. John Grundy and Assoc. Prof. Jean-Guy Schneider. I am also indebted to the team at Deakin Software and Technology Innovation Laboratory for providing feedback on the research and suggestions for improving RAPPT. Thanks to Dr. Leonard Hoon, Maria Mitrevska, Dr. Iman Avazpour, Dr. Anthony Tang, Prof. Rick Kazman, Dr. Mohamed Abdelrazek and Dr. Andrew Cain for their valuable discussions. I would also like to thank all the participants of the user studies for their time and patience. I am grateful to Swinburne University of Technology for providing the resources and opportunity to pursue a research higher degree.

Finally, I would like to thank my parents Neil and Julie Barnett for their support and motivation.

Scott Barnett, 2017
Declaration

I declare that this thesis contains no material that has been accepted for the award of any other degree or diploma and to the best of my knowledge contains no material previously published or written by another person except where due reference is made in the text of this thesis.

Scott Barnett, 2017
Publications Arising from this Thesis

The work described in this thesis has been published as described in the following list:


2.4.3 Mobile App Development Frameworks .......................... 29
2.5 Software Architecture ............................................. 31
  2.5.1 Quality Tactics Lack Specificity ............................ 33
  2.5.2 Design Patterns are Too Low Level ....................... 33
  2.5.3 Technical Domains Lack an Architectural Viewpoint .... 34
  2.5.4 Reference Architectures lack Technical Domain Concerns 36
2.6 Model Driven Development ....................................... 37
  2.6.1 Model Driven Architectures ............................... 38
  2.6.2 MDA and the Technical Domain ............................ 38
  2.6.3 Platform Independent Models are Insufficient ............ 39
  2.6.4 Software Product Lines ................................. 40
2.7 Summary .................................................................. 42

3 Research Methodology .................................................. 44
  3.1 Introduction ......................................................... 44
  3.2 Method ............................................................. 46
    3.2.1 Stage 1. Capturing Technical Domain Concerns .... 46
    3.2.2 Stage 2. Modelling Domain Specific Abstractions .... 47
    3.2.3 Stage 3. Automation of the Technical Domain ....... 48

4 The Technical Domain .................................................... 50
  4.1 Domain Tactics, Architectural Tactics and Patterns ........ 51
  4.2 A New Method for Mining Technical Domain Knowledge ... 53
6 Rapid APP Tool (RAPPT) 120

6.1 Motivation ........................................ 121

6.2 Influence of the Technical Domain ............... 125
   6.2.1 Multiple Abstraction Levels in Mobile Apps ........ 125
   6.2.2 Tool Stakeholders ............................. 127
   6.2.3 Multiple Views of a Meta-model ............... 130

6.3 The RAPPT Approach ............................. 133
   6.3.1 Partial Model ................................ 134
   6.3.2 Generate an App for a Single Platform .......... 137
   6.3.3 Multiple Views with Overlapping Abstraction Levels 139
   6.3.4 Partial Synthesis ............................. 141

6.4 Architecture and Implementation ................. 142
   6.4.1 Development Architectural View ................ 142
   6.4.2 Model Transformation .......................... 145
   6.4.3 Deployment Architectural View .................. 148
   6.4.4 Technology Choices ........................... 149

6.5 Summary ......................................... 151

7 Evaluation and Discussion 152

7.1 Interviews and Questionnaire: Evaluating the Technical Domain Model ...................... 154
   7.1.1 Preparation .................................. 154
   7.1.2 Participants .................................. 156
   7.1.3 Results and Discussion ....................... 157
7.1.4 Threats to Validity ........................................ 164

7.2 Replicating Existing Apps .................................... 166

7.2.1 App Selection ............................................ 167

7.2.2 Findings .................................................. 168

7.2.3 Threats to Validity ........................................ 169

7.3 User Evaluation ................................................. 170

7.3.1 Experiment Setup ........................................ 171

7.3.2 Results and Discussion .................................... 173

7.3.3 Threats to Validity ........................................ 182

7.4 Case Studies: Technical Domain Analysis ................... 182

7.4.1 Malaysian Bus Tracker .................................... 183

7.4.2 Google IO 2014 .......................................... 188

7.4.3 Threats to Validity ........................................ 195

7.5 Industry Case Study: Prompa ................................. 196

7.5.1 Study Questions .......................................... 198

7.5.2 Study Propositions ....................................... 199

7.5.3 Case Study Design ....................................... 200

7.5.4 Findings and Discussion .................................. 201

7.5.5 Threats to Validity ........................................ 202

7.6 Summary ....................................................... 203

8 Conclusions and Future Work .................................. 205

8.1 Conclusions ................................................... 205
8.2 Future Work ........................... 208
  8.2.1 Generalisation .......................... 208
  8.2.2 Automatic Layout and Theme Extractor ........... 208
  8.2.3 Reverse Engineering .......................... 209
  8.2.4 Cross Platform ............................. 209
  8.2.5 Live Visualisation ............................ 210
  8.2.6 3rd Party Frameworks .......................... 210
  8.2.7 Extensibility ............................... 211
8.3 Closing Thoughts ............................. 211

A Building Data Intensive Mobile Apps 213
B Domain Tactics .............................. 221
C RAPPT's Interface ............................ 235
D User Interviews ............................... 242
E Ethics Application ............................. 294
F Meta-model Process Examples ................. 363
G User Evaluation ............................... 373
H User Evaluation Criteria ........................ 388
I Mobile App Tools .............................. 389
J Analysed Apps ................................. 392
CONTENTS

K  Prompa Case Study 394

L  Generalisation 400

References 402
List of Figures

1.1 The general concerns of mobile app development (shown in the large box) and Android specific concerns (inset). 4

2.1 Relationship between the Event Management and E-Banking application problem domains, with the technical solution and implementation domains. 14

2.2 Key challenges from the technical domain facing mobile app developers. 16

2.3 Impact of the Technical Domain on Software Requirements 22

3.1 Research method used to address our goal to improve our understanding of how the implementation details influence software architecture. Coloured boxes indicate the outcome for each stage, Stage 1 Blue and Red, Stage 2 Green. 45

4.1 Method for identifying technical domain knowledge. 54

4.2 Steps for Process 2 with the output from Process 1 as input. 62

4.3 An example of a Technical Concern, Quality Attribute and Problem Context for the software operating in a trauma centre. 65

4.4 Relationship between Technical Concerns, Quality Attributes and the Problem Context 66

4.5 Influences on a Software Architect [1] with added technical domain knowledge. 66
List of Figures

4.6 Relationship between Technical Concerns, Quality Attributes, Problem Context and Domain Tactics ............... 71

4.7 Concerns and relationships that make up the technical domain model for data-intensive mobile apps. White boxes show how the concerns relate to a mobile app .................. 77

4.8 lifecycle .................................................. 82

4.9 lifecycle .................................................. 83

5.1 Phase 1 process: (Outputs are rectangles, ovals are steps) 103

5.2 An example of the Dashboard classification for mobile app screen. ................................................. 107

5.3 An example of the Splash classification for mobile app screen. ......................................................... 107

5.4 An example of the Collection classification for mobile app screen. .................................................... 108

5.5 An example of the Complex Collection classification for mobile app screen. ................................ 108

5.6 An example of the Multicollection classification for mobile app screen with two components a 1) Horizontal scrolling widget and 2) a vertical scrolling component. ................. 108

5.7 An example of the Detail classification for mobile app screen.109

5.8 An example of the Inform classification for mobile app screen.109

5.9 An example of the Form classification for mobile app screen.110

5.10 Phase 2 process for building a DSL and evaluating generated apps. ................................................. 111

5.11 The process for evaluating the meta-model and DSL with professional mobile app developers. ................. 114

5.12 The relationship between the technical domain concerns, meta-model, functional and non-functional requirements. 116
5.13 A UML class diagram showing the meta-model for data-intensive mobile apps. .................................. 117

6.1 The Popular Movies, About, and Details screens for MovieDB, an app based on the MovieDB API. .................... 122

6.2 The different view types of a meta-model all of which extend the concept of a view. .......................... 131

6.3 RAPPT assists software developers by generating the initial architecture for their app. ....................... 134

6.4 Levels of abstraction in a mobile app. .................. 140

6.5 Major architectural components of RAPPT ................... 143

6.6 An example of our approach used to describe a list item that displays data fetched from a webservice. .......................... 146

6.7 Example transformation showing how a row controller class is synthesised from the Partial Model. .......................... 147

6.8 The deployment of RAPPT's key components. .................. 148

7.1 Participant responses to the demographic questionnaire. .................. 157

7.2 Example screens generated by RAPPT for the Centrelink app. .................................. 166

7.3 Screenshots showing the app for the Movie DB evaluation task. .................................. 173

7.4 D1 - How many years experience do you have in mobile app development? .......................... 174

7.5 D2 - How many apps have you built? .................. 174

7.6 User demographic questions 1 - 2 and participants' responses. .................................. 174

7.7 D3 - What is your background? .................. 175

7.8 D4 - What is the highest qualification you have? .................. 175
7.9 D5 - How would you rate your skills to be able to build a simple data driven mobile app? .................................. 175

7.10 User demographic questions 3 - 5 and participants’ responses. ................................................................. 175

7.11 Q1 - How confident are you to build a small data-driven app with RAPPT? Scale ranked: 1(Not very confident) to 5 (Very confident) ........................................ 177

7.12 Q2 - How well do you understand the constructs in the RAPPT? Scale ranked: 1(Not very well) to 5 (Very well) ... 177

7.13 Q3 - How would you rank your abilities as an Android app developer? Scale ranked: 1(Poor) to 5 (Excellent) ........ 177

7.14 Post learning task questions and participants’ responses.
All questions were on a scale 1-5 ........................................ 177

7.15 UI screen designs for the Malaysian Bus Tracker Customer app. ................................................................. 184

7.16 Main data sources for Google IO 2014 app. .................. 190

A.1 Features of a screen from the Eventbrite app that need to be implemented by the developer. ......................... 216


C.2 Screenshot of the AML editor: 1. DSTL Editing Pane, 2. Compiler Errors Pane, and 3. Samples Pane. ............ 238

C.3 Code Browser. 1. Project Directory Pane and 2. Generated Code Browser. ................................................. 240

C.4 Complete source code for the MovieDB app showing code generated by the DSVL, code reused from copy-pasting AML samples and manually edited DSTL code. .......... 241

F.1 A Complex screen taken from the Footy Live app. ........ 364
List of Figures

F.2 Subset of classification framework for mobile app screens without support for Complex screen. .......................... 365
F.3 Classification framework for mobile app screens including the Complex screen. ............................................. 366
F.4 MovieDB app showing actions from the action bar. ........ 367
F.5 Subset of the meta-model showing View, Style and Event that can be used to model actions. ............................... 367
F.6 AML code for OAuth authentication with the Twitter API (A) and generated twitter login screen (B) ......................... 369
F.7 Actual Twitter app (A) compared with the generated Twitter app (B). Major differences in the UI are numbered 1 - 4 . 370
F.8 AML code from an early version describing a screen that showed event details to the user. ................................. 371

L.1 The start of a technical domain model for a “smart home”. 400
L.2 The start of a technical domain model for developing “AI Solutions”. .......................................................... 401
List of Tables

2.1 Comparison of mobile app Domain Specific Languages. Key: - for unsupported features, y for supported features and (y) if partially supported. The grey highlighted area shows the gap in the current approaches. 26

2.2 Comparison of mobile app graphical tools. Key: - for unsupported features, y for supported features and (y) if partially supported. The grey highlighted area shows the gap in the current approaches. 28

2.3 Comparison of app development frameworks. Key: - for unsupported features, y for supported features and (y) if partially supported. The grey highlighted area shows the gap in the current approaches. 30

2.4 Common Architectural View Model. 35

4.1 Table showing the 6 questions to define a technical domain. 55

4.2 Table showing Problem Contexts and related Domain Tactics for data-intensive mobile apps. 92

5.1 The 30 apps selected for use in Phase 1 and Phase 2. 101

5.2 Baseline Classification Framework derived from Google and Apple’s UI guidelines and initial apps. 104

5.3 The Refined Classification Framework derived from app analysis. 106
List of Tables

6.1 The level of support for the major components of a mobile app in RAPPT’s DSL ........................................ 136

6.2 Example visual elements that make up RAPPT’s Visual Language ................................................................. 141

7.1 Interview questions asked to all participants .................. 158

7.2 Problem Contexts for mobile app development ............. 159

7.3 Importance of problems as indicated by the participants. ... 160

7.4 Interview questions for participants who indicated they were app developers .............................................. 161

7.5 Interview questions for participants who indicated they were not mobile app developers ......................... 163

7.6 Apps selected from the top ranked apps on the Google Play Store ................................................................. 168

7.7 Survey questions and their associated research questions. 178

7.8 Sample questions of the questionnaire. Likert points have been given score of 1 to 5 representing *Strongly Disagree* to *Strongly Agree*. ................................................................. 180

7.9 LOC for each of the major packages in Google IO 2014 ... 189

J.1 Table showing data-intensive mobile apps used in analysis. 393
Chapter 1

Introduction

1.1 Research Challenges

The conventional view in software engineering states that software quality is achieved through good requirements [2], appropriate architecture [1], a productive and skilled team [3], and suitable processes [4]. In Software Architecture, current best practices recommend using Design Patterns [5], Quality Tactics [1], Reference Architectures [6], and automation such as advocated by Model Driven Development (MDD) [7] to achieve quality.

However, these approaches lack the specificity required for addressing challenging technical implementation concerns. Architectural decisions are also costly and time consuming to change later in the development process [8], so it is paramount that good decisions and design choices are made up front. To guide architects in this process, there are architectural standards such as the ISO Standard 42010 [9] and architecture views [1], but these lack support for analysing implementation concerns – domain specific implementation details such as mobile sensor usage cannot be expressed using these techniques.

Architects also use techniques such as MDD which has been observed to improve the quality of architecture and automation [7,10–12]. MDD consists of techniques for raising the level of abstraction through the
use of models. These high level models undergo numerous model transformations ending in a code synthesis step to realise a concrete architecture. MDD approaches such as Software Product Lines [13–16], Domain Specific Languages (DSL) [17–19] and Model Driven Architectures [10, 20] lack guidance for ensuring that 1) a model captures the essential concepts required for generating quality architecture, 2) transformations from model to code will result in a high quality architecture and 3) template creation is left as an implementation detail.

Reference architectures [6], if available, are another popular solution to achieve quality as they demonstrate a working solution. However, reference architectures do not provide a means for integration into a model based tool, have a steep learning curve and are costly to build [21, 22]. While reference architectures assist in following best practices, architects are unclear as to the clear interpretation of what reference architecture means, find that they are too abstract, have bad documentation, are of poor quality or too limiting [23].

In addition to these gaps, developers need to address technical implementation concerns when realising an architecture. These technical implementation concerns are issues that arise when developers start to implement an architecture. Typically these problems are domain specific although similar concerns are shared across domains. These technical implementation concerns are covered in more detail in Chapter 4.

In this thesis we address the lack of support for analysing the implementation concerns by defining a technical domain model and by automation for the domain of mobile apps. We chose the domain of mobile apps due to their widespread growth and use, ability to have a tight focus on a single domain, and limited prior work that focuses exclusively on mobile in the literature.

In 2017, Gartner estimates that apps will generate $77 billion and that mobile apps will be downloaded more than 268 billion times.¹ Currently there are well over 1 million apps available for each of the major

¹http://www.gartner.com/newsroom/id/2654115
mobile app platforms\(^2\) and many new apps are released daily. These apps face plenty of competition due to the low barrier of entry for individual developers [24].

Unlike traditional software distribution methods, apps are distributed through an app store\(^3,4\) and are readily available for download. More importantly, users rank and review these apps and comment on issues they have with an app such as frequent crashes. Success of an app, downloads, is often riding on favourable reviews. Recent research shows that generally user rankings do not change over time [25] and there is emerging evidence that mobile app users have short attention spans [26], together this creates pressure on developers to produce high quality apps early in their evolution – in effect they may not have the luxury to slowly improve quality and if errors are reported, developers need methods where they can rapidly reason and respond.

In this context, developing an appropriate architecture is difficult as there are many implementation details that developers need to get right and mobile phones also bring with them additional non-trivial constraints – limited memory, inherently unreliable mobile networks with changing speeds, a diverse set of sensors, high priority interruptions that need to be addressed (e.g. phone rings), limited screen space, and a relatively low-powered processor. Dealing with mobile data transmission is also a fundamental concern when building mobile apps and is fraught with many architectural problems [27]. Specifically there are issues with the availability of data due to variable network connectivity, lost data due to a dropped network connection and unavailable services due to no network connection. In many ways, mobility brings challenges that traditional desktop and web applications do not need to treat as first-class citizens when designing the architecture.

Creating a quality architecture is also challenging in the mobile domain as developers have to understand a vast number of concepts to design, build and test mobile apps as shown in Figure 1.1. A standard mobile

\(^3\)https://play.google.com/store?hl=en
app screen that displays a list of data to the user consists of multiple UI components (e.g. labels, buttons, images etc.), data formatting, custom styles, integration with the apps primary and secondary navigation patterns, search functionality, filtering, contextual actions, event handling and data retrieval. At the visual and data binding level, there is a lot for a developer to get right when building the user interface of a mobile app. Furthermore, a significant number of mobile apps rely on external data sources to provide the core functionality. A typical app consists of many aspects including visual components, background services, functional aspects, data model, security model, connection to API, caching data, rendering the data within a certain time interval, feedback model, and data persistence. In addition each mobile platform have additional concerns that need to be considered (see Figure 1.1 insert).

![Figure 1.1:](https://developer.android.com/guide/index.html)

In summary, current Software Architecture best practices lack the specificity required for addressing the technical implementation concerns of mobile app development. Quality architecture is of utmost importance in the domain of mobile apps where quality plays a vital role to the success of an app due to end user engagement through user reviews.

---

However, building quality architecture is a non-trivial task as there are many concerns developers need to address and domain specific considerations such as network connectivity when building mobile apps.

*Fundamentally, current literature does not consider the impact technical implementation concerns have on software quality or view them as a first class citizen.*

In this thesis, we will explore the importance of the technical implementation concerns when building mobile apps. This will be done by studying the technical concerns of mobile app development, adapt well understood techniques from MDD to leverage automation to address these technical concerns and realise an implementation to evaluate our approach with practitioners.

### 1.2 Research Questions

Software development begins by understanding the domain [28], the essential concepts and the relationships between them. These domain entities improve communication between stakeholders as they all share a common vocabulary that can be used to describe the problem. Developers create domain models which include the domain entities and their relationships to better understand the business requirements. Domain models are also a first step before building parts of the system such as the database schema.

Typically a domain model is associated with a specific business domain but there are other definitions of a domain. For example a domain can encapsulate the data, functional and procedural aspects as well. These domain models focus on an aspect of the system that relates to the *functional requirements* of a system. However, it is known that the *non-functional requirements* exert the strongest influence upon software architecture [1]. Thus a domain model that relates to the *non-functional requirements* is vital for achieving a high quality architecture. In this thesis we use the term *technical domain* to refer to the concepts that have an influence on the non-functional requirements of a system.
The overarching goal of this thesis is to improve our understanding of how the implementation details influence software architecture. To do this we will consider software architecture from the context of mobile app development.

To achieve this goal we propose the following three research questions:

**RQ1. What are the concerns of the technical domain that influence architecture and how can this information be captured?**

To answer this research question we will explore the role that the domain plays in influencing architects’ early design decisions. We also want to create a method for extracting this information in a reusable manner so that other developers can benefit from this knowledge. We will focus on extracting reusable information from the technical domain of mobile apps.

**RQ2. How do we empirically build a meta-model that contains the core abstractions a developer needs to use to build mobile apps?**

Model-based tools rely on an underlying meta-model to capture the core abstractions required for describing the system. Building better meta-models ensures that tools capture the core concepts used by practitioners to realise quality apps. It will also allow us to provide insights into how to develop the right meta-model for the problem at hand. Answering this question will also provide us with a set of guidelines for the development of meta-models for other domains.

**RQ3. How does the technical domain influence the design and implementation of tools for automation?**

Answering this research question shows how automation can be achieved by combining the concepts developers use to build apps with the core principals of app architecture. We hope to establish principles for a new generation of model-based tools that consider the influence of the technical domain at three levels: 1) the interface to the model-based tool, 2) internal model representations and model transformations, and 3) the generation of robust architectural scaffolding. We will build upon the
meta-model developed to answer RQ2 so our tool will be focused on the technical domain of mobile apps.

Each of these research questions is addressed in more detail in the following chapters.

1.3 Research Methodology

To improve our understanding of how the implementation details influence software architecture we focus our research approach 1) on empirical analysis to better understand the influences acting upon the architecture of mobile apps, 2) develop a method and means for automating it to evaluate the method as well as the potential and viability of automation to ensure it is useful to real developers, and 3) run a series of surveys and user evaluations to understand how developers create and evolve mobile apps. Below we describe our approach for addressing each of our research questions. Although the steps are grouped and explained in a linear narrative structure the underlying process was far more inter-dependent.

To address RQ1. What are the concerns of the technical domain that influence architecture and how can this information be captured? we did a comprehensive analysis of the domain including documentation, relevant literature and source code of apps. Once we had an understanding of the domain we used this information to inform a second analysis stage designed to extract architectural knowledge.

To answer RQ2. How do we empirically build a meta-model that contains the core abstractions a developer uses to build mobile apps? we analysed a set of mobile apps and realised a meta-model in a primitive tool. Each of the apps were analysed from the user perspective to determine the key elements that make up an app. We then followed an iterative process to extract the concepts for the meta-model. During each iteration we modified the meta-model and the process in which we analysed the apps. We built a DSL and an appropriate translator engineering tool chain to validate the viability of the meta-model to describe new apps.
At this stage we discovered the key gap that the technical domain is important and the current recommendations in the literature for software architecture and MDD were insufficient.

To answer RQ3. How does the technical domain influence the design and implementation of tools for automation? required the development of a tool in order to evaluate our method. Based on the results of the previous two research questions we derived a set of principles for a model based tool. Our evaluation involved a series of surveys of professional developers to determine the viability of automation and a user study to evaluate the concepts of our meta-model. We also conducted a series of case studies to explore how architectural knowledge can benefit the requirements analysis stage.

1.4 Key Research Contributions

In this thesis we address the problem of current literature failing to consider the implementation details as a first class citizen for mobile app development. The main contribution of this thesis is a technical domain model for mobile apps that guides developers in making informed design decisions that improve quality.

Key contributions of this thesis also include:

**Major contributions**

- A Technical Domain Model for data-intensive mobile apps. We used our method for analysing the domain of data-intensive mobile apps and constructed a model of the core influences they have on software architecture. Developers can use this model to understand the unique characteristics of building a mobile app which differs from developing other systems such as desktop applications. Preliminary work on two additional technical domains of Smart Homes and AI Solutions are available in Appendix L.

- A method for constructing a Technical Domain Model. This method
provides a way for developers to analyse the impact a specific domain has on quality attributes. By using this method developers can identify the core elements that influence software architecture before they start development. The method developed in this thesis focuses on the building a method from the technical domain of mobile apps.

• A method and model for mining reusable domain specific architectural knowledge. Our technical domain model can be used to further analyse the domain to identify and extract architectural knowledge. The result of this stage is a set of Domain Tactics for the technical domain of mobile apps.

• Domain Tactics. Domain specific information is documented in a Domain Tactic, a pattern like structure to encourage reuse. A Domain Tactic provides developers with information about the problem it addresses, the technical domain concerns that are influenced by using this tactic, the benefits, liabilities and an outline of a solution for realising that tactic.

• A meta-model for data-intensive mobile apps. We use a bottom up process for developing the meta-model to ensure that the concepts in the meta-model are actually used in existing systems. Tool builders can build upon this meta-model to improve the level of automation for mobile app developers.

Minor Contribution

• Toolchain. An implementation based on our meta-model that generates architectural scaffolding with a comprehensive evaluation of our approach. Developers use our tool to generate working data-intensive mobile apps that can be deployed to a device. Developers then build upon these generated apps using standard development tools and methods to add the final polish to the app. The toolchain includes:

  – RAPPT – an online tool for developers to describe and generate a working mobile app.
- A Domain Specific Textual Language (DSTL) used to describe the core features of a data-intensive app.
- Domain Specific Visual Language (DSVL) – A visual modelling language for describing high-level concepts for a mobile app. Based on our meta-model we created a visual language to model concepts such as navigation flow through a mobile app. This allows developers to quickly get started with building a new app as well as for rapid prototyping.

1.5 Thesis Organisation

The rest of this thesis is organised into 8 chapters outlined as follows:

Chapter 2 - Background Provides details about what is involved in mobile app development. Also included is an overview of the current literature focusing on Model Driven Development and Software Architecture.

Chapter 3 - Research Methodology. Describes our method for improving our understanding of how the implementation details influence software architecture.

Chapter 4 - The Technical Domain presents a method for systematically analysing and documenting domain specific architectural knowledge focusing on the domain of mobile apps.

Chapter 5 - A Method for Meta-model Construction. In this chapter we describe an iterative process for creating a meta-model based on the analysis of existing systems. We also include a meta-model for data-intensive apps.

Chapter 6 - Rapid APP Tool (RAPPT) describes the tool prototype RAPPT that realises mobile app architecture. This tool is based on principles derived from the analysis of the technical domain of mobile apps.

Chapter 7 - Evaluation and Discussion discusses our evaluation of
RAPPT involving 2 industry case studies, interviews with 10 practitioners, user acceptance with 20 developers and generating existing apps.

**Chapter 8 - Conclusions and Future Work** summarises the contributions, recommendations and describes key directions for future work.
Chapter 2

Background

To improve our understanding of how the implementation details influence software architecture, we need to first understand how software is developed. This requires separating the problem that developers have to solve from the solution that gets implemented. Once that is defined we can then explore how the developer goes about implementing a solution.

In the context of mobile apps, this involves identifying what is required to develop a mobile app and how an app is developed. When building a mobile app, developers have to consider the platform typically Android or iOS, understand the target users and their goals, and be knowledgeable about the implementation concerns such as hardware constraints and threading issues.

With this foundation we are then ready to consider how an appropriate software architecture can be designed. The relationship between an appropriate architecture and quality is well understood [1, 8, 29–31] so it follows that how the architecture is implemented will affect quality. As such, in this chapter, we review techniques for achieving an appropriate architecture from the perspective of a mobile app developer.

Defining an appropriate architecture is not sufficient, the architecture still needs to be implemented by a developer. As such we review the
state of the art in automation techniques focusing on the area of Model
Driven Development as a viable solution to the challenges facing mobile
app developers.

This chapter is divided into two sections, required background knowl-
edge and related work. The background section includes a discussion
on the different domains of software in Section 2.1 and presents the
case for considering the technical aspects of a system as a first class
citizen during the design phase in Section 2.2. In Section 2.3 we will
discuss the process of how a mobile app is constructed. In the related
work section we compare and contrast popular mobile app development
frameworks in Section 2.4. Following that we will provide an overview
of the related literature in Software Architecture (Section 2.5) and MDD
(Section 2.6). Finally, we conclude this chapter with a Summary in
Section 2.7.

2.1 The Domains of Software

The word domain is an ambiguous term in Software Engineering but is
commonly used to refer to the environment in which the software will
operate – the Application Problem Domain. The application problem do-
main is the field or area of expertise that needs to be studied to solve
a problem [7]. Software design best practices recommend a separation
between the application problem domain and the Technical Solution Do-
main [32] which describes the solution to the problem. Primarily, this
is to avoid arriving at a solution before considering other viable alter-
natives.

A Domain Model is a conceptual model that captures the essential el-
ements of a solution, the core entities, their relationships, attributes
and roles as well as the rules and constraints between them. Software
engineers often realise a solution by converting the knowledge in the
domain model into an implementation. This Implementation Domain
involves the tools and languages used to build the software. How these
domains relate to one another is shown in Figure 2.1 and we will dis-
cuss each aspect below in more detail.
Figure 2.1: Relationship between the Event Management and E-Banking application problem domains, with the technical solution and implementation domains.
2.1.1 The Application Problem Domain

The Application Problem Domain represents the domain in which the software will operate – typically this is the business environment. For example, online software for booking events are considered part of the event management domain where as an online bank account is part of the banking domain. Developers typically create a domain model to describe the core entities and relationships for that particular application problem domain – an event booking site would have domain entities for event, attendee etc. Both the Event Management and E-Banking domains are depicted in Figure 2.1 along with their related domain entities. Notice that application problem domains can share concepts such as account and payment. While these entities represent the same concept, their attributes and how they are used can vary based on the application problem domain. For example, an account for the Event Management App may have an attribute that represents a users event preferences where as a bank account is likely to have attributes concerned with transactions such as debit and credit amounts.

Developers often use the domain model of the application problem domain to further refine a solution such as defining an entity relationship diagram for realising a database schema. This refinement process moves closer to realising an implementation. The relationship between the application problem domain and technical solution domains is shown in Figure 2.1. In the following section we will discuss the Technical Domain.

2.1.2 The Technical Solution Domain

The shared technical domain in our example is that of the Mobile App technical domain, displayed in Figure 2.1 as the Mobile App Technical Domain. Building mobile apps presents a number of challenges (see Figure 2.2) specific to app development [27, 33–37]. These challenges are particularly relevant to mobile app development even though they may also be present in other technical domains. For example, mobile apps are 1) subject to being interrupted by phone calls and will need to
handle this event, 2) designed to be displayed on small screens, 3) used on the go without a stable network connection and 4) published through an app store which has a review process. All of these characteristics are intrinsic to mobile app development and need to be considered by developers.

Figure 2.2: Key challenges from the technical domain facing mobile app developers.

Referring back to Figure 2.1, notice that the application problem domains lead to the creation of two separate mobile apps that fulfil the respective functional and non-functional requirements. While two separate apps are created they both share the same technical domain irrespective of the technologies used to implement each app including the target platform. Each app domain leads to a separate implementation, the Event Management is an iOS app and the E-banking, an Android app. By analysing the technical domain we can capture knowledge that is independent of the application problem domain and of the implementation details.

The core aspect of a Technical Domain is a model that represents the
concepts that impact software developed in that sphere of influence. By sphere of influence we refer to software that share the same Technical Domain. The Technical Domain is explored in much the same way as traditional domain analysis is conducted for an application problem domain. This involves identifying the key entities that impact software, their relationships and discovering how they impact software. Below are the major characteristics of a Technical Domain.

- **Engineering Concerns:** Characteristics that influence the quality attributes of the software being developed in that technical domain are Engineering Concerns. These concerns are independent of the organisation developing the software and the application problem domain although they may change due to advances in technology, use of tools and frameworks. For example, building mobile web applications was challenging on older devices due to the poor support of standards by the web browsers. Smart phones with modern browser capabilities changed all this. The Engineering Concerns that are applicable to any given system are dependent on the requirements for that system i.e. not all mobile apps use location tracking features and thus, not susceptible to battery drain by using the GPS.

- **Independent of the Application Problem Domain:** Another facet of a technical domain is that of independence from the application problem domain i.e. the Engineering Concerns are relevant for an accounting and a sports app. This means that technical domain knowledge can be shared between application problem domains which encourages reuse.

- **Complexity:** In the Cambridge dictionary\(^1\) Complex is described as something that has a large number of inter-connected but related parts i.e. having many different components and relationships between these components. The Technical Domain is complex as there are many Engineering Concerns that are interrelated and cannot be analysed in isolation. For example, using sensors on a mobile device involves considering other Engineering Concerns.

\(^1\)http://dictionary.cambridge.org/dictionary/english/complex
such as power efficiency, hardware limitations and design patterns for using sensors – failing to consider all concerns can lead to a poor implementation i.e. poor usability, not backwards compatible etc. In addition, there are multiple connections between the software requirements and the Engineering Concerns i.e. hardware constraints, sensors and issues around concurrency for mobile may all be influenced by different requirements for the software. This is the aspect of the Technical Domain that is dependent upon developer experience [38]. For example, the concerns could be taught to an undergraduate student but an experienced developer is needed to translate these concerns into a high quality implementation due to the need to understand the relationships between concerns and make sound tradeoffs.

• **Implementation Independent:** A Technical Domain is a meta-level concept and does not include: platform, programming frameworks, tools and languages. For example, in mobile app development the technical domain presents the same issues (limited screen size, short battery life etc.) independent of the platform – the same issues arise whether developing Android, iOS or Windows Phone apps. However, the solutions and implementation strategies that platforms provide to overcome these issues may differ. Independence is an important characteristic as it allows the technical domain to be analysed separately from the implementation details. This allows the technical domain to be studied and knowledge documented for reuse across implementations.

**Defining a Technical Domain**

A technical domain needs to be well defined before it can be analysed so that software of the same technical domain can clearly be identified. This is a crucial step as concerns of the Technical Domain are ideally discovered by analysing existing systems – concepts are empirically derived from software similar to design patterns [5,39]. A Technical Domain is centred on the engineering concerns of building a system and is best understood from a professional developers point of view. Typically a domain expert is involved in defining what is included as part
of the domain [40]. In order to define the boundaries of a Technical
Domain we define a set of questions that need to be answered. These
questions are based on the different stages of software development in-
cluding, deployment, development and testing as well as external influ-
ences such as user behaviour, hardware requirements and integration
with other systems [28]. Each of the previous stages and influences
were brought to our attention after implementing the initial version of
RAPPT described in Chapter 6. Answers to these questions provide a
foundation for further exploration as they highlight how developing for
that technical domain differs from other domains. The following is the
list of questions to answer.

1. How is software deployed?
2. What are the usage patterns for the software in that space? (when,
   where and how is the software used)
3. What hardware capabilities are available to the developer?
4. What are the implications of the platform APIs and platform im-
   plementation decisions?
5. What are the external or 3rd party drivers for the software?
6. How is the software developed and tested?

In this thesis we elaborate and expand upon the technical solution do-
main of mobile app development. Chapter 4 builds upon the outline of
the technical solution domain as discussed here providing specifics for
the mobile app domain.

2.1.3 The Implementation Domain

After the solution has been designed for the application problem do-
main and the technical domain, an implementation is required. The
implementation domain is concerned with the programming languages,
frameworks and libraries that make up a software system. While the
high-level constraints have been addressed by the previous stages this stage still has unique constraints. For example, desired accuracy for a sensor reading may not be available due to the limited hardware.

The Implementation Domain involves developed applications. It also includes the environment where the solution is developed and deployed. In this domain, developers use a variety of tools and programming languages such as IDEs, build scripts, emulators, text editors and version control tools. Developers are tasked with using these tools and frameworks to realise a concrete implementation based upon the application problem domain and the technical solution domain. The output from this domain is software that can be run or used and may include, images, configuration files, videos, binaries and scripts.

The technical domain influences the implementation domain as it acts as a guide for how things should be done. For example, the technical domain specifies what the hardware limitations are such as battery life for a mobile phone. With a thorough understanding of the technical domain a developer will be aware of strategies for addressing these issues using the tools, libraries and languages of the implementation domain. The typically mobile app development journey is covered in Section 2.3.

2.2 Technical Domain as a First Class Citizen

Current software engineering practices provide general guidance for achieving quality such as through the use of Design Patterns [5], Quality Tactics [1] and automation through Model Driven Development [10]. However, these approaches do not have the specificity for addressing domain specific implementation concerns – the technical concerns are left as an implementation detail. For example, MDD generates an architecture for a mobile app but does not provide a rationale for the design decisions made i.e. mobile devices have limited resources so a sensor design pattern could be selected for its efficiency.

In this thesis we argue that the technical domain needs to be considered as a first class citizen when developing software. The primary reason for
Chapter 2. Background

this is that the technical domain provides the developer with additional information about the system they are building that otherwise has to be gained through experience. To illustrate this point consider Figure 2.3 which shows the relationship between stakeholders, requirements and the technical domain.

Stakeholders have requirements for the system, and some of these requirements are shared with stakeholders. For example, two different stakeholders may need the same report. Shared requirements are shown in Figure 2.3 by the intersection between the two stakeholders. Requirements are related and influence other requirements shown by the connected lines. Software developers are tasked with eliciting the requirements by engaging with stakeholders. However, the technical domain also influences the requirements of a stakeholder in a number of different ways such as:

**Conflicting Requirements.** A requirement can be impossible to achieve based on the constraints of the technical domain. An example of a conflicting requirement for a new app is that: it must constantly download content to display to the user. Mobile devices travel with the user and there are situations where the device has no network connectivity so this requirement cannot be satisfied. Trade-offs have to be made for requirements that conflict with the technical domain as the concerns are rigid and cannot be changed by the developer.

**Influence on Requirement.** Concerns from the domain may not prevent a requirement from being satisfied rather it may just influence the implementation. For example, when building mobile apps it is important to understand the impact the threading model has on the architecture. Failing to consider the threading model can lead to a poor user experience which impacts upon user ratings and ultimately the success of the app.

**Indirect Influence or Conflict.** The interconnected characteristic of the technical domain can make it difficult to reason about the influence it has on the requirements. By analysing the impact a concern has on a requirement it is possible to identify a relationship between two
requirements that, initially, do not appear to influence each other at all. For example, one stakeholder may desire that a mobile app track the users all the time which drains the battery and another stakeholder may need to use the app while they are in a remote location where the phone cannot be charged. These two requirements only conflict when considering the domain concern of Hardware Constraints and Sensing and understanding how these influence each other.

Understanding the technical domain can reveal additional information about the requirements for a system. In addition, it provides an insight into conflicts between requirements prior to the development process. Concerns in the technical domain are also interrelated and these relationships also need to be understood so that the proper impact they
assert on the requirements can be identified.

### 2.3 Typical Mobile App Development Journey

In this thesis we will focus our analysis on the technical domain of mobile app development. Before analysing the Technical Domain, it is paramount to have a thorough understanding of what is involved in building software in that sphere of influence. In this section we describe the typical app development process at a high level. We have included additional background information on data-intensive mobile apps and a more detailed description of what is required to build a mobile app in Appendix A.

Mobile app development follows a user-centred design process that relies on real-world human experience as the main driver for application design [41]. Typically, a professional designer iteratively develops screen mockups with the client to ensure that the final app matches expectations. This process begins with simple sketches either on a white board or on paper. From these early sketches full screen mockups are created that model aspects of the visual design of the app and the navigation flow. At this stage in the process designers may use a prototyping tool such as Fluid UI\(^2\) to refine the initial concept with the client.

By the time the architecture for a mobile app needs to be designed there are screen mockups of how the app will look, a basic outline of the functionality and some validation of the User Experience. The developer will then design the architecture of the app based on the desired functionality and UI mockups. With the design in place the developer will then create the first version of the app and take it to the client for feedback. This is an iterative process that will continue throughout the development lifecycle until all of the functionality has been added to the app.

On completion of the app the developer will prepare the app for submission to an app store which will require creating a build for production and signing the app for unique identification on the app store. Some

---

\(^2\)https://www.fluidui.com/
app stores such as Apple’s App Store require the app to be reviewed internally before being made public. This process can take a couple of weeks and may require tweaks to the final app so it is paramount that the developers follow the recommended best practices throughout the development process.

Unlike traditional software distribution channels mobile apps are distributed through app stores. Each of the major mobile platform vendors have an app store for distributing their apps: Google’s Android has the Play Store and Apple’s iOS has the App Store. Users can download apps from the app store and provide reviews for the apps. These user reviews and rankings directly impact the success of a mobile app as a poorly ranked app will not be featured by the platform vendor restricting discoverability and influence the decision of new users to purchase the app. Deploying apps to the app store is a simple process, developers upload their apps after filling out an online form which greatly reduces the barrier to distributing apps to end users. This characteristic has helped contribute to the number of apps available and users can easily download a competitor’s app. Thus, User Experience (UX) and User Interface (UI) design play a vital role in the construction of a mobile app.

Due to the central role that a user plays in the success of an app responding to the complaints and criticisms of an app in a timely manner is vital. Developers frequently integrate crash reporting tools and analytic frameworks to capture additional information about crashes and to provide insights into how the app is being used. Mobile apps have a number of monetisation strategies including in-app purchases, ads and paid apps which all influence how the app is designed and implemented. In addition to this there are a plethora of 3rd party libraries that developers use to integrate including libraries that interface with 3rd party hardware devices. Finding, selecting and evaluating 3rd party libraries is an integral part of the app development process.

---

3 https://play.google.com/store?hl=en
5 http://mobilestethoscope.com/
2.4 State of the Art Mobile App DSLs, Tools and Frameworks

Many frameworks and tools exist to aide in the development of mobile apps [42–49]. However, these approaches do not 1) provide a method for analysing the technical domain model, 2) do not consider or only partially consider the technical domain model and 3) provide no guidance on how to construct the underlying meta-model if present at all. The technical domain is also not considered as a first class citizen – the technical concerns are left as an implementation detail for the developer.

We considered both research prototypes and commercial products in our analysis. Almost exclusively, mobile app frameworks and tools are designed to address the problem of platform fragmentation – these techniques address the need to target multiple platforms. The following sections will discuss Mobile App Domain Specific Languages, Mobile App Generation Tools and the Mobile App Development Frameworks.

2.4.1 Mobile App Domain Specific Languages

In this subsection we discuss Mobile App Domain Specific Languages. Interestingly, the majority of Domain Specific Languages (DSL) were developed as research projects rather than commercial products. These DSLs typically lack details on how the code templates have been created to address the technical domain concerns and focus almost entirely on the entities that make up the language. A number of the main mobile app DSLs are shown in Table 2.1.

These approaches typically target developers who are already familiar with using text based programming languages and aim to provide a productivity improvement. Mdsd [50] focuses on generating key features of a data-intensive mobile app. However, they target developers who have little or no experience with the underlying mobile platforms and concentrate on generating prototypes. MD² [47] is another prototyp-
Table 2.1: Comparison of mobile app Domain Specific Languages. Key: - for unsupported features, y for supported features and (y) if partially supported. The grey highlighted area shows the gap in the current approaches.

<table>
<thead>
<tr>
<th>Platform</th>
<th>Meta-model</th>
<th>Meta-model Method</th>
<th>Technical Domain Model</th>
<th>Technical Domain Analysis</th>
<th>For Developers</th>
<th>Code Generation</th>
<th>Cross-platform</th>
</tr>
</thead>
<tbody>
<tr>
<td>MD² [47]</td>
<td>y</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>y</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>Mdsd [50]</td>
<td>y</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>y</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>MobiDSL [45]</td>
<td>y</td>
<td>(y)</td>
<td>-</td>
<td>-</td>
<td>y</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>Steiner et al. [42]</td>
<td>y</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>y</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>Mobl [51]</td>
<td>y</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>y</td>
<td>y</td>
<td>y</td>
</tr>
</tbody>
</table>

An approach that lets developers build upon the generated output but they do not address any of the technical domain concerns. However, MD² provides a backend generator for generating part of the service layer which is an important aspect to building new mobile apps.

Text based DSLs have also been used to create mobile web applications such as MobiDSL [45]. This approach is designed to remove the need to learn web development skills. Out of the approaches that we looked at this was the only one that presented a meta-model for the concepts used in the language. However, they constructed their meta-model by simplifying the work done on WebML [52] but do not provide a method for constructing a meta-model for other technical domains.

Mobl [51] is a DSL targeting mobile web apps by the authors of WebDSL [53]. It borrows many of the same principles of WebDSL. While this approach generates source code improving developer productivity it does not provide a model of the technical domain. Neither does it provide an insight into how the technical domain was analysed and embedded into the code templates.

Steiner et al. [42] take a different approach by proposing the use of
multiple DSLs for the different facets of mobile app development. They propose a DSL for the UI logic, Business Logic and Entity description as well as describing the data model of the app. However, this approach prohibits flexibility as the DSL hides the details of the general purpose tools and methods.

### 2.4.2 Mobile App Graphical Tools

The next category of mobile app development techniques that we looked at is that of graphical tools. Typically these approaches include a drag-and-drop interface for connecting components together to construct a mobile app. Many of the commercial tools and development frameworks (discussed in the next section) can broadly be categorised as either targeting hybrid web app frameworks (Cordova, Appery, AppGyver) or focus on native apps (Xamarin, Appcelerator). These approaches require the developer to address the technical domain concerns manually, provide no automation and need to be developed in the same way a developer would build an app if they were using the base platform’s APIs. A comparison of the Mobile App Generation tools are shown in Table 2.2.

MobiA [46] is a tool that does away with code focusing on a graphical modelling language for specifying mobile apps. The focus is on the single domain of health monitoring applications rather than on being a general tool for data-intensive mobile app generation [54]. Emphasis of this work is to ensure that the non-technical user can easily build mobile apps rather than on addressing the technical concerns of app development.

App Inventor [56] and Nitrogen [43] focus on end user developers and were designed to simplify the development of mobile apps. By targeting non-technical users, these approaches had to hide significant implementation details that restricted the types of apps that could be generated. While these tools had a meta-model for the concepts that were exposed to developers neither approach focused on the conceptual concerns of the technical domain. As a consequence these approaches hide the technical domain concerns from the end users which prevents de-
### Table 2.2: Comparison of mobile app graphical tools. Key: - for unsupported features, y for supported features and (y) if partially supported. The grey highlighted area shows the gap in the current approaches.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>WebRatio [55]</td>
<td>y</td>
<td>-</td>
<td>-</td>
<td>(y)</td>
<td>(y)</td>
<td>y</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>Nitrogen [43]</td>
<td>y</td>
<td>-</td>
<td>-</td>
<td>(y)</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MobiA [46]</td>
<td>(y)</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>y</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>App Inventor [56]</td>
<td>y</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>AppGyver</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>(y)</td>
<td>y</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Appery.io</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>(y)</td>
<td>y</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Developers from making certain design decisions.

Nitrogen [43] is a codeless and cloud based mobile development platform aimed at enterprise employees who do not have a technical background. Nitrogen makes use of models but how the models were created is not explicitly stated. There is also no consideration as to how the technical domain concerns will be overcome and there is limited flexibility due to the codeless environment – not all of the concepts of a mobile app are made available to the user.

Out of the tools that we found, WebRatio [55] was the only commercial tool that explicitly took a model based approach to realising mobile apps based on an extension of the Object Management Group standard for the Interaction Flow Modeling Language [57]. This approach generated cross platform hybrid apps using the Cordova framework but does not consider the implementation concerns when generating source code.

App Inventor [56] is a tool designed to help children build mobile apps using based on a visual programming language. Primarily this tool is

---

6 https://cordova.apache.org/
used as a teaching tool and does not provide the needs of a professional developer namely, flexibility, code generation or address the technical domain concerns.

AppGyver\(^7\) and Appery.io\(^8\) offer an online cloud platform for developing mobile apps. These tools simplify development and do not have an underlying meta-model or approach for addressing the technical domain concerns.

2.4.3 Mobile App Development Frameworks

In this section we discuss Mobile App Development Frameworks and a number of related approaches to building mobile apps. We focused on four main frameworks, Appcelerator\(^9\), Cordova\(^10\), Ionic\(^11\) and Xamarin\(^12\). Appcelerator uses a model based tool but does not specify what the underlying meta-model is. This method uses Javascript for building mobile apps that run on multiple platforms. Appcelerator does not include abstractions for dealing with the technical domain concerns. Xamarin is a framework that lets developers build apps for any platform using C#. This approach wraps the underlying mobile platform API to ensure developers can build any functionality that they desire. Cordova is a hybrid development framework that hides the platform specific details but does not provide guidance or abstractions for dealing with the technical domain concerns. Ionic is built on top of Cordova and includes core UI components for building hybrid mobile apps that look like native apps and there are a number of 3rd party generators for generating parts of the mobile app\(^13\). For a comparison of the four main Mobile App Development Frameworks see Table 2.3.

There are many other frameworks for building mobile apps other than the ones we have covered here. We have compiled a list of more than 70

\(^7\)http://www.appgyver.com/enterprise/
\(^8\)https://appery.io/
\(^9\)http://www.appcelerator.com/product/
\(^10\)https://cordova.apache.org/
\(^11\)https://ionicframework.com/
\(^12\)https://xamarin.com/
\(^13\)https://github.com/diegonetto/generator-ionic
Table 2.3: Comparison of app development frameworks. Key: - for unsupported features, y for supported features and (y) if partially supported. The grey highlighted area shows the gap in the current approaches.

<table>
<thead>
<tr>
<th>Platform</th>
<th>Meta-model</th>
<th>Meta-model Method</th>
<th>Technical Domain Model</th>
<th>Technical Domain Analysis</th>
<th>Flexibility</th>
<th>For Developers</th>
<th>Code Generation</th>
<th>Cross-platform</th>
</tr>
</thead>
<tbody>
<tr>
<td>Appcelerator</td>
<td>y</td>
<td></td>
<td></td>
<td></td>
<td>y</td>
<td>y</td>
<td>-</td>
<td>y</td>
</tr>
<tr>
<td>Cordova</td>
<td>-</td>
<td>-</td>
<td>y</td>
<td>y</td>
<td>-</td>
<td>y</td>
<td>y</td>
<td>-</td>
</tr>
<tr>
<td>Xamarin</td>
<td>-</td>
<td>-</td>
<td>y</td>
<td>y (y)</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>Ionic</td>
<td>-</td>
<td>-</td>
<td>y</td>
<td>y (y)</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
</tr>
</tbody>
</table>

Other mobile app prototyping tools and frameworks available online\textsuperscript{14} and replicated in Appendix I.

Other approaches to building mobile apps include Cloud Twin which deals with the heterogeneity of mobile platforms by enabling the possibility of running apps from one platform natively on another platform [58]. It works by having a version of the app running on an the original platform in the cloud.

There has also been work done on the reverse engineering of mobile apps for testing [59, 60] and for constructing UI artefacts from an image [61]. These approaches can augment our work by improving the generation of a UI as our approach generates an entire app rather than just focusing on the UI.

Recent work has looked at improving mobile app development through a recommendation tool for API usage [62]. This approach helps developers build solutions but does not provide them with background information as to why these API calls should be used together. By combining this approach with the technical domain tactics we can provide developers with more informed recommendations explaining why this

\textsuperscript{14}https://github.com/ScottyB/rapt-resources
API call was recommended.

The analysis of the mobile app frameworks revealed two major gaps. First, the majority of approaches did not include a meta-model of the underlying concepts used to build mobile apps – either the meta-model was lacking or they replicated the mobile platforms APIs. Second, when a meta-model was specified there was no repeatable method for reconstructing a meta-model and there was no way to reuse this approach for other domains. To address this problem we present our first research question How do we empirically build a meta-model that contains the core abstractions a developer uses to build mobile apps? Basing our method on existing apps is essential to ensure that we capture concepts that developers use rather than what we think they should use. Answering this question will not only provide us with a meta-model for building apps but also provide us with a method that can be applied to other technical domains.

2.5 Software Architecture

Quality attributes assert one of the strongest influences upon the design of software architecture and are frequently the cause for a redesign or rewrite of existing systems [1]. One of the challenges of designing a system to meet desired quality attributes is that they often conflict i.e. performance is at odds with achieving maintainability [63]. As software architecture is known to assist in achieving these desired quality attributes [1], architects need tools and approaches to reason about competing quality attributes to make informed tradeoffs during the design phase. In addition, architecture remains fixed in the face of software evolution and requires a significant amount of effort to change later on in the development process [64]. In short, architects need to understand how quality attributes will be influenced by their design decisions.

Research shows that documenting design decisions during the early phases of development is an important step in software architecture design [38,65,66] that helps share understanding between stakeholders,
and improves maintenance and evolution of the architectural design. Despite the benefits of documenting design decisions that standard, ISO 42010 Systems and Software Engineering–Architecture Description [9], contains no means to capture technical domain specific information – no concept that represents the Technical Domain. To illustrate this point consider the definitions from the the closest related concepts from the standard, Concern and Environment:

- **Concern** – “interest in a system relevant to one or more of its stakeholders. **Note:** A concern pertains to any influence on a system in its environment, including developmental, technological, business, operational, organizational, political, economic, legal, regulatory, ecological and social influences.”

- **Environment** – “context determining the setting and circumstances of all influences upon a system. **Note:** The environment of a system includes developmental, technological, business, operational, organizational, political, economic, legal, regulatory, ecological and social influences.”

Both these descriptions are very broad and encompass many different concepts and lack the specifics required for dealing with mobile concerns. For example, neither capture the idea that network availability is a concern when developing apps on ubiquitous mobile devices.

Another important aspect of the description is the concept of a Concern is its association with stakeholders. In this thesis we take the definition of stakeholder as – “A stakeholder in the architecture of a system is an individual, team, organization, or classes thereof, having an interest in the realization of the system” [8, 9]. Stakeholders have a different set of concerns some of which contradict and compete with each other as discussed in Section 2.2 and it is the role of the architect to make the necessary tradeoffs. The key here is that a stakeholder is an entity that can be approached where as the technical concerns are based on the architects experience. In addition, architects tend to make design decisions on their own and desire to have support for accessing architectural knowledge [67]. This means that knowledge captured by studying
the Technical Domain cannot easily be discovered through stakeholder engagement and is not well documented.

### 2.5.1 Quality Tactics Lack Specificity

Quality tactics are considered the basic building blocks of architecture design [1] and are used to achieve a quality attribute requirement [68]. Architects select suitable tactics during the design phase of the system and then leave it up to the developers to interpret how the tactic is implemented. Quality tactics, as the primitives of software architecture design, are general principles that are applicable to many different systems. As a consequence, quality tactics do not contain technical domain specific variations neither do they provide any information on how they should be implemented. However, research shows that the domain impacts upon software quality [69–71]. This has been found to be of particular importance when developing mobile apps [72–74] where mobile specific Engineering Concerns such as the use of sensors, leads to the creation of new usage patterns – the technical domain influences how software is constructed. What is needed is an approach that captures known information that is specific to a technical domain so that architectural knowledge can be reused. This information will also provide implementation details that developers can use to help achieve desired quality attributes.

### 2.5.2 Design Patterns are Too Low Level

Design Patterns [5, 75–83] is a general reusable solution to a reoccurring problem in a given context. As with quality tactics, design patterns can help improve the quality of the system [84]. Design Patterns have also been used to capture business domain knowledge and there are emerging design patterns for developing mobile apps [85–90]. Work to date focuses on a specific facet of mobile app development such as information systems [88], UI design patterns [87] or low level Object Oriented design patterns [85, 86, 90] rather than take a holistic view of the technical domain. However, there are no methods to clearly define
the technical domain so that all facets of an implementation can be analysed to identify technical domain specific patterns.

Many researchers have proposed methods to automatically detect the use of design patterns [91–101]. These approaches focus on locating already understood and documented design patterns. By using such methods architects can reason about the structure of the software and quickly grasp the design of the system. However, these methods cannot be readily adapted for locating and finding new domain specific technical patterns as they are used to locate known design patterns rather than discover new patterns. What is needed is a method that can be used to identify potential patterns by analysing software of a specific technical domain. In addition, a method is required to clearly identify the technical domain to define the search space for identifying technical domain patterns.

2.5.3 Technical Domains Lack an Architectural Viewpoint

In the ISO 42010 standard [9] Stakeholders have Concerns that are shown by an Architecture Viewpoint [102]. Through using a viewpoint of a system architects can view architecture from a particular point of view which enable them to reason about the constraints on the system and communicate with stakeholders. Viewpoint Frameworks are used by architects to document their architectures [103, 104]. Researchers propose a Common Architectural View Model which is intended to be a meta-model for View Models [105] – a Viewpoint Framework consists of views which are of a type or sub-type of the views captured in the Common Architectural View Model. This View Model is summarised in Table 2.4. The mobile app specific implementation concerns will be documented in different Viewpoints making them more difficult to reason about. As we will in Chapter 4 these concerns need to be represented as first class citizens and viewed together as they are interrelated.

What is needed is a concept to model the technical concerns of develop-
Table 2.4: Common Architectural View Model.

<table>
<thead>
<tr>
<th>Viewpoints</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements</td>
<td>Documentation of the architecture requirements</td>
</tr>
<tr>
<td>Logical</td>
<td>Documentation of the architecture design</td>
</tr>
<tr>
<td>Data</td>
<td>Documentation of aspects with regard to saving, manipulating, managing, and distributing data</td>
</tr>
<tr>
<td>Implementation</td>
<td>Documentation of the implementation structure and the implementation infrastructure</td>
</tr>
<tr>
<td>Process</td>
<td>Documentation of the control and coordination of concurrent building blocks</td>
</tr>
<tr>
<td>Deployment</td>
<td>Documentation of the physical deployment of software building blocks</td>
</tr>
</tbody>
</table>
Chapter 2. Background

Developing a data-intensive mobile app and the impact they have on non-functional requirements. Tang et al. provide a classification for architectural elements which has the concept of a Technology viewpoint [106]. The Zachman Framework [103] also has a Technology view defined as: “A view concerned with the technological implementation of a system” Neither approach goes into the specifics of a technology viewpoint of a system that architects can use to guide an implementation. While researchers recognise that a System Context viewpoint is needed to identify external entities [107] the intrinsic quality requirements that they may place on a system are not detailed. Although apps are not considered external entities the inherent limitations and attributes of the mobile platform influences software architecture. These viewpoints lack the information required for developers to build quality mobile apps leaving the technical concerns as an implementation detail.

2.5.4 Reference Architectures lack Technical Domain Concerns

A software Reference Architecture [108] describes the essential components of a system architecture [109]. Reference architectures that focus on a specific domain are known as Domain-specific software architectures [108, 110, 111] and focus on a very narrow domain to increase the descriptive power of the structures described [6]. These reference architectures ensure interoperability between the components through standardisation and provide many benefits such as reduced development costs, improved communication between stakeholders, reduced time to market and reduced risk [22]. Reference architectures also enable the assembly of new systems quickly by selecting and configuring the components for the new system. However, reference architectures are costly to setup and have a steep learning curve [21].

While there are recommendations for building reference architectures on empirical evidence [109], there is no concept of technical domain concerns, how to identify this information neither is there a way to document this information. Without considering the technical domain concerns as a first class citizen the wrong reference architectures may
be constructed. Understanding the technical domain concerns and the trade-offs is also required to provide developers with an appropriate interface for building a tool that assembles software architecture.

Software architecture literature lacks a holistic approach to addressing the specific needs of mobile app development. To overcome this limitation we propose the following research question *What are the concerns of the technical domain that influence mobile app architecture and how can this information be captured?* By analysing the technical aspects we can identify domain specific architectural knowledge which can benefit developers who build mobile apps. Once this information has been discovered we need a way that developers can use this information. Answering this research question provides a foundation for reasoning about domain specific design decisions that influences software architecture.

### 2.6 Model Driven Development

Advances in MDD enable the synthesis of an app through a model transformation process concluding with a code generation step [7]. MDD is heralded as improving productivity [112–114] and quality [10] as well as reducing risk and providing tools for formal analysis [115]. MDD is fundamentally based on the same principles that have proven successful in software engineering - raising the abstraction level and increasing the level of automation. However, these approaches lack a way to analyse the implications of the technical domain as the implementation details are hidden from the end user. Implementation decisions for Android development such as which states of the app lifecycle control the sensors are not present in high level models – when to control enable/disable a sensor impacts on performance. Only by analysing the technical concerns around hardware limitations, sensors and the app lifecycle does a developer know these impact each other and that trade-offs need to be made.
2.6.1 Model Driven Architectures

The Object Management Group\(^{16}\) has set up a key initiative of MDD, that of Model Driven Architecture (MDA) \([10,20,116,117]\). MDA is a set of guidelines and tools based on the Object Management Group standards that enable the modelling of both the problem and the solution to increase the reliance on automation \([20]\). At the core of MDA is a hierarchy of models: 1) Computation Independent Model which captures the business requirements, 2) Platform Independent Model use for capturing parts of the system architecture and 3) Platform Specific Model used for capturing platform details \([11]\), and model transformations that enable one model to be transformed into another. This approach allows the details of the target platform to be captured at the model level, the Platform Specific Model. Another advantage of MDA is that there are two way transformations between models enabling developers to view the system at different levels of abstraction.

2.6.2 MDA and the Technical Domain

MDA does not consider the technical domain as a first class citizen which is manifested in three ways. First, MDA focuses on the separation of the business functionality from the technical implementation. This means that while the business requirements for a system can easily be expressed using MDA, developers are encouraged to ignore the technical implications. However, the technical domain concerns may influence a business requirement as shown in Section 2.2 and need to be considered at the same time. Second, MDA provides specifications for describing models and model transformations but does not provide any guidance on how this information is used to implement an appropriate architecture – this information is left as an implementation detail of the transformations between a Platform Specific Model and code. Finally, developers must create suitable code templates for the target platform and need to ensure that the code generated addresses the technical concerns. How this is done is not specified in the MDA standard neither are guidelines for identifying the core concerns of the

\(^{16}\)\url{http://www.omg.org/}
technical domain.

Currently the tool builders of MDA software need to thoroughly understand the technical domain to ensure that all of the constructs can be used to address these problems. For example, a Platform Independent Models indicates that the mobile app being built needs to use the location sensor but it is not clear as to which other concerns are related to using the sensor – Developers still need to use the modelling constructs to address the technical concerns. By having a technical domain model the developer can immediately see that this influences the app architecture. With a method for analysing the technical domain developers can ensure that MDA based tools support the creation of solutions that address the technical concerns.

2.6.3 Platform Independent Models are Insufficient

The Platform Independent Models is the first model that is understandable by software developers and contains information that is suitable for a number of different platforms. What is not included in a Platform Independent Models is information about how the technical domain concerns are going to be addressed and these concerns are the same for each target platform. Consider a Pizza app that enables customers to follow the driver as they deliver the order to their house. To achieve this functionality the delivery driver’s location needs to be constantly updated to provide real time information. The location information can be updated every so often or batch uploaded and each option has a number of different drawbacks and benefits. Dealing with the sensor data for deliver driver location and handling the network issues around providing real-time data are technical domain concerns. These concerns are the same whether developing an Android or an iOS app. This information may be present in the Platform Independent Models but is left up to the software developers experience to be included.

The Platform Specific Model needs to include concepts for handling the uploading of location data in background services that run even when the driver is not using the app. How this functionality is modelled and
then implemented is independent of the business domain yet needs to be reinvented for each MDA based approach for mobile app development. In MDA the technical domain concerns are not made first class citizens, rather they have to be discovered through the requirements elicitation process.

### 2.6.4 Software Product Lines

Software Product Lines (SPL) [13–16, 118–126] is a development approach that focuses on the reuse of pre-built software artifacts for the development of new software [127]. SPLs do not provide the generality to be applicable to an entire technical domain rather these approaches are associated to a specific class of software. For example, SPLs have found great success in the automotive industry where variations of the same software is needed to run in each type of car [128]. In this regard SPLs are focused on the specific business domain and do not treat the technical domain as a first class citizen.

There are two processes in a SPL, Domain Engineering which focuses on the identification and creation of usable software assets and Application Engineering that focuses on the construction of the final product using assets created during the Domain Engineering phase [129]. The result of a SPL is a Product Family which consists of a set of apps with similar features [130]. Domain Analysis is a fundamental component of the Domain Engineering lifecycle [129]. This stage is designed to identify common features needed for the product family being developed and does not contain generally applicable concerns to all apps developed in that technical domain. Different product families can share the same technical domain but may have entirely different features. SPL have also been used to create mobile apps [131, 131–134]. These approaches focus on identifying features [135–139] during the domain analysis stage rather than taking a holistic view of the domain. The underlying concerns of the technical domain are not presented as first class citizens and are not reasoned about during SPL Engineering.

Due to the constant changing environments and need for a high de-
gree of adaptability from the software researchers have created Dynamic SPLs [140–143]. A Dynamic SPL produces software that is adaptable to the needs of the user and changes in its environment by addressing variability at runtime. The dynamic variability in Dynamic SPLs help adapt to the changes in the technical domain but this area of research does not have any method for modelling the concepts that cause variability.

Related to the field of SPL is the area of Software Factories [144, 145]. This approach draws parallels with traditional manufacturing processes where software is assembled from pre-made parts. Using this approach developers can quickly build new systems building upon the experienced gained building similar products. While this encourages reuse the fundamental concepts that influence the software are not captured in a reusable manner and are only useful to products from that factory i.e. a e-commerce factory only produces e-commerce software.

In summary, SPLs provide a way to produce similar software products reducing cost and improving quality but lack the generality required for modelling the technical domain – many different product lines share the same technical domain.

One of the easiest ways developers can benefit from insights gained from analysing the technical domain is through automation. To this end we present our third and final research question How can the technical domain influence the design and implementation of tools for automation?. This research question seeks to build upon our understanding of the concepts used to build mobile apps and the architectural influence of the technical domain. We consider how the findings from the previous two research questions can be combined to create tools that improve software architecture. Answering this research question will also show the practical implications of understanding the technical domain and provide insights for the development of new tools.
2.7 Summary

We presented the Application, Technical and Implementation domains and highlighted the impact the technical domain has on architectural design decisions. As the technical domain influences architecture we argue that it needs to be considered as a first class citizen when developing software and thus its value in the context of improving mobile app quality. We then described the development journey followed when building a mobile app.

Next we presented current mobile development frameworks, both commercial and research based. These approaches 1) lack a method for building a meta-model and 2) do not treat technical domain concerns as first class citizens and hence they are not adequately considered in either the meta-model, or at the model transformation stage. We then discussed software architecture in the context of achieving quality. These approaches either lack the specificity required for addressing the technical domain concerns of mobile apps or are too low level to reason about the decisions during the design phase.

Finally, we discussed MDD for realising software architecture focusing on Model Driven Architectures. The Model Driven Architectures approach treats the technical domain concerns as an implementation detail and fails to demonstrate how the generated architecture will satisfy the mobile app concerns e.g. what tradeoffs were made when the device is unable to access the network. We also discussed how Software Product Lines are used to generate software systems in a narrow business domain and lacked generality.

With this background information we can see that the technical implementation domain is not well understood as an isolated concept. In addition, the current techniques for creating and realising a software architecture do not take the technical implementation domain into consideration.

From our discussion of the technical domain and a review of the related literature we have derived the following research questions that will be
addressed in the following chapters:

- **RQ1.** What are the concerns of the technical domain that influence mobile app architecture and how can this information be captured?

- **RQ2.** How do we empirically build a meta-model that contains the core abstractions a developer uses to build mobile apps?

- **RQ3.** How can the technical domain influence the design and implementation of tools for automation?
Chapter 3

Research Methodology

3.1 Introduction

In this chapter we describe the research method used in this thesis. Our research method centres around the empirical analysis of mobile apps, engagement with domain experts for iterative feedback via surveys and interviews, and a tool for evaluation and refinement of our model. We also conducted industry case studies and a user evaluation using professional developers.

We chose to focus our method on empirical analysis and rapid prototyping for three main reasons. 1) Empirical analysis of software artefacts enabled us to quickly determine what was actually done by software engineers. Contrast this approach with a survey which involves the developer describing what they think they do. 2) Using prototypes enabled us to test out our ideas with developers in a way that they were familiar with and ensures that our research could be immediately applied to real world solutions. 3) By building the tools that use our concepts we can gain hands on experience with the concepts and refine our ideas based on personal practical experience.

Our method is inspired by the work done on WebDSL [53] and informed by the guidelines for building a DSL [146]. The full details of the are discussed in Chapter 4 and Chapter 5. The main change that we added
is two domain analysis phases that capture the architectural domain knowledge in a technical domain model that is used to perform another domain analysis phase.

Our research method is divided into three stages one for each of our research questions as shown in Figure 3.1. The details for each stage including the description of the process, outcomes and evaluation are covered below.

**Figure 3.1:** Research method used to address our goal to improve our understanding of how the implementation details influence software architecture. Coloured boxes indicate the outcome for each stage, Stage 1 Blue and Red, Stage 2 Green.
3.2 Method

Each stage is addressed by a proceeding chapter which covers the Research Input, Process and Outcome components. The evaluations for each topic chapter are included in Chapter 7.

3.2.1 Stage 1. Capturing Technical Domain Concerns

Stage 1. was designed to address our first research question, RQ1. What are the concerns of the technical domain that influence architecture and how can this information be captured? By addressing this question we wanted to identify concerns for a specific domain that influenced software architecture but were typically left to the developer to address. We also wanted to come up with a way of extracting and documenting this domain specific architectural knowledge. Documenting this information provides developers with a way to apply our techniques to other domains which encourages reuse by capturing tried and tested architectural knowledge. Chapter 4 describes in detail how we answered RQ1.

Answering RQ1 in the context of mobile app development involved the empirical analysis of apps, documentation and related literature as shown (left-right from RQ1) in Figure 3.1. From this domain analysis stage we extracted a technical domain model that includes the key concerns that influence mobile app architecture. This technical domain model was used to analyse additional mobile apps to extract solutions to the concerns raised by the model during the Informed Domain Analysis stage. The outcome from this second analysis stage was a set of domain tactics which describe solutions to the concerns presented in the technical domain model.

Evaluating the technical domain model involved case studies covering both open source and commercial apps. These case studies were designed to show how the technical domain model was used to analyse software architecture. We also conducted interviews with software engineers to evaluate our domain tactics and the technical domain model.
The technical domain model was evaluated for completeness, usefulness to novice developers and relevance in the mobile app domain. By evaluating the domain tactics with practitioners we can assess whether we have identified architectural knowledge that is worth knowing before building a mobile app. The interviews also assessed if the problems in the technical domain model represented common problems facing practitioners.

Our focus on RQ1 was to identify the key elements that influence architecture from a developers point of view. To achieve this, we analysed existing systems which, in the mobile domain, are readily available thanks to the plethora of apps in the app stores. To ensure that we extracted concerns that developers address in a commercial setting we conducted industry case studies and a user evaluation.

3.2.2 Stage 2. Modelling Domain Specific Abstractions

Next we analysed the technical domain from the developers perspective seeking to identify the abstractions used to address the technical domain concerns. Specifically we sought to answer RQ2. How do we empirically build a meta-model that contains the core abstractions a developer uses to build mobile apps? Creating a meta-model forms the foundation for tool automation and increases our understanding of the core concepts required for developing a mobile app. The details of how we addressed RQ2 can be found in Chapter 5.

To answer RQ2, we analysed data-intensive mobile apps from the users perspective specifically the user interaction. Data-intensive mobile apps were selected as they 1) contain common elements provided by the underlying platform rather than highly customised apps such as games and 2) many clients require these types of mobile apps [50].

The focus was on identifying the key concepts of a mobile app from the user’s perspective and the concepts a developer uses to build the interactions. The primary outcome from this stage is a meta-model that contains a set of concepts developers can use to model mobile apps
and capture the user interaction model. We followed an iterative process when analysing apps and one of the stages required implementing a tool based on the evolving meta-model. This ensured that the meta-model could be used for existing systems and captured the core concepts required for building an app.

The results from this stage were evaluated through the construction of a tool implementation. By building a tool we can generate new apps to assess if the attributes in our meta-model are sufficient for modelling other apps. We used our tool to generate the architecture scaffolding for a set of data-intensive apps and found no new concepts that could not be modelled by our meta-model. Building a tool also meant we could give it to developers and determine if the concepts identified map to the concepts used by practitioners to build apps. We conducted a Concept User Evaluation with software engineers and found that the concepts we had identified mapped to a developers mental model.

Our meta-model is intended to capture the concepts that are used by developers to build software. To achieve this we analysed existing mobile apps to capture the concepts that had been used by developers. Our evaluation was based on a tool implementation to ensure that practitioners could build new apps using the vocabulary that they were familiar with. We rebuilt existing mobile apps to ensure that our model could 1) be used to model real world examples and 2) model all of the common elements of an app.

### 3.2.3 Stage 3. Automation of the Technical Domain

Finally, we developed a tool for professional developers to address the concerns of the technical domain to answer RQ3. *How does the technical domain influence the design and implementation of tools for automation?* Our intention was to adapt contemporary practices such as those proposed by Model Driven Development to address the technical domain concerns as a first class citizen. By doing this we wanted to improve the state of automation for building new data-intensive mobile apps. Chapter 6 describes our approach to RQ3 and explains how our
tool prototype works.

To answer *RQ3* we built a tool prototype RAPPT as shown in Figure 3.1. RAPPT was based on the output from the previous stages including the meta-model for data-intensive mobile apps, the technical domain model and domain tactics for addressing domain specific concerns. RAPPT was designed to generate the architecture scaffolding for a new app. Developers use RAPPT to generate the initial version of the app and then write low level platform code to finish off the app.

RAPPT is designed to be used by practitioners so our evaluation involved an User Evaluation. This was primarily designed to gauge the acceptance of using RAPPT in a commercial environment to build the initial version of an app. This evaluation consisted of developers building a small app using RAPPT and answering a questionnaire. We also wanted to see how the affects of software evolution would influence the code generated by RAPPT. To do this we conducted a case study on a commercially developed app and analysed the state of the generated architecture after 2 years of on-going development. From this study we conclude that developers had a tendency to copy the generated code to other aspects of the project and additional studies are needed to explore how generated code is used.

A strong motivation for our work was the need to help architects improve the quality of the software they design and then build. We built a tool based on our theoretical findings to ensure that developers could leverage the benefits as soon as the research was done. This focus on helping practitioners also influenced our evaluation to include real end users to lower the bar for the uptake of our findings.
Chapter 4

The Technical Domain

The technical domain captures the domain specific concerns that influence the non-functional requirements of a system. To improve our understanding of how the implementation details influence software architecture we analyse these concerns in detail. The impact non-functional requirements have on software architecture is well understood [1, 63, 147–154]. Software architecture literature suggests that non-functional requirements are achieved by analysing the different viewpoints [102], leveraging reference architectures [108], and implementing design patterns [5, 39] and quality tactics [1].

Current literature tends to view architecture primarily through the lens of functional and non-functional requirements. The concerns of the technical domain are left to the intuition and experience of an architect – addressing these concerns is more of an art form than an engineering discipline. Neither, is the influence the technical domain has on non-functional requirements considered. In addition, these approaches do not provide a holistic view specific to mobile app development.

In this chapter we address this issue by answering the question What are the concerns of the technical domain that influence mobile app architecture and how can this information be captured? By understanding these concerns we can develop mobile app specific guidelines, processes and tools that improve the quality of software architecture. This chapter
focuses on analysing the technical domain of mobile app development.

Capturing information about the technical domain also provides a means for developers to analyse their design decisions before commencing development. In addition, documenting the technical domain encourages the reuse of specific patterns. In this way a technical domain model can be used as a way of transferring knowledge about the technical concerns from an experienced developer to novice developers.

To analyse the technical domain of mobile apps we look at what concerns influence mobile app architecture, how these concerns influence architecture i.e. which quality attributes do they impact, and when do these concerns come into affect i.e. under which conditions do these concerns need to be considered. We also describe how technical domain specific architectural knowledge can be mined and documented for reuse.

This chapter is organised as follows: Section 4.2 presents the process used for mining and documenting domain specific architectural knowledge followed by Section 4.3 which describes how the technical domain concerns influence software architecture and how this information can be documented. Then we discuss the results from mining domain knowledge from mobile apps that includes, a set of Problem Context, a set of Domain Tactics and Domain Concerns in Section 4.4. The chapter concludes with a wrap up in Section 6.5.

4.1 Domain Tactics, Architectural Tactics and Patterns

In this chapter we present the new concepts of Domain Tactics and technical domain concerns. These contributions overlap with a number of different research areas including software product families, requirements engineering and software architecture.

A core aspect of building Software Product Families is domain engineering [155] which focuses on reusing elements from the domain. The
technical domain model and the domain tactics capture elements of the
domain but this is not done as part of the software development phase.
In addition, the technical domain model and the domain tactics are ap-
picable to multiple product families so they can be used to inform the
domain analysis, domain design and domain implementation steps of
domain engineering [155].

Requirements Engineering is the first step in a software development
project, focusing on documenting and analysing non-functional and
functional requirements [156]. Domain analysis plays a vital part in
the requirements gathering process which can be greatly improved by
understanding the technical domain model. The core concerns of the
domain are part of the technical domain model which will inform devel-
opers of the non-functional requirements that influence their software.

We have decided to present Domain Tactics from the software architec-
ture perspective. The main reason for this is that Software Engineering
literature indicates that quality is achieved through an appropriate ar-
chitecture [1,8,29–31] and Domain Tactics are associated with achiev-
ing quality attributes. Another reason for using the architecture view-
point is that we feel this provides developers with the greatest benefit
from this research.

Architectural Tactics are described as “a means of satisfying a quality-
attribute-response measure by manipulating some aspect of a quality
attribute model through architectural design decisions” [157]. Domain
Tactics differ in that they are not directly related to a quality-attribute-
response measure and they are focused on a specific technical domain.
However, Domain Tactics have both positive and negative influence on
quality attributes. We have chosen the name Domain Tactics to connect
with people’s familiarity with architectural tactics and the association
they have with quality attributes. The Domain part of the term captures
the concept of a sphere of influence over multiple systems.

Domain Tactics can also be described as a pattern using the following
description “A pattern is the abstraction from a concrete form which
keeps recurring in specific non-arbitrary contexts.” [158]. This descrip-
tion is rather broad and does not convey that domain tactics influence quality attributes. There are also many different kinds of patterns for different purposes and we wanted to ensure that domain tactics could be identified as related to software architecture. Our template for documenting domain tactics has been adapted from a pattern template described in Section 4.3.

4.2 A New Method for Mining Technical Domain Knowledge

Technical domain knowledge is discovered rather than created just like Design Patterns [5] are well tried and tested solutions. This is to ensure that the extracted domain knowledge represents well tried and tested solutions present in the domain. As such our method focuses on the study of example systems from the specific technical domain. We need a systematic approach for identifying and validating domain knowledge from existing resources. Our approach is a specification of the domain analysis process [159, 160]. Common approaches to domain analysis rely on feature models [161], where as our approach models the technical domain and we analyse the user interaction model, source code implementation in addition to the features (See Figure 4.1).

Our method is divided into two separate processes. The first models the domain producing a set of domain concerns, quality attributes, problem contexts and a collection of solutions to the problem contexts. The second process validates and refines the technical domain knowledge into structured reusable content known as Domain Tactics. How these two processes are related and the steps required for each process is shown in Figure 4.1.

4.2.1 Defining the Mobile App Technical Domain

Developing mobile apps differs significantly from developing for desktop computers or other software systems. There are at least three charac-
characteristics of mobile devices that separate them from other software systems: 1) they are rarely turned off, 2) they often travel with the user and 3) there are multiple channels of real-time communication due to sensors. In this section we elaborate on these differences by defining the technical domain for data-intensive mobile apps.

In Chapter 2, we proposed 6 questions (repeated in Table 4.1) that when answered define a technical domain. Here we will address each of these questions in the context of mobile app development to define the boundaries for a new technical domain. Using the answers to these questions, developers can determine if software fits in the technical domain of mobile app development. Answering these questions involves studying multiple examples of software from the mobile app technical domain to ensure that the answers are generalisable to all software in that technical domain rather than to that individual solution. Two key aspects of a technical domain are its independence from the application domain and the implementation domain, and that its technical concerns are discovered rather than created. The 6 questions are answered within
Table 4.1: Table showing the 6 questions to define a technical domain.

<table>
<thead>
<tr>
<th>Question</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. How is software deployed?</td>
</tr>
<tr>
<td>2. What are the usage patterns for the software in that space? (when, where and how is the software used)</td>
</tr>
<tr>
<td>3. What hardware capabilities are available to the developer?</td>
</tr>
<tr>
<td>4. What are the implications of the platform APIs and platform implemention decisions?</td>
</tr>
<tr>
<td>5. What are the external or 3rd party drivers for the software?</td>
</tr>
<tr>
<td>6. How is the software developed and tested?</td>
</tr>
</tbody>
</table>

these constraints and are described below.

1. How is software deployed?

Mobile apps are deployed through app stores where end users can search for and download apps. These app stores enable end users to rank and review apps. Each of the major platforms, Android and iOS, have a single official app store where all of the apps are available for download, often with constraints that apps must satisfy. Processing time for checking an app against these constraints must occur before the app can be made public and can take up to 14 days. This extra time needs to be accounted for during project planning stage. Another aspect of deploying apps, is that each app installed on a device has access to its own resources which are typically private to that app. However, mobile phones can be “jail broken”, where these software restrictions are removed. This is something that developers need to consider in the context of app security.
2. What are the usage patterns for the software in that space? (when, where and how is the software used)

Mobile devices are used on the go and always travel with their users. Thus, user interaction patterns are typically shorter than the usage of desktop applications as the user can quickly open an app anywhere and anytime. Mobile apps have a wide range of applications as there are apps that can function as alarms, track sleeping patterns, measure distances etc. This variety of apps means that mobile devices can be used anywhere and anytime providing different functionality. For example, when travelling a user may quickly check the time of a flight and may also use an app to learn more about the country they are visiting. Another key feature that leads to different interaction patterns is that many phones have a touch interface without a pointer. This impacts the design of apps as all buttons and interactive contact has to be designed for a finger. In addition to the UI, the touch interface also changes the way users interact with the device and how they explore new apps to uncover new features.

3. What hardware capabilities are available to the developer?

All software is limited by the hardware that it runs on and in this respect mobile app development is the same as other traditional development platforms such as desktop software. However, mobile devices come with a wide range of sensors that provide additional capabilities to developers including the use of geofences, location tracking, motion sensors, cameras, Bluetooth connectivity, Near-Field Communication chips and screens supporting a multitude of touch gestures. Using these sensors presents unique challenges to mobile app developers. Other systems such as desktop software can take advantage of hardware capabilities such as cameras in the same way mobile apps can. However, all modern mobile devices have cameras whereas not all desktops have web cameras as these devices have to be purchased from a 3rd party. In this manner we see that dealing with a camera is a first-class citizen of mobile app development but not desktop even though both can use this hardware feature.
4. What are the implications of the platform APIs and platform implementation decisions?

Mobile apps predominately rely on APIs that are provided by the target platforms. In the case of mobile apps, platform vendors provide low level APIs that control every aspect of the mobile device. This control and flexibility contrasts with web apps that have limited access to the low level hardware due to the limitations of having to run in a web browser. On Android devices, developers can create widgets as well as apps. A widget is an app like program that displays information to the user on their home screen without the app being opened. Widgets are ideal for showing up-to-date information such as weather, stock market information or breaking news.

Mobile devices have limited resources and there are many operating system level design decisions made to minimise resource consumption. One reoccurring strategy is to aggressively reclaim resources such as shutting down long running apps that use too much power. Another implementation based decision is that of having a single UI thread which requires developers to write long running operations in a background service to avoid blocking the interface.

5. What are the external or 3rd party drivers for the software?

In mobile app development the main 3rd party influence on software is that of the platform vendor (Apple for iOS and Google for Android). These vendors control the types of apps that get uploaded, the ecosystem in which apps are available to the public and control the ranking boards for which apps get shown to users, impacting discoverability. Another main influence of the platform vendors is that of the UI design language. These companies are responsible for designing a consistent design language for all apps that are developed for that platform and these languages change frequently requiring developers to update their designs.

Supporting the mobile app ecosystem are many different vendors that provide services to app developers including Mobile Backend as a Ser-
vice, ad networks, push notification services, analytics frameworks and bug reporting systems. Deciding to use a 3rd party service depends on a multitude of factors such as cost, expertise and time to implement. Integrating these systems so that they work together and designing an appropriate software architecture is left to the developer.

6. How is the software developed and tested?

App developers use an IDE such as Android Studio to build, test and debug a mobile app. Testing mobile apps requires running the app on a device although some IDEs have built in device emulators. Emulation is an important aspect of app development as there are many different types of devices with varying screen sizes that an app needs to be tested against. Deploying to a device requires a cable and may require additional software to install the same app on multiple devices at once. Often apps will need to run on more than one platform or there will be different versions of the app, one for each platform. All of these variations need to be carefully tested for all supported scenarios. To help with the testing process there are a wide range of frameworks for automation to avoid the tedium of manual testing. In addition, helper apps can be used to mock information such as user location so that apps that use sensors can be tested in an office environment.

4.2.2 Process 1. Modelling the Technical Domain

The focus of modelling the technical domain is to identify what influences software architecture. These form the technical domain concerns and are usually uncovered by looking at how the technical concerns influence architecture. The how represents the quality attributes and the Problem Context describes when these concerns will influence software. During this process potential solutions to these Problem Contexts will arise and these form the tactic candidates to be refined in the next stage. Each of the steps to model the Technical Domain will be described below.
Domain Analysis

Modelling the technical domain begins by analysing different aspects of the domain. Domain Analysis is performed by studying the domain and answering three questions that pertain to software architecture. These three questions are outlined below:

• **What technical concerns of the domain influence software architecture?** To answer this question we sought to identify elements of the system that would influence the architectural decisions. Predominately this would require structural changes that would influence the design of multiple modules or were not easy to reverse later on in the development cycle. These concerns formed the first set of domain concerns for the technical domain.

• **How does the domain influence software architecture?** Addressing this question involved looking at the quality attributes that were of particular importance for the technical domain. By importance we mean the frequency in which these quality attributes had to be addressed in the technical domain or the impact that they would have if ignored. For example, ignoring the quality attribute of accuracy in a real-time system such as train signals would be catastrophic where as an app that tracks the distance you ran can afford to be less accurate – out by a few meters is not going to kill anybody.

• **When the technical domain concerns influence software architecture?** Not every software system developed in a technical domain will be influenced by each of the technical concerns – each system has different requirements with different features. Answering the **when** provides developers with a way to determine if they need to address each technical domain concern based on the requirements of their app. This was answered by studying the situations when developers need to address either a quality attribute or a domain concern. These situations became the Problem Context.

While each of these questions is presented in isolation in reality they are interconnected. For example, a technical concern cannot be identified
unless it’s influences on software architecture is also known – the how. What these provide is a new way to analyse the domain and a structured way to determine concepts that relate to software architecture.

The focus of this stage is to identify how the concerns influence architecture and when these concerns are relevant to new projects. Input to the Domain Analysis stage includes experience, source code, literature and documentation as shown in Figure 4.1.

When analyzing input sources, a Problem Context for when a concern is relevant, along with a corresponding solution, may also emerge. This is especially true when analyzing documentation for a particular technology where common problems and their solutions are recorded. These solutions are Tactic Candidates that are analysed further in Process 2 to determine if they are Domain Tactics.

Once the initial domain analysis has been completed experts review the initial model of the technical domain.

**Expert Review**

The Expert Review stage is used to validate the mined domain knowledge to ensure it captures architectural concerns. Mined domain knowledge at this stage consists of a set of technical domain concerns, a list of Problem Contexts with some related quality attributes and collection of tactic candidates. At this stage the Problem Contexts may be associated with a single quality attribute for example the Problem Context: “Tracking a User” has been associated with performance but accuracy has yet to be identified as a tradeoff. The focus of this stage is on validating the relationships between the Technical Domain Concerns, the quality attributes that they influence, and the Problem Context that identify when the concerns influence software. Experts are asked if the problems identified in the Domain Analysis stage are common and if they are aware of other solutions. These solutions get added to the set of identified Domain Tactics to ensure that all potential solutions are found.
Another focus of this stage is to get the Expert to analyse the language used to describe the problem. The problems need to be recognisable to professional developers so they can quickly navigate through the tactic catalogue and find the solutions they are looking for. The Expert is also asked to share any other problems that they commonly face when developing mobile apps. Not all of these problems will be specifically related to the technical domain which is why a Consensus Building stage is needed which is the next stage in this process.

**Consensus Building**

This stage focuses on building a consensus between the architect’s understanding of the domain and the expert’s. By analysing the feedback provided by the expert review areas can be identified that need more information. One example of this would involve checking academic papers for references to a problem suggested by the expert. This stage is also used to validate the suggested problems with experts. Care has to be taken to ensure that all problems that are proposed are unique to the technical domain and are independent of the business domain.

During the Consensus Building stage additional information may be required. For example, we may need to analyse more projects to see if a Problem Context is common to this technical domain. Multiple iterations of this process may be required to build out the list of Problem Contexts and to gather potential Domain Tactics. This process continues until no new problems are being presented. Then it is time to move onto Process 2 where we mine Domain Tactics.

### 4.2.3 Process 2. Validating Domain Tactics

Now that we have a set of Problem Contexts and a set of Candidate Domain Tactics we need to validate these tactics. This stage involves studying software from the domain to extract and document Domain Tactics. Each of the steps that makes up Process 2 is shown in Figure 4.2.
Finally, the tactics are evaluated by a domain expert to ensure that the tactics accurately represent solutions for this technical domain. By a domain expert we mean a senior software developer with at least 5 years experience developing mobile apps. After this process is complete architects have a set of Domain Tactics for a specific technical domain.

**Software Analysis**

Architectural knowledge builds upon what is done in practice, i.e. solutions that have been tried and tested. As with architectural tactics, Domain Tactis are not created rather they are discovered. At this stage of the process we are looking to refine the Tactic Candidates produced from the previous process by checking that they are truly general solutions to the problem they solve. In order to verify the generality of these solutions we need a collection of software representative of the technical domain under analysis. Open source software is readily available from online repositories such as Github\(^1\) that provide an obvious (and rich) source of exemplars for domain tactics. Ideally, commercial software would also be analysed to ensure that professionals use these tactic candidates but access to such software is not always available. To overcome this problem we include a review with domain experts.

---

\(^1\)https://github.com/
Software repositories should be collected that address the problems discovered for the domain as identified in the previous process. This ensures that, when searching for Domain Tactics, you can be sure a solution does exist although it remains to be determined whether any identified solutions are common. Mining Domain Tactics is a manual task, as the form of a tactic is not known before it has been identified.

**Technical Domain Expert Review**

The final stage of the process is to review and validate the Domain Tactics with an expert. An important part of the reviewer’s task is to ensure that a Domain Tactic is documented correctly. The appropriate quality attributes that are influenced by the Domain Tactic must be part of the documented tactic. These are core elements of a Domain Tactic that architects will use to reason about, and consider tradeoffs when designing their systems. Domain experts also need to ensure that there is sufficient information in the solution of a Domain Tactic to guide the implementation. The solution of a Domain Tactic is specific enough that a developer can implement it but must also be generally applicable to software developed in that technical domain.

**4.2.4 Extending a set of Domain Tactics**

Software in a domain changes frequently and this is especially true for the domain of mobile app development. Platform vendors constantly release a new Software Development Kit with new features and hardware vendors release increasingly advanced mobile devices. As both of our processes are based on the analysis of software and related resources they can both be used to update a set of Domain Tactics. Online documentation provided by the platform vendors is one source that is likely to be updated frequently as new features come out. As developers use the new technology new tactics will start to form and can be mined at a later date. The set of Domain Tactics capture the current best practices that have been tried and tested.
4.3 The Technical Domain and Software Architecture

By following the process described above we analysed the mobile app technical domain. In this section we define the different characteristics required for modelling the technical domain, how it relates to software architecture and how this information encourages knowledge transfer.

To understand how the technical domain influences software architecture consider software developed for use in a trauma centre. A trauma centre is an area of a hospital designed to treat patients suffering from traumatic injuries. Surgeons frequently make decisions that impact a patient’s life based on information provided by software and are often under a high degree of stress. Software designed to operate in this environment needs to take these factors into consideration. For example, in a trauma centre there is software that monitors a patient’s vital signs in real time which needs to provide accurate information to ensure that the patient is not given the wrong treatment [162]. Thus, accuracy is an important quality attribute to consider for any system that monitors a patient.

In this example, we have three distinct parts (see Figure 4.3): the real time monitoring of a patient’s vital signs, the quality attribute accuracy and the life critical nature of the operating environment. The critical nature of the software operating in this environment can be considered as the technical domain concern “Life Critical System” as this is a key characteristic of this technical domain. Here the technical domain concern is associated with the quality attribute accuracy through a feature of the software: real time monitoring of a patient. We will refer to these features as a Problem Context as they may not always be a feature of the software, i.e. different countries may have different laws that influence software architecture.

From this example we can see that the Problem Context tells us when a technical domain concern influences software architecture i.e. when the software needs to consider that concern, the quality attribute tells us how the architecture is influenced and the technical concern tells
The Problem Contexts are encountered by developers over time as they gain experience building applications in that technical domain. Intuitively, through experience developers will also come to understand the influences the different aspects of the technical domain have on the architecture of the application being built. With a technical domain model this information can be presented to a novice developer upfront and provide awareness of the types of challenges they are likely to face. In this manner the technical domain model can transfer deep technical domain knowledge to less experienced developers.

The general relationship between the Problem Context, Technical Domain Concerns and Quality Attributes are shown in Figure 4.4. Each Problem Context can influence multiple Quality Attributes. In our example above, the quality attribute of Robustness is also very important as system failure at a crucial time during surgery could cost a life. Multiple Technical Domains may also be influenced by a Problem Context. In the trauma centre, the technical concern of Hardware Requirements captures the need for information from the monitoring systems to be easy to understand at a quick glance. The primary reason for this is that the surgeons work in a stressful environment and cannot afford to make careless slip ups. This section describes the Technical Domain Concerns, Quality Attributes, Problem Context and Domain Tactics in more detail.

**Figure 4.3:** An example of a Technical Concern, Quality Attribute and Problem Context for the software operating in a trauma centre.

us *what* was influenced.
4.3.1 Technical Domain Concerns

To understand how the study of technical domain concerns fit in with the architecture process consider Figure 4.5. Here we can see that the technical influence upon the architecture is extracted through stakeholder engagement. In this section we present how technical domain knowledge can help software architects directly. The technical domain concerns are interrelated and impact each other often requiring developers to make tradeoffs on competing quality attributes. Thus, these concerns need to be reasoned about together so that the relationships between them can be analysed. Currently, to analyse these concerns, multiple viewpoints of an architecture have to be studied or the level of abstraction is too high for architects to make informed decisions on the specific details of mobile apps.
What is a Technical Domain Concern?

The Technical Domain Concerns describe the key elements of the technical domain that influence software. These concerns are often what distinguish technical domains from one another for example mobile apps have the technical concern of limited screen real-estate where as desktop software does not. One way to consider the Technical Domain Concerns is that they describe *What technical concerns of the domain influence software architecture*. The Technical Domain Concerns are *What* influences architecture – they indicate which concern from the Technical Domain is influencing the architecture. Each concern is connected to one or more quality attributes in a Problem Context. Each of the technical domain concerns for the mobile app development domain will be discussed in Section 4.4.1.

Elements of a Technical Domain Concern

From our trauma centre example, we can see that there are a number of characteristics of a technical domain concern. We cover these below:

- **Common**: The concern must frequently arise when developing software for that particular technical domain.

- **Technical**: A technical domain concern must influence software architecture. One way to check this is to see if early design decisions need to be made relating to this concern. Below we describe how a concern can influence architecture through quality attributes.

- **Relevant**: The concern must be particularly relevant to the technical domain.

Example Domain Concern

Consider the technical domain of data-intensive mobile apps and an app designed to track your Pizza delivery. This app must track the driver’s location to notify users where their pizzas are. This example is related to the Technical Domain Concern of Sensing as the GPS is used
to locate each driver. Sensing is a technical domain concern because the developers need to understand the concerns related to using this features such as power consumption, processing data, device availability, accuracy etc. Sensing is of particular importance to the mobile app technical domain because all smart phones come with a wide range of sensors that developers can use in their apps. This is contrasted with other technical domains such as that of desktop apps. Desktop applications can be developed that use sensors but require additional hardware and thus are not considered first class citizens of the that technical domain.

4.3.2 Quality Attributes

Understanding a domain from an architectural perspective requires not just understanding the key concerns of the domain but also How the architecture is affected. Quality Attributes are known to exert a strong influence on software architecture [1] and specify how the technical concerns influence the architecture. Addressing a technical concern does not guarantee that the influenced quality attributes will be satisfied. The main reasons for this is that there are additional quality requirements specific to the software being developed, tradeoffs have to be made between competing quality attributes and the implementation also influences quality attributes.

Example Quality Attributes

The Quality Attributes can be identified by answering the question How does the domain influence software architecture? In the context of the pizza tracking app the How is that a delivery driver app influences the quality attributes of both performance and accuracy. Polling the driver’s location frequently improves the accuracy of the system but negatively impacts performance. On the other hand, improving performance by rarely updating the delivery driver’s location adversely impacts accuracy. This shows that tradeoffs have to be made when attempting to address the quality attributes influenced by the technical
domain concerns. Tradeoffs may also need to be made between quality attributes raised by stakeholders and quality attributes related to the technical domain concerns. For instance, the users who order pizzas and want to track their food require a certain level of accuracy for the location feature to be useful.

4.3.3 Problem Context

The Problem Context describes the situation in which the current problem applies to software being developed. Not all Problem Contexts will be relevant to all software developed in a specific technical domain and the description helps developers determine which problems are relevant. Consider the our pizza app example with the feature to track the delivery driver. This feature requires the app to identify the customer, fetch the location from a web service and then provide real-time data on a map to show the user where their pizza is. In addition, the pizza drivers need another app that constantly tracks their location and sends the data to the system. Here we see that the Problem Context involves the need to track the driver in real-time to show to the customer. This Problem Context involves Sensing as the GPS is required for location data and influences the quality attribute of performance. The “What” is the sensors (i.e. Sensing technical domain concern), the Problem Context is to provide an overview of users on a dashboard and the “How” is performance. This Problem Context is only relevant because the app involves providing real-time location data.

What is the Problem Context?

A Problem Context describes when the technical domain concerns influence software architecture.

Not all technical concerns will influence software as it depends on the features of the app being built. The Problem Context is used to inform the architect when a particular Domain Concern is relevant to the system they are designing – it focuses on the When. For our example, if the app did not have the tracking feature the technical concern of Sensing
would not influence the software architecture. Problem Contexts can be discovered by identifying common problems that architects face and can also be found by reading requirement documents.

Another way to understand a Problem Context is to consider the architect’s role in eliciting a system’s requirements. This is typically achieved through stakeholder engagement to establish the functional and non-functional requirements for the system. Currently architect’s rely on their experience to elicit the requirements from the technical domain as it is not represented by a stakeholder i.e. they cannot ask anyone for the requirements of the technical domain. The Problem Context acts as a way for an architect to “query” the technical domain to find out how the architecture is influenced by the technical concerns.

A Problem Context that pertains to a specific technical domain consists of the following:

- Technical Domain Concern: Each Problem Context has 1 or more associated technical domain concerns. These concerns describe which aspects of the technical domain are influenced by this problem.

- Quality Attribute: A quality attribute describes how the Problem Context impacts software architecture.

- Description: This describes when the problem context arises and contains specific technical domain concerns and quality attributes.

### 4.3.4 Domain Tactics

While a Problem Context describes the situation when a technical concern influences software Domain Tactics describe a solution in this context. See Figure 4.6 to see how Domain Tactics are related to the Problem Context, Domain Concerns and Quality Attributes. In this section we describe the characteristics of a Domain Tactic and how they can be used to document technical domain specific architectural knowledge.
Example Problem Context

Considering our pizza delivery example, the architect gathers the requirements for the pizza delivery app. One of these requirements from the stakeholders is the ability for a user to track where their pizza is. The problem context here is “Track a User” with the associated technical concern of Sensing that influences the quality attribute of performance. Here we can see how the Sensing concern is linked with a quality attribute. From this the architect can then see clearly What (Sensing) and How (Performance) the software architecture will be influenced. This becomes a concern when any app needs to track location. Problem Contexts capture a facet of the technical domain and thus, have technical domain specific solutions. These solutions are referred to as Domain Tactics and are discussed in the next subsection.

What is a Domain Tactic?

A Domain Tactic is a solution to a specific Problem Context and is associated with a particular technical domain.

Domain Tactics are a way to capture technical domain specific architectural knowledge in a structured format and provide guidelines on how to implement a solution to that Problem Context. This is similar to Design Patterns that outline a solution and architectural tactics that are as-
associated for specific quality attributes. In contrast to Design Patterns, Domain Tactics are only relevant to the specific technical domain and do not provide any code examples rather providing a high level overview of what a solution would look like.

Domain Tactics differ from Architectural Tactics in three key ways 1) Architectural Tactics address a quality attribute model [68] where as a domain tactic addresses a technical domain concern that might involve multiple quality attributes, 2) Architectural Tactics are not limited to any specific technical domain where as a domain tactic may only be relevant to a few technical domains i.e. limited screen space in the mobile app technical domain is not relevant for robot control software and 3) Architectural Tactics provide no information on the technical details for realising an implementation. Domain Tactics on the other hand provide implementation details with tradeoffs and limitations that need to be considered.

Idioms and patterns are two commonly used techniques when building software [163]. Idioms are low level features or constructs that may be present in one or more programming languages. These are basic building blocks that developers use to construct software. Above idioms are design patterns [5] that address a reusable solution to a reoccurring problem in a given context. Domain Tactics may require multiple design patterns and idioms to satisfy the Problem Context that they are designed to address. Domain Tactics also depend on the technical domain of the software being developed where as a Design Pattern such as the Observer Pattern is usable in software developed across all domains.

Domain Tactics are a technical domain specific building block used to construct software architecture and are often a specialisation of an architectural tactic. Domain Tactics contain more specific information relating to a single technical domain where as an architectural tactic is a means of satisfying a quality-attribute-response measure [157]. However, Domain Tactics are intended to be platform and programming language independent. Domain Tactics also include information on how they influence the software architecture, such as the quality attributes
that they help to achieve and the quality attributes that are adversely affected. These quality attributes aid in comparing tactics and guide the architect in making tradeoffs.

In the mobile app technical domain there is the Domain Tactic of Shipping Data on the device, where data is loaded onto the device when it is first installed and the data updates in the background. This tactic is not unique to mobile apps as many applications contain data and make use of background services. However, users often use their mobile apps on the go where they may be unable to access a network connection such as in a tunnel. The likelihood of a mobile app being used when the user has no network connection is greater than a desktop app that can be assumed to have a constant connection. As such the Domain Tactic of shipping data on the device to provide content to the user is of particular relevance to mobile apps.

The following subsections will provide more details of a domain tactic and key attributes that need to be documented.

**Elements of a Domain Tactic**

To document a Domain Tactic we developed a pattern template similar to those used for design patterns [5]. Our structure for describing a domain tactic is inspired by that of design patterns it includes specifying the benefits and limitations of using this tactic. How they are created is described below in Section 4.2.

Our pattern template is based on the elements of a pattern as described by Appleton [81] with a few modifications. The first key point is that each domain tactic is associated to a specific Problem Context which means that multiple domain tactics will address the same problem. As this describes when the domain tactic is to be used we removed the problem and context elements from our template.

Another major modification we made was the division of the element “Forces” into Benefits and Liabilities. We did this to explicitly describe the quality attributes that were positively and negatively impacted by
using this domain tactic. At a glance developers can clearly see the impact of choosing to implement this domain tactic.

We also added the concept of Technical Domain Concerns which maps the domain tactic to the relevant concern from the domain. By doing this we were able to provide developers with a rationale for using this domain tactic and specify the resulting context. The Technical Domain Concerns also provided another way to identify related domain tactics as many tactics addressed the same technical concern.

The Domain Tactics outline the quality attributes they help satisfy and the ones that they negatively impact. The template used to describe Domain Tactics is outlined below:

- **Name:** A short name of the tactic to aid communication between stakeholders.

- **Technical Domain Concerns:** The attribute from the domain that influences the implementation of this tactic. Listing the technical concerns helps developers choose between competing Domain Tactics as they need to factor in the other requirements for their app as well as business constraints, skills, money and client.

- **Benefits:** A list of quality attributes that this tactic positively impacts. These are used to compare tactics and to help satisfy desired quality attributes for the system.

- **Liabilities:** A list of quality attributes that are negatively impacted by this tactic.

- **Solution:** Contains a list of actions to take to implement the tactic. These actions describe the solution at a high level and it is up to a developer to realise an implementation. Domain Tactics are designed to be used by professional developers who need guidance at a conceptual level for the problems and solutions they will face for a new technical domain.
Example Domain Tactic

In this section we provide a detailed example of a Domain Tactic that addresses a specific Problem Context. A common problem in mobile app development is the need to show the user information about a new event. For example, a shopping app may wish to inform the user of this week’s special or a newspaper app wants to let the user know that there are new articles to download. While this problem is common to other technical domains the Domain Tactic of using Push Notifications is a critical solution used in mobile apps and is particularly important. The Domain Tactic will be described below with a discussion on each of the main aspects to document covered below.

Name: Push Notifications.

Problem: Users need to be notified of new content or a new event.

Technical Domain Concerns:

• Ecosystem – Mobile platforms and 3rd party vendors provide a mechanism for pushing content to a device. This avoids the need to manually poll a server to determine if new content is available.

• Network Connectivity – Push notifications require a network connection to function correctly.

Benefits:

• Usability – Real-time data can be provided to the user when and if they need it. This improves usability as notifications can be shown to the user as soon as the data is available even when they are not using the app.

• Availability – Content is made available to the user as soon as it is ready.

Liabilities:
• Usability – Abuse of the notification system by triggering too many notifications can cause devices to ring or vibrate which can irritate users. Notifications should be used only for the most important or most current information.

Solution:

• Integrate with a push notification service or use the platform specific APIs.

• Ensure that there is a screen in the app that can manage notifications. Common functionality of a notification screen includes viewing a list of recent notifications, highlighting which ones that have not been read and being able to delete 1 or many notifications at once.

Notice that the Domain Tactic describes the core aspects of the solution and avoids code specific examples. The main reason this is done is to ensure that the Domain Tactic is platform independent and focuses on capturing common solutions of the technical domain. From our evaluation, discussed in Chapter 7 we found that we did need to provide links to some concrete code examples that developers could run, an area for future work.

4.4 Results of Analysing the Mobile App Technical Domain

In this section we present our results from analysing the Mobile App Technical Domain. Technical Domain knowledge was extracted following the process described in Section 4.2 which is based on the analysis of mobile apps. A technical domain model for mobile apps can provide a way to transfer technical domain specific knowledge to novice developers who do not have the experience to know these concerns intuitively. The technical domain knowledge is presented as described in
the previous section in terms of Domain Concerns, Problem Contexts and Domain Tactics.

### 4.4.1 Mobile App Technical Domain Concerns

From analysing the mobile apps we created a technical domain model which is shown in Figure 4.7 and will be described in more detail below.

![Figure 4.7: Concerns and relationships that make up the technical domain model for data-intensive mobile apps. White boxes show how the concerns relate to a mobile app.](image)

As users often carry their devices with them, developers have to contend with issues pertaining to *Network Connectivity* – namely handling a sudden drop in connection and variable signal strengths. When developing for the desktop, the assumption was that a steady network
connection was available and a lost connection was a rarity rather than an eventuality – cares of a desktop server application often need to be treated as first class citizens in a mobile app.

Mobile devices also contain sensors that enable developers to take advantage of their ubiquitous nature. The Sensing capabilities of a mobile device come built in and provide developers with a means to determine a users context such as their current mode of transport or their location. The plug-and-play hardware available for other computer systems are generally designed for a static indoor environment. Other distinguishing characteristics of mobile app development relate to Hardware Constraints due to size limitations, compact hardware, and the ecosystem surrounding the mobile platforms. Integrating with the Frameworks and Ecosystem of a mobile platform provide benefits to developers such as enabling them to sell their apps in an app store and add analytics to their apps for better error reporting.

The two remaining concerns that impact upon the architecture of a mobile app are Lifecycle representing the state model of an app and Threading Model used by the mobile frameworks. Another core aspect of mobile app development is that of Social and Personal. Mobile devices enable a high degree of communication and interaction with other people. In addition mobile devices are often owned by a single user who customises and tailors the applications to their preferences.

These concerns impact the architectural decisions that need to be made when developing mobile apps. This list was developed iteratively over the course of this thesis and covers all common concerns. However, as the answers to the questions in Section 4.2.1 change this model is expected to evolve. The following sections will provide additional detail for each of the domain concerns for the mobile technical domain.

**Hardware Constraints**

Power consumption represents one of the greatest concerns to both users and mobile app developers. Users of mobile apps rarely power off their devices and recharge them infrequently as they often keep them
nearby. Thus, users have an implicit expectation of efficiency as a quality attribute of an app. Mobile devices also have limited battery capacity due to their size which developers need to account for. The two major causes of power consumption on a mobile device is lighting up the touch screen and frequently processing sensor events. Relying upon hardware features of mobile devices creates its only set of problems.

New mobile devices with new hardware capabilities are frequently released to the market. While this creates new opportunities for apps based on these more powerful devices it also creates compatibility issues for developers especially as their mental models do not move as quickly. Older devices often lack features of newer devices requiring developers to provide an alternative means to achieve the same outcome or display a message to the user. Providing alternative functionality in this manner is known as degrading gracefully and can impact upon the architecture of a mobile app. A tradeoff needs to be made between usability and availability – taking advantage of the newest hardware may improve usability yet only be supported on a few devices, impacting availability. In some situations alternative functionality may not be possible and the decision to not support certain devices has to be made.

To build mobile apps that are energy efficient and have a smooth user experience developers need to understand the hardware limitations of mobile devices.

**Sensing**

A boon of mobile devices is the wide range of sensors available to app developers. These sensors provide information about the device’s orientation, motion, elevation and location. Mobile devices can also interact with their surroundings such as connecting to nearby devices through bluetooth, scanning barcodes and interacting with other devices via Near Field Communications (NFC).

Incorporating the use of sensors into a mobile app raises concerns relating to accuracy, reliability and timeliness. All hardware sensors have a margin of error in their readings and sensors found on a mobile de-
vice are no exception. The accuracy for sensor readings may also be influenced by other factors such as whether the device is indoors, outdoors or near noise sources (e.g. magnets). Both of these concerns need to be considered when designing the algorithms for processing sensor values. A common solution is to combine the readings of multiple sensors for processing. Reliability of a system can be influenced greatly by the sensor readings. Sensor readings are often presented as raw values and it is up to the developer to process them into something meaningful such as the users context. To smooth the values from the sensors developers need to apply various filters to the data and develop the necessary infrastructure to perform this processing. The sampling rate of the sensors can be another way to obtain reliable values. This approach impacts upon the timeliness of result as time passes between multiple readings of the sensor.

Sensors enable developers to add context aware functionality to their apps but their impact upon power consumption influences the architecture design i.e. processing is often performed on a server rather than on the device.

**Threading Model**

Most mobile operating systems typically run one app in the foreground and enforce a single User Interface (UI) thread model. A single UI Thread responds to user events and updates the UI. Any long running operations on this thread will impact greatly upon the usability of an app and can cause the UI to stop responding if the app architecture is not designed for this. This is in contrast to desktop applications where an app can crash but the whole user environment does not hang – the user can open up another application and continue using their system.

Mobile app developers need to create background tasks that smoothly interface with the UI thread for long running operations such as fetching data from an API and only updating the UI from the main thread. Another constraint faced in this model is that an app will be suspended if a user starts another app. Long running operations need to be able
to pick up from where they left off once the user navigates back to the suspended app. Running the appropriate tasks for an app on the appropriate thread can improve the performance of an app greatly. The suspended threads also create the issue of availability as the mobile operating system is optimised to deallocate both priority and resources aggressively. Should important time dependent tasks run on threads that get suspended for an extended period of time the accuracy of the results is impacted.

There are threads that do not get suspended by the mobile operating system and run in the background. These threads do not require the app to be running on the UI thread to be operating and perform tasks such as fetching data from an API or polling sensor data. Design decisions need to be made on which type of thread to perform the long running tasks with the implications of each considered.

Developers must pay careful attention to the use of threads in the mobile app as some mobile operation systems shutdown apps that lock the user interface.

**Lifecycle**

On a mobile device there are multiple concurrent events, each one of them interrupts and competes for a user’s attention. Some apps such as the Dialer app have a higher priority as a user most likely wants to be interrupted from their current task by an incoming phone call. When this happens the current app on the main thread is suspended and the Dialer app replaces it. Once the call has ended the user navigates back to the original app and expects it to be just as they left it. To handle this, mobile frameworks have a state machine that an app transitions through which enable developers to appropriately save and restore user state in the event of being suspended. The state model for a mobile app is known as the lifecycle of a mobile app (see Figure 4.8 and Figure 4.9). Mobile operating systems differ from traditional operating systems in that they will shut down an app if it consumes too many resources or has not been activated by the user recently. Developers also need to
gracefully shutdown the app in this situation and restore the state as
the user expects it next time the app is opened. Developers need to be
aware of this, understand these states and ensure that the appropriate
operations are performed at the appropriate states.

![Lifecycle state model for an Apple iOS app](image)

**Figure 4.8**: Lifecycle state model for an Apple iOS app\(^2\).

Some mobile platforms also allow apps to provide functionality to other
apps. For example, on the Android platform the Camera screen from
the Camera app can be started from other apps. This functionality
needs to be designed into the app and the appropriate data passed
back to the calling app once the user has finished their task. Care also
needs to be taken with the navigation model as users may not want
to navigate through the called app. Long running operations should
be stopped as soon as they can be or the operating system will do it
automatically. Operations such as animations or playing a video do not
need to continue if there is a UI component covering them. To build
robust mobile apps, a thorough understanding of the app lifecycle is
needed so each state can be suitably designed for.
Network Connectivity

Network Connectivity plays a vital role in data driven apps as data needs to be passed to and fro between client and server. Mobile devices are often used on the go and have variable connectivity: 3g, 4g vs wireless – often changing over a few minutes. This variable network connectivity influences how much data can be passed between server and client. For example, downloading images or streaming movies on a wireless connection will be a lot faster than on a slow 3g connection. Ignoring this issue can involve apps being built that have poor usability.

Another issue relating to network connectivity is the sudden drop in connectivity. For instance, users can lose and regain connections when travelling on trains when they go through different cells. In addition to this, the user may open up the app for the very first time without any network connection. Displaying information that is still relevant to a user although slightly out of date is one way to address a lack of connectivity when the app first opens. These are usability and performance issues that cause poor user reviews. To overcome such issues, developers need to cache data and store information on the phone to provide a smooth user experience even when there is no network connectivity.

In the case where a thread is performing a network operation, once the user returns to the app or the network connection is re-established the
thread will continue trying to load information potentially from a partial
transaction. Unless the server infrastructure is designed to send part
of the data back to the client the app will crash. Developers need to
cater for this issue and design their server layer and the client apps
appropriately.

Another concern for mobile devices is that they are generally left in
roaming mode. This allows mobile devices to establish connections with
any available WiFi point creating concerns about the security and pri-
vacy of data. This is especially pertinent to data-intensive apps that
have access to private data such as credit card information.

Network Connectivity is of paramount importance for data-intensive
apps as it influences data availability and the user experience.

Framework and Ecosystem

Framework and Ecosystem refers to the environment in which mobile
apps are deployed and run in. This includes the integration of 3rd
party frameworks to provide additional services such as ad networks
and analytic frameworks. These 3rd party frameworks are crucial to
the success of an app and are commonly used in data-intensive mobile
apps.

There are four sub-categories that make up the Framework and Ecosys-
tem concern.

- **Platform Mandates**: A major part of app development is integrat-
ing into the current framework and ecosystem developed by mobile
platforms, iOS and Android being the most common. Apps are
mainly distributed though an app store pertaining to each plat-
form and in some cases developers have to satisfy certain require-
ments to get their apps released. For example all apps submitted
to Apple’s App Store are reviewed and developers must ad-hear to
their App Review Guidelines4.

• **Platform Trends:** Visual appeal and a great user experience are key ingredients in developing a successful app. This is in part due to the app stores encouraging greater user interaction through user rankings and reviews. One aspect of satisfying these requirements requires developers to build apps that match the latest UI trends and patterns to ensure that their apps do not look outdated. Platforms create these fashions and create UI guidelines that developers can follow. However, often the styles outpace the API development and developers often have to implement new widgets and UI patterns from scratch. Android’s design language Holo\(^5\) has recently been replaced by Material\(^6\). In addition to UI trends the mobile ecosystems evolve quite quickly with new tools, libraries, frameworks and languages coming out frequently.

• **3rd Party Frameworks:** In order to optimise the benefit of developing as part of a mobile app ecosystem developers often find the need to integrate 3rd party frameworks into their app. A common and increasingly popular practice in mobile app development is to integrate bug reporting functionality. Due to the nature of mobile apps bug reporting functionality is often built into an app. This enables developers to get more information about a bug and potentially decreases the time it takes to find and fix the issue. This is essential for users who actively engage in the ecosystem and review apps. Mobile apps also offer novel ways to make money for native apps. Two common strategies involve showing fullscreen or part screen advertisements and having a paid app often with a free lite or demo version. Each of these approaches influences the design of the architecture for an app and need to be considered early on in the development lifecycle. To monitor the success of these approaches developers integrate analytics frameworks into their apps that provide information about user clicks and where the user spends the most time. These frameworks have to be included within an app and setup appropriately, yet another influence on the software architecture.

• **Cross Platform:** To reach as many customers as possible devel-

\(^{5}\)http://developer.android.com/design/style/themes.html

\(^{6}\)http://developer.android.com/design/material/index.html
operators need to support multiple mobile platforms. Mobile websites built with Javascript and HTML 5 can easily support multiple platforms but do not have the benefits of being distributed through an app store. Hybrid apps aim to overcome this limitation by providing a web container for the web app allowing developers to wrap their web apps for distribution. This approach also allows access to certain native features of a mobile device such as its sensors. Choosing between a native or hybrid app influences app architecture as a hybrid app’s UI predominately uses web technologies and patterns. Native apps have greater flexibility as they are not restricted by the constraints imposed by a web container and a 3rd party wrapper.

In summary the Framework and Ecosystem capture the involvement of 3rd party frameworks and the influence of the mobile platforms have on data-intensive mobile apps.

**Social and Personal**

Mobile devices provide the opportunity for greater social interaction than other software systems as mobile phones are always on and always with their users providing an open channel of communication. In addition to the traditional, calling and texting, mobile apps also provides means to communicate through social media platforms such as Twitter\(^7\), Facebook\(^8\) and Google+\(^9\). Here users can not only communicate with other users but interact with content others have created expressing their opinions through comments and/or “like” functionality. Developers can take advantage of these features in their apps by interacting with other social media apps directly such as through Android’s intent mechanism\(^10\) or through integrating a 3rd party Software Development Kit (SDK). The choice of integration depends on the level of functionality desired and impacts upon the software architecture as

---

\(^7\)https://twitter.com/  
\(^8\)https://www.facebook.com/  
\(^9\)https://plus.google.com/  
\(^10\)http://developer.android.com/reference/android/content/Intent.html
a decision has to be made on where the integration will be implemented. Another big feature of social media is the concern of sharing – where content you have created is shared with the world. Sharing functionality influences the architecture of a mobile app as decisions have to be made surrounding what is shared, where and by what means.

Mobile devices often travel with their owners enabling the completion of tasks on the go that once required sitting down at a computer. A consequence of this is that mobile devices are generally only used by a single user as opposed to multiple users having private accounts on the family PC.

As mobile devices are a personal device many mobile apps provide settings for users to be able to configure an app to function according to their preferences. Settings functionality needs to be designed into an app and influences the design of an app. User preferences can also be an aide to improve the usability of an app by allowing users to access information more easily that is relevant to them. The concept of a favourite screen is one such feature. A favourite’s screen allows users to save items that are of interest to them allowing them to jump straight to the information that they have previously saved as interesting to them.

Another consequence of being such a personal device is that mobiles often contain personal information such as contact lists, emails, friend’s addresses etc. This raises the concern of users’ privacy that needs to be addressed by developers. Mobile platforms help protect users privacy by forcing developers to enable permissions for certain features. As mobile devices are more mobile than other software systems they are more susceptible to being lost, damaged or stolen. Care needs to be taken deciding what is stored on a device and the architecture of the app needs to reflect these decisions. Although mobile platforms have a security model that prevents apps from accessing the data from other apps instructions are easily accessible online for overcoming these safeguards.
4.4.2 Mobile Domain Problem Contexts

Outlined in this section are the identified Problem Contexts for data-intensive mobile apps.

**Multiple platforms need to be supported.** According to Netmarket-share\(^{11}\) 94% of the mobile devices run either Android or iOS requiring developers to support both platforms. As such developers need tools, techniques and frameworks that assist in supporting multiple platforms which is related to the Frameworks and Ecosystem domain concern. Apps that are developed for one platform need to be portable so they can be extended for another platform. Supporting two or more platforms also means there is a trade-off between maintainability of feature set and flexibility by leveraging platform specific functionality.

**App depends heavily on data stored in the cloud.** Many apps rely on fetching content from a webservice to show to the user. The main domain concern here is that of Network Connectivity as a mobile device moves through many networks which influence the quality attribute of Reliability. A lack of data on the device also impacts usability as data driven apps rely on data.

**The battery drains quickly.** Mobile devices have limited battery capacity and are always with their users so battery life is a main concern. In our domain model we capture this by the Hardware Constraints concern. Quality attributes impacted by this Problem Context include performance as a low battery means slower performance. Sensing is also related to this problem context as using the sensors on a device is not power efficient.

**Must support many devices with different hardware capabilities.** There are a plethora of different mobile devices available with different hardware capabilities, Hardware Constraints. Some of these capabilities are dependent on the mobile platform vendor (Frameworks and Ecosystem) – Apple creates the iOS platform and the devices that run

\(^{11}\)https://www.netmarketshare.com/operating-system-market-share.aspx?qprid=8&qpcustomd=1
iOS. This problem context directly influences usability as some functionality may not be available if the device does not support the necessary capabilities.

**Users need to be made aware of fresh content (or new event).** Data-intensive apps often require fresh content to relevant data to the user. This problem context relates to Framework and Ecosystem as both the major platforms support a push notification service and overlaps with the concern of Network Connectivity. Notifications provide a means of directly showing content to the user at any time which improves usability as users do not need to open the app. However, spamming the notification system can annoy the user and provide a negative user experience.

**App needs to run on devices with varying screen sizes.** Both iOS and Android support a wide range of devices that have different screen sizes for mobiles and tablets (Frameworks and Ecosystem). To adequately cater for the different screen sizes developers need strategies for handling layouts and assets especially for small screens such as Apple Watch\(^\text{12}\) (Hardware Constraints). The main reason for supporting multiple screen sizes in an app is to support more devices and attract more users.

**Data from the app needs to be shared with other people or integrated with other apps.** The Android platform provides a way for data from an app to be shared with a social media platform or integrated into another app. One of the main quality attributes that need to be considered when implementing this functionality is privacy and security of the data being shared. Sharing content between apps is a Framework and Ecosystem concern as not all platforms support it and will need another strategy for handling this functionality.

**App relies on the current user’s context i.e. location, time etc.** A prime strategy for reaching mobile app users is to provide relevant content to their user when they need it which is achieved through identifying the user’s current context. This relates to the quality attribute

of Functional Suitability as functionality is provide to the user when it is most relevant. Primarily this is achieved through using the devices sensors to determine context such as users current location or activity (Sensing).

**Users need to authenticate with the App.** Data-intensive mobile apps require secure authentication methods to be able to protect their user’s data. This problem context directly relates to the concerns of Social and Personal as often the data stored is private to that user. Security is the main quality attribute related to this Problem Context.

**App frequently freezes or crashes.** Mobile platforms implement a single UI threaded model as captured by the Threading Model concern which if not addressed can cause an app to become unresponsive. This is also related to the Lifecycle concern as the threading model needs to be handled in the appropriate state in the apps lifecycle. Preventing an app from crashing improves the usability of an app.

**User needs to customise the behaviour of the app.** Mobile devices are typically used by a single user which is represented in our model by Social and Personal. Allowing a user to customise their app influences maintainability as these preferences need to be made available throughout the whole app.

### 4.4.3 Mobile Domain Tactics

We now present a set of Mobile Domain Tactics identified by the mining processes described in Section 4.2. These domain tactics had to be found in multiple examples and address a technical domain concern in order to be recorded. We continued the mining process until no new domain tactics were discovered and our evaluation in Chapter 7 with developers indicated we had a broad coverage with our domain tactics.

For each of the problem contexts described above there are associated domain tactics that provide a solution to the problem. Developers compare the problem contexts against the features they have to implement and then from the related problem context select a domain tactic. We
map the domain tactics to the relevant problem context in Table 4.2.

Below we provide a list of the mined domain tactics for the mobile app technical domain with a short description for each. The full details of each domain tactic can be found in Appendix B.

- **Thin Clients** - For this domain tactic developers move the business logic to a webservice which provides the content for the mobile app. The mobile app is then responsible for rendering the content to the screen and on providing a great UX. A thin mobile client also focuses on logic for fetching content, using sensors and user interactions.

- **Hybrid Approach** - Web technology has made significant advances which has lead to plethora of hybrid platforms that leverage both native and web technologies. These approaches take advantage of a web view, an embedded web browser, to render parts of a website and native to handle heavy processing or rich interactions. As the web technology advances more of the app gets implemented in the web view. However, there are a number of complexities with this approach such as connecting content in the web view with native components.

- **Ship Data** - Mobile devices can be deployed with data to display to the user as soon as the app opens. This is in contrast to a website that must make an initial call to the webserivce before any content can be displayed. As mobile devices suffer from Network Connectivity issues this is a good tactic to use when data does not change frequently such as a train timetable.

- **Store Data on the Device** - Network Connectivity issues means that storing data on the device can greatly improve the usability of an app. Strategically implementing this tactic can lead to the creation of an “offline” app which seemingly does not require any network connection at all. In reality the connection with the network occurs in the background.

- **Batch Process** - Processing sensor values one at a time can be very time consuming and drain the battery. This tactics involves
Table 4.2: Table showing Problem Contexts and related Domain Tactics for data-intensive mobile apps.

<table>
<thead>
<tr>
<th>Problem Context</th>
<th>Domain Tactics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiple platforms need to be supported.</td>
<td>Thin Clients</td>
</tr>
<tr>
<td></td>
<td>Hybrid Approach</td>
</tr>
<tr>
<td>App depends heavily on data stored in the cloud.</td>
<td>Ship Data</td>
</tr>
<tr>
<td></td>
<td>Store Data on the Device</td>
</tr>
<tr>
<td>The battery drains quickly.</td>
<td>Batch Process</td>
</tr>
<tr>
<td></td>
<td>Limit Sensor Usage</td>
</tr>
<tr>
<td>Must support many devices with different hardware capabilities</td>
<td>Degrade Gracefully</td>
</tr>
<tr>
<td></td>
<td>Avoid Legacy Platforms</td>
</tr>
<tr>
<td>Users need to be made aware of fresh content (or new event).</td>
<td>Push Notifications</td>
</tr>
<tr>
<td>App needs to run on devices with varying screen sizes.</td>
<td>Specialise Asserts</td>
</tr>
<tr>
<td></td>
<td>Use Alternative layouts</td>
</tr>
<tr>
<td></td>
<td>Resize Images</td>
</tr>
<tr>
<td>Data from the app needs to be shared with other people or integrated with other apps.</td>
<td>Integrate with Social Media</td>
</tr>
<tr>
<td></td>
<td>Import/Export</td>
</tr>
<tr>
<td>App relies on the current user’s context i.e. location, time etc.</td>
<td>Determine Context</td>
</tr>
<tr>
<td>Users need to authenticate with the App.</td>
<td>Store Credentials</td>
</tr>
<tr>
<td>App frequently freezes or crashes.</td>
<td>Respect Single UI Thread</td>
</tr>
<tr>
<td>User needs to customise the behaviour of the app.</td>
<td>Implement a Settings Screen</td>
</tr>
</tbody>
</table>
batching sensor values together and sending them all to the server for processing in one go. The trade off here is accuracy for performance as the sooner you process values the more accurate the response but the more susceptible you are to noise.

- **Limit Sensor Usage** - Sensors place a significant strain on a mobile devices limited resources. This tactic provides guidelines for limiting the use of sensors and the benefits of doing so.

- **Degrade Gracefully** - Not all devices support the capabilities required for certain features such as NFC chips are not present in every smart phone. To support as many users as possible alternative features need to be enabled or available when devices do not support certain hardware capabilities. Developers provide fallback options to the user to achieve the same functionality or hide a feature so that the user does not feel as though they are missing out.

- **Avoid Legacy Platforms** - The easiest way to ensure that all the features are available to all users is to heavily restrict the number of supported platforms. Ideally developers will only support the latest and greatest platforms and devices. However, to reach the greatest number of customers this is not always possible and a tradeoff needs to be made between the latest technology and the number of customers.

- **Push Notifications** - Mobile devices have a mechanism for sending notifications to a device much like a text message that pops up on the screen. Users can be notified when they are not in the main app or even using their device. This mechanism provides a way to interact with a user on demand and needs to be carefully managed to avoid spamming the user.

- **Specialise Asserts** - Mobile platforms provide conventions to ensure that developers create asserts for the appropriate screen sizes. Developers need to be aware of this and follow the conventions used by each platform these guidelines are available in the official app development documentation\(^\text{13}\). Assets also need to be created

\(^{13}\)https://developer.android.com/guide/practices/screens_support.html
for each device that the platform supports.

- **Use Alternative Layouts** - Both iOS and Android use allow developers to create different layout files for different devices. This enables additional information to be displayed on a larger device such as a tablet. Developers should be aware of this tactic and follow the platform conventions to take advantage of alternative layouts.

- **Resize Images** - Another way developers can support multiple screen sizes is to have fixed assets that get resized on the device. By resizing on the device developers can ensure that each image is optimised for every screen size.

- **Integrate with Social Media** - Data-intensive mobile apps contain content that users may wish to share with their friends and followers which requires integrating a social media platform. Developers need to be aware of the privacy and security concerns that arise when sharing data. Some mobile platforms like Android provide inbuilt functionality to share content with other apps that are installed on the device.

- **Import/Export** - Data-intensive apps may store content which can be shared between platforms or that need to be backed up such as contents or calendar activities. Data needs to be exported in a format that is compatible with other tools and apps such as in a comma separated list file.

- **Determine Context** - One of the key characteristics of a mobile app is the many inbuilt sensors that developers can leverage when building their apps. By combining data from different sensors and using the latest algorithms the users context can be determined. This allows developers to customise the behaviour on the app based on what the user is doing such as notifying the user of shop sales when the user walks into the store.

- **Store Credentials** - Data-intensive apps often require a user to login to be able to use the app. To simplify this process Store Credentials involves storing an access token so that the user only needs to login to the app once.
• **Respect Single UI Thread** - Mobile apps have a single UI thread which requires long running processes to be performed in the background. Failing to do this can cause the app to hang or crash. Handling the UI thread requires understanding the lifecycle of the app and the types of operations that you are performing.

• **Implement a Settings Screen** - Integrating a customer’s preferences into a mobile app is usually done by creating a settings screen. On a settings screen a user can enable/disable features and specify configuration details such as their desired currency. These preferences are then integrated throughout the app to provide a custom experience.

### 4.5 Summary

In this chapter we answered the question *What are the concerns of the technical domain that influence mobile app architecture and how can this information be captured?* By answering this research question for the technical domain of mobile apps: we showed that considering the technical domain 1) provides a way to extract and document technical domain specific architectural knowledge, 2) informs architects of the major influences of the technical domain on software architecture and 3) helps architects make tradeoffs between competing solutions to common problems.

We addressed this problem by following two iterative processes, Process 1: Problem Context Analysis to analyse the domain and Process 2: Validating Domain Tactics to identify reusable technical domain specific architectural knowledge. Our analysis stage focused on looking at which concerns of the technical domain influenced quality attributes and when these concerns need to be addressed by developers. These processes enable the analysis of software architecture at the design stage as well as domain tactics for guiding a technical domain specific solution.

The major contributions of this chapter as follows:
• A method for capturing the characteristics of the technical domain that influence software architecture.

• A model for understanding the influence a technical domain has on mobile app architecture. In the following Chapter we build upon this model to build a meta-model for describing data-intensive mobile apps.

• A model for documenting technical domain tactics: reusable domain specific architectural knowledge. We build upon these tactics in Chapter 6 to realise a tool for generating sound software architecture for mobile apps.

• A set of concerns that influence mobile app architecture. These concerns influence the design of a meta-model for describing mobile apps presented in Chapter 5.
Chapter 5

A Method for Meta-model Construction

An appropriate meta-model is vital for the development of model based tools, DSLs in particular, that generate software architecture. Although the need for a good meta-model is well understood, current literature lacks a repeatable process for constructing a meta-model and evaluating the meta-model [42, 44, 47, 50]. Critically, current approaches fail to consider the technical domain as the meta-models do not consider concepts from the underlying technical domain as first-class citizens.

Compounding this situation, the current meta-models are created top-down – the full meta-model is designed before any development takes place and tend not to be based on the analysis of existing systems. In this chapter we address these gaps by answering the research question How do we empirically build a meta-model that contains the core abstractions a developer uses to build mobile apps? The meta-model considers the technical domain and also the core abstractions a developer uses to build mobile apps. Our method is based on the analysis of a set of apps, incorporates feedback from domain experts and is influenced by current literature.

The meta-models discussed in this chapter map to software developed in a specific technical domain. As such a meta-model and a technical
domain model are complimentary, each providing a different view on the software being developed. The technical domain model captures the technical concerns of the domain where as the meta-model describes the core concepts used to model software in that domain. Our method for developing a meta-model focuses on the technical domain of mobile app development.

This chapter is divided into the following sections: we talk about the relationship between a meta-model and an implementation of a DSL in Section 5.1. Next, we present our method for constructing a meta-model and how this method progresses smoothly into the design and implementation of a DSL in Section 5.2. Then we describe a mobile app meta-model in Section 5.3. Finally, we conclude with a summary of the chapter in Section 6.5.

5.1 Meta-models and Domain Specific Languages

In Model Driven Development (MDD), a model is a graph based structure that describes a system and conforms to another structure called a meta-model [164]. A meta-model is a model of a model provided that the respective model-of relationship is not transitive [165]. That is, given that a system is the object being described by a model, a meta-model describes those models, but cannot be used to describe the system. In this thesis we use the following definition for a DSL: “A domain-specific language (DSL) is a programming language or executable specification language that offers, through appropriate notations and abstractions, expressive power focused on, and usually restricted to, a particular problem domain” [166]. A DSL “is often a reduced language devoted to a specific domain and it is defined by creating abstractions that model the concepts of the domain and the interrelations between them” [167].

For a developer to leverage the full advantage of a meta-model the model needs to be included in a tool such as a DSL. Due to this tight relationship between a DSL and the underlying meta-model, we have relied on
an iterative process to co-evolve both the DSL and the underlying meta-
model. Our DSL is intended to be used by professional app developers
to describe the fundamental components of their app and then gener-
ate the foundation of a mobile app – one that respects technical domain
concerns as a first class citizen.

While the DSL and the meta-model were evolved together, there is a
key distinction to make between the two: a DSL is an interface to the
meta-model and is used by humans. Where as, a meta-model is con-
cerned with the concepts and their relationships. This has a number
implications that were considered in our research 1) a meta-model may
contain a few core constructs but may have many different DSLs, 2) the
DSL needs concepts that make it easy for developers to understand
and to use, 3) a meta-model is a conceptual model where as a DSL is
a realisation coupled with a toolchain and 4) developers need to be in-
volved in the DSL creation process as their preferences will dictate the
language design decisions.

Our approach builds upon the earlier work done on WebDSL, a DSL
targeting data-intensive web applications [52, 168]. The developers of
WebDSL\(^1\) had three guiding principles for the development of their DSL:
Domain Analysis, Technology Driven and Iterative [53]. All three prin-
ciples were incorporated into our method. Our approach also expanded
upon these principles with the inclusion of two additional guiding prin-
ciples, Model Solutions and Deploy.

Model Solutions involves using our DSL to model multiple apps to en-
sure that we have captured features that are applicable to a broad range
of data-intensive apps. Data-intensive mobile apps are also readily
available due to the large number of apps provided by the app stores.
By modelling existing apps we were able to validate and update our
meta-model throughout the development process.

The Deploy principle focused on the evaluation of the DSL, and there-
fore the meta-model, in the correct context of use – professional de-
velopers used our DSL as part of the development of real world apps.

\(^1\)http://webdsl.org/
5.2 Our New Method for Meta-model Construction

Our method for developing a meta-model consists of three phases: Phase 1: Mining Screen Categories which focuses on analysing the UI composition and information density of apps, Phase 2: Analysing App Composition involved building a DSL and generating mobile apps, and Phase 3: Domain Expert Feedback involved a domain expert using our language to provide feedback on the concepts it exposed. Each of these phases takes a different viewpoint of the technical domain in identifying concepts for the meta-model: Phase 1 looks at data and the presentation of content, Phase 2 looks at the components that make up a mobile app as viewed by an end user such as lists, images etc. and implemented by a developer such as web services, caching, etc. and finally, Phase 3 treats the DSL as an interface to the meta-model and takes the viewpoint of an end user.

For Phase 1 and Phase 2 we wanted to select a representative sample of apps from a variety of application domains. To this end we selected 30 data-intensive apps from the Play Store ensuring that we selected apps from different categories and star rankings. We also ensured that we selected apps that had different types of UIs i.e. screens with lists and custom layouts. These apps were selected based on the key features of a Data-intensive app as described in Appendix A, namely they were Thin Data-driven Clients, Relied on Web Services, Context Aware, UI/UX Focused and used Data Storage. This was primarily to ensure that the apps were a representative sample from the mobile app technical domain. Each of the 30 apps selected are shown below in Table 5.1 and were selected from all apps available from the Play Store.

Each phase followed an iterative process that expanded upon the results of the previous phase. This transition moves from domain analy-
Table 5.1: The 30 apps selected for use in Phase 1 and Phase 2.

<table>
<thead>
<tr>
<th>Number</th>
<th>App</th>
<th>Play Store Id</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Prompa</td>
<td>net.prompa.production.release</td>
</tr>
<tr>
<td>2</td>
<td>TED</td>
<td>com.ted.android</td>
</tr>
<tr>
<td>3</td>
<td>Gmail</td>
<td>com.google.android.gm</td>
</tr>
<tr>
<td>4</td>
<td>UBank</td>
<td>com.ubank.internetbanking</td>
</tr>
<tr>
<td>5</td>
<td>Eventbrite</td>
<td>com.eventbrite.attendee</td>
</tr>
<tr>
<td>6</td>
<td>ABC</td>
<td>android.AbcApplication</td>
</tr>
<tr>
<td>7</td>
<td>Shopping List</td>
<td>com.shoppinglist</td>
</tr>
<tr>
<td>8</td>
<td>Australia Post</td>
<td>au.com.auspost.m</td>
</tr>
<tr>
<td>9</td>
<td>Westpac Banking</td>
<td>org.westpac.bank</td>
</tr>
<tr>
<td>10</td>
<td>Barcode Scanner</td>
<td>com.google.zxing.client.android</td>
</tr>
<tr>
<td>11</td>
<td>Twitter</td>
<td>com.twitter.android</td>
</tr>
<tr>
<td>12</td>
<td>Flipboard</td>
<td>flipboard.app</td>
</tr>
<tr>
<td>13</td>
<td>Google I/O</td>
<td>com.google.samples.apps.iosched</td>
</tr>
<tr>
<td>14</td>
<td>HN</td>
<td>com.manuelmaly.hn</td>
</tr>
<tr>
<td>15</td>
<td>Gumtree AU</td>
<td>com.ebay.gumtree.au</td>
</tr>
<tr>
<td>16</td>
<td>Evernote</td>
<td>com.evernote</td>
</tr>
<tr>
<td>17</td>
<td>IMDb</td>
<td>com.imdb.mobile</td>
</tr>
<tr>
<td>18</td>
<td>MovieBase</td>
<td>de.linuxwhatelse.android.moviebase</td>
</tr>
<tr>
<td>19</td>
<td>Cricket Australia</td>
<td>au.com.cricket</td>
</tr>
<tr>
<td>20</td>
<td>Smart Measure</td>
<td>kr.sira.measure</td>
</tr>
<tr>
<td>21</td>
<td>GeoQuiz</td>
<td>org.urban.android.quiz.geographyquiz</td>
</tr>
<tr>
<td>22</td>
<td>V/Line</td>
<td>com.vortilla.myline</td>
</tr>
<tr>
<td>23</td>
<td>Melbourne Train</td>
<td>com.tincan.traintimes</td>
</tr>
<tr>
<td>24</td>
<td>Facebook</td>
<td>Facebook</td>
</tr>
<tr>
<td>25</td>
<td>WhatsApp Message</td>
<td>com.whatsapp</td>
</tr>
<tr>
<td>26</td>
<td>YOW! 2012</td>
<td>com.confui</td>
</tr>
<tr>
<td>27</td>
<td>Coles App</td>
<td>com.coles.android.shopmate</td>
</tr>
<tr>
<td>28</td>
<td>eBay</td>
<td>com.ebay.mobile</td>
</tr>
<tr>
<td>29</td>
<td>Woolsworths</td>
<td>com.woolworths</td>
</tr>
<tr>
<td>30</td>
<td>NYTimes</td>
<td>com.nytimes.android</td>
</tr>
</tbody>
</table>
sis to evaluation of the DSL with developers. Although these iterations are presented in a linear fashion, it was necessary to overlap these iterations at various stages of development and feedback from phases informed previous and next phases as well as future iterations. For example, informal evaluation of the DSL occurred through conversations with professional developers which began after the tool implementation. All of the iterations relied on using existing apps in some manner either to inform the meta-model or to evaluate the concepts. Each of the iterative processes used to construct the meta-model are described below and worked examples are included in Appendix F.

5.2.1 Phase 1: Mining Screen Categories

Purpose

Mining Screen Categories was the process we followed in order to identify abstractions based on the information density for a screen. This was done so that we could focus on the data aspect of data-intensive apps and to identify concepts that provided the highest level of abstraction. This also allows for the analysis of mobile apps from an end user perspective to get an insight into what they expected from an app.

Process

Outlined below are the major steps used for this phase of developing our meta-model and are shown in Figure5.1.
1. Collect Data-intensive Mobile Apps. For the first step we focused on collecting screens for data-intensive mobile apps as described above.

2. Analyse Information Density for a Screen. Information Density is defined as “The criterion information density concerns the users’ workload from a perceptual and cognitive point of view with regard to the whole set of information presented to the users rather than each individual element or item” [169]. Based on this definition we defined two main questions to answer about a screen:

   • What task is the user intended to do on that screen? For example, view a summary of results for a cricket match.

   • How is the information as a whole presented on the screen? For example, the information is presented as a list.

These two questions help us to classify the first few screens that we analysed. These initial classifications were combined with the UI guide-
Table 5.2: Baseline Classification Framework derived from Google and Apple’s UI guidelines and initial apps.

<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Splash</td>
<td>A screen first shown to the user when they open the app often just displaying the company logo.</td>
</tr>
<tr>
<td>Cards</td>
<td>A screen that shows a collection of information with individual content shown on cards.</td>
</tr>
<tr>
<td>Master</td>
<td>A screen that displays a list of related data.</td>
</tr>
<tr>
<td>Detail</td>
<td>The screen navigated to from a Master screen that displays information about a single item.</td>
</tr>
<tr>
<td>Map</td>
<td>Displays a single map to show the users location or plot a path.</td>
</tr>
<tr>
<td>Settings</td>
<td>Displays configuration options that the user can set that change the behaviour of the app.</td>
</tr>
<tr>
<td>Form</td>
<td>Collection of input fields that the user has to fill out.</td>
</tr>
</tbody>
</table>

UI guidelines provide by Google\(^2\) and Apple\(^3\) to develop a baseline classification framework shown in Table 5.2. The UI guidelines provided best practices for building an app and encouraged developers to create certain screen types. From this baseline we iteratively and incrementally evolved the framework by classifying additional screens.

3a. **Add New Category.** Not all of the screens we analysed could be classified and required modifications to the classification framework. Modifications were either adding an additional category based on the characteristics of the screen or updating the questions that were used to make a classification. Adding a new category required including new questions in the framework. Once the screen could be classified we moved to Classifying that Screen.

3b. **Classify Screen.** For each screen we first tried to make a classi-

\(^2\)https://www.google.com/design/spec/material-design/introduction.html  
fication using the framework. By doing this we could identify screens with similar information densities and gain an insight into screen types which are commonly used. After a screen has been classified we collected additional apps to continue the analysis step. This process continued until the classification framework ceased to be updated and every new screen could be classified.

**Outcome**

At the conclusion of Process 1 there were two main outcomes a set of categories for data-intensive mobile app screens and a framework for classifying screens. As we classified more screen types we came up with a set of questions that could be used to classify each new screen. Using this framework we can quickly classify a mobile app screen. At the very least the language needs to be able to describe these types of screens as this is what actual developers are building.

The screens were classified based on their information density. The information density of a mobile app screen is the amount of content that is displayed to the user and how it is presented. We chose to use information density for analysing apps as we wanted to focus on high level constructs that would improve developer productivity. The original framework proposed in Table 5.2 was improved throughout the process with the final outcome shown in Table 5.3 with details outlined below:

**Dashboard.** A Dashboard is a screen that shows a summary of data from multiple sources and is often the first screen shown when the app is opened. An example of a Dashboard screen is shown in Figure 5.2.
Table 5.3: The Refined Classification Framework derived from app analysis.

<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Splash</td>
<td>A screen first shown to the user when they open the app often just displaying the company logo.</td>
</tr>
<tr>
<td>Inform</td>
<td>Displays information in a single screen not part of a Master/Detail pattern.</td>
</tr>
<tr>
<td>Detail</td>
<td>The screen navigated to from a Collection, Complex Collection or Multi-Collection screen that displays information about a single item.</td>
</tr>
<tr>
<td>Form</td>
<td>Collection of input fields that the user has to fill out.</td>
</tr>
<tr>
<td>Dashboard</td>
<td>A screen that shows a summary of data to the user.</td>
</tr>
<tr>
<td>Collection</td>
<td>A screen that shows a collection of related data presented in the same manner.</td>
</tr>
<tr>
<td>Complex Collection</td>
<td>A collection made up of different row elements.</td>
</tr>
<tr>
<td>Multi-Collection</td>
<td>A screen that has more than one collection on the screen</td>
</tr>
</tbody>
</table>
Figure 5.2: An example of the Dashboard classification for mobile app screen.

Splash. Often mobile apps need to execute a long running operation when the app first loads such as fetching data from a webservice and a Splash screen is usually displayed to the user until the process is complete (See Figure 5.3). While this pattern can be considered an anti-pattern, as it prevents the user from using other aspects of the app, we have included it as we encountered this pattern frequently.

Figure 5.3: An example of the Splash classification for mobile app screen.

Collection. A Collection screen is a screen that is predominantly used to display a collection of data such as a list. An example collection is shown in Figure 5.4.

Complex Collection. Cards are a common feature of mobile apps where data is grouped together and displayed together as a single UI element such as a row in a list. A Complex Collection is a collection that is made up of different row elements. Consider, Figure 5.5, this screen consists of a Facebook link and a row about an upcoming event.
Chapter 5. A Method for Meta-model Construction

Figure 5.4: An example of the Collection classification for mobile app screen.

Figure 5.5: An example of the Complex Collection classification for mobile app screen.

**Multicollection.** A Multicollection is a screen that displays two more collections on the screen. These collections need not both be lists. For example, a Multicollection screen shown in Figure 5.6 could be a horizontal scrolling widget for showing scores of games (1) or show a list as seen in (2).

Figure 5.6: An example of the Multicollection classification for mobile app screen with two components a 1) Horizontal scrolling widget and 2) a vertical scrolling component.
**Detail.** The master/detail UI pattern [170] is commonly used in mobile apps which consists of a collection screen followed by a Detail screen that shows more information about the selected list item. An example of a Detail screen is shown in Figure 5.7.

![Figure 5.7: An example of the Detail classification for mobile app screen.](image)

**Inform.** Inform is a screen similar to the Detail screen except it displays data about a single concept – it is not navigated to by drilling down from a collection screen. An Inform screen can be seen in Figure 5.8.

![Figure 5.8: An example of the Inform classification for mobile app screen.](image)

**Form.** The final screen type is predominately used to capture input from the user and is known as a Form. (Figure 5.9).

We continued adding categories until no new categories were found and the screens from the apps could all be classified. In addition, we also had a collection of classified screens that could be analysed at a later stage. With these categories we now had an understanding of how infor-
information was presented to users and what the common types of screens were. Based on the framework and the screen categories in the previous section we have identified high level concepts for the meta-model.

5.2.2 Phase 2: Analysing App Composition

Purpose

Categories for mobile app screens provide us with the high level abstractions but do not tell us about the components that make up these screens. Phase 2: Analysing App Composition, was designed to help us uncover the elements that were used to build these screens both from an implementation perspective and from a user perspective. Now that we are analysing the implementation details we needed a way to evaluate the concepts that we captured. We achieved this by developing a tool that realised a DSL based on the meta-model which we used to generate data-intensive mobile apps.

Process

Each of the major steps for Phase 2: Analysing App Composition are described below and shown in Figure 5.10.
Chapter 5. A Method for Meta-model Construction

1. **Meta-model Construction.** The apps shown in Table 5.1 were carefully studied to identify common features that were not tied to the business domain of each app. We also analysed the screen categories that we identified in the previous phase to determine which components were commonly used to present the information. In addition, we analysed the relationships between these components, for example, the different ways users can navigate from screen to screen. This first version of the meta-model became the basis for our DSL.

2. **Building DSL.** Building a DSL on top of the meta-model was crucial for determining if the model could generate working apps. We focused on building the tool from end-to-end that is from DSL description to generating an Android app. By building an end-to-end solution we could test each aspect of our approach – developer evaluation was now possible. During this stage we concentrated on evaluating the meta-model rather than on language design as we wanted to ensure we captured the key concepts. Once the meta-model had the correct concepts and relationships we could allocate time to designing an appropriate language. We made sure that the generated apps worked and could be run on a device so that we could quickly compare the generated app with the actual app from the users perspective.
3. Generate Existing Apps. After the tool was built we began to generate existing apps to identify gaps in our meta-model. This approach showed us that basing our meta-model on the screen categories meant that a number of abstractions had to be expanded – the abstractions were too high-level and missed necessary details. The categories for the screen were sufficient for describing the information density but did not contain sufficient details to realise an implementation. These low-level details only came to our attention when we started implementing apps and realised that there were concepts that were needed for multiple screen categories.

By building apps with our tool we realised that different abstractions were needed in the implementation but were not required in the DSL. For example, the concept of a data model must be handled by the tool i.e. be part of the meta-model. However, data model abstractions do not need to be part of the DSL as the data model can be inferred from how data is used as expressed by the DSL. In this manner the DSL was an interface to the meta-model only exposing what was needed to describe an app.

4. Evaluate Generated App. Evaluating the apps at this stage involved carefully analysing the features of the app that could not be generated. This was done by comparing a generated app with the actual app. From this evaluation two things became evident: 1) missing features from the DSL could still be supported by our meta-model and 2) DSL design and development although influenced by the meta-model needed a separate refinement process. During this process we wanted to focus on the meta-model, so missing features from the DSL were left as future work. We focused on the core task of modelling app features such as navigation, fetching data and rendering data to the screen.

Evaluating apps often required modifications to the meta-model and by building new apps we were able to identify the key features of data-intensive mobile apps. The tool and the language were then updated to reflect the changes in the meta-model which enabled us to develop new apps that had these missing features. We continued this process until no new concepts were added to the meta-model.
Outcome

At the conclusion of this phase we had a meta-model for data-intensive mobile apps and an implementation of our DSL. This language acted as an interface to the meta-model and focused on modelling these concepts rather than on the needs of a developer. Our implementation of the DSL could generate working mobile apps that had a range of different features and could be used as the basis for developing new apps. While the tool could support the generation of mobile apps it still needed ongoing development to make it robust and support the wide range of functionality required for developing mobile apps.

5.2.3 Phase 3: Domain Expert Feedback

Purpose

The final phase for constructing a meta-model is to ensure that the concepts present can be used by a professional developer. Developers need to understand the constructs in the meta-model and be able to describe data-intensive apps with the DSL.

Process

Phase 3 focused on getting professional developers to use our tool and provide feedback. The steps in the process is displayed in Figure 5.11 and will be described in more detail below.
Figure 5.11: The process for evaluating the meta-model and DSL with professional mobile app developers.

**Developer.** Our DSL targets usage by professional app developers so the final phase of the construction of our meta-model involved getting developers to use the tool to generate apps. The reason behind this is that the concepts in the meta-model, and thus the DSL, should map with concepts that a developer is familiar with. Phase 3 is also designed to refine the language by developing the syntax for practitioners.

**Feedback.** Developers who use the language often commented on limitations of the implementation or desired language features. Most of these concerns were not related to the meta-model and they could build simple apps with the concepts we provided.

**Evaluation.** For our evaluation of the process we got 20 senior software developers who had at least 3 years experience developing mobile apps to use our DSL (See Chapter 6 for full details). Developers had to use the tool to build a simple app then provide feedback on problems they faced, recommendations for enhancing the tool or limitations to the language. Based on this feedback we decided if the changes were required for the meta-model or were for future work. Changes to the meta-model were made and the implementation updated to reflect these changes.
Outcome

After Phase 3 we had a language that was usable by professional app developers for building data-intensive mobile apps. We also had a robust meta-model that contained the core concepts for describing data-intensive mobile apps.

5.3 Mobile App Meta-model

In this section we describe the mobile app meta-model that was developed by following our method for meta-model construction. The relationship between the meta-model and the technical domain concerns is shown in Figure 5.12. Both the non-functional requirements and the domain influence the technical domain concerns. The meta-model is influenced by the domain but is used to address the technical domain concerns and the functional requirements of the app. The full meta-model can be seen in Figure 5.13 which shows the core concepts and the relationships between them. Each of the concepts will be described below.

App. The App concept represents the core features that influence the entire mobile app being modelled. This concept models information such as the starting screen and configures third party frameworks during the app loading stage.

Theme. Each mobile app has a theme that influences the entire presentation of the mobile app. This theme is based on the style guidelines outlined by each of the target platforms and includes the key concepts for the app. Information such as the colour palette, font styles and app icon are all captured by the concept of the Theme. A Theme consists of a number of Styles.

Style. Where as the Theme describes the presentation for the entire app a Style describes how a single element is displayed. For example, there may be different styles for the font that display a user error to the font that is used to display general information. In a large mobile app
UI elements will have a single style that encourages reuse throughout the app and ensures a consistent UX.

View. A View represents the visual components of a mobile app – the elements that the user interacts with. Often Views consist of other smaller views. For example a list contains rows that hold text fields, buttons and images. Every data-intensive app must have at least one View to display to the user. Typically, users interact with the Views through a touch screen interface that triggers Events.

Event. An Event is triggered in response to user interaction, an action from the system or by a sensor. A user triggers different events depending on the action they last performed that is clicking on a button will fire a different event to double clicking the same button. The Event concept is used to model all of the different touch interactions a user may perform using a mobile device. After an Event is fired a number of instructions are performed.

Instruction. An Instruction represents something that the mobile device
needs to execute. There are a wide range of different instructions. Examples include: navigating to another screen, displaying a popup and making a phone call. These instructions are essential for performing the use cases required for the mobile app. Not all instructions can be run from an Event as some of them may block the user interface and need to be offloaded to a background Service.

**Service.** Mobile devices also include background components known as services that perform tasks without updating the user interface. These tasks are typically performed to overcome the single UI threaded technical concern by offloading long running tasks to a background service. Many different Instructions may be run from a background service the most common is retrieving data from either a database or a web service.

**Sensor.** Sensor captures all of the concepts relating to the use of a mobile device’s sensors. A sensor can be used to fire off continuous Events. For example using the GPS to track where a person is walking,
used to trigger one off events such as using Near-Field Communication to share data with another device or be used to fetch current data such as the users context.

Data. Central to data-intensive mobile apps is the concept of Data. Data is operated on by an instruction and may involve, formatting, retrieving, saving, validating, updating etc.

Storage. Data storage concerns are modelled by the concept of Storage. This concept includes on device storage such as local storage, databases or the use of SD cards.

Model. The model is the structure used to describe the data that is used in the mobile app.

5.4 Summary

This chapter describes our method of empirically creating a meta-model to describe data-intensive mobile apps. Our method of analysing information density and screen composition, studying implementation details and incorporating developer feedback ensures that we consider all facets of the technical domain when constructing a meta-model. As a consequence our method is repeatable for other technical domains and provides a way to iteratively evaluate the meta-model.

First, we discussed how the meta-model and a tool implementation need to be iteratively developed together. Second, we presented our method for analysing the mobile app domain and identifying core concepts for the meta-model. Our method is based primarily on the analysis of a set of apps to identify the interaction model. It also included expert feedback, related literature and a tool implementation. Finally, we presented our meta-model for data-intensive mobile apps.

The core contributions of this chapter are as follows:

- A meta-model consisting of the core abstractions of data-intensive
mobile apps.

• An empirical based method for developing a meta-model for data-intensive mobile apps.

Minor contribution of this chapter include:

• Categories for classifying data-intensive mobile apps.
Chapter 6

Rapid APP Tool (RAPPT)

In this chapter we seek to improve our understanding of how the implementation details influence software architecture by building a tool based on domain specific knowledge. We expand upon the meta-model work presented in the previous chapter by creating a tool for automation. Techniques from the area of MDD are ideal for applying architectural knowledge but lack guidance for how to go from a meta-model to code generation.

Specifically we want to answer the following question How can the technical domain influence the design and implementation of tools for automation? We achieved this by developing RAPPT a prototype tool for realising architectural scaffolding for new apps. We look at how the technical domain influences the UI/UX, the internal model transformations and the design of code templates.

RAPPT leverages model driven techniques to bootstrap mobile app development through the use of a Domain Specific Textual Language (DSTL) and a Domain Specific Visual Language (DSVL). It provides multiple views to developers ranging from detailed view (code) to abstract view (e.g. page navigation). Multiple views help developers overcome the limitation of model driven approaches by enabling them to specify and edit app details at different levels of abstraction. These views also improve communication between stakeholders by presenting a snapshot of the
application for a specific concern.

This chapter is organised as follows: Section 6.2 describes the influence the technical domain has on an MDD based tool, the influence of the technical domain on the model transformations are covered in Section 6.3 and we finish up with the Architecture and Implementation of RAPPT in Section 6.4. An overview of RAPPT’s interface is included in Appendix C for completeness. We conclude with a summary in Section 6.5.

6.1 Motivation

As a way to motivate the requirements for a new tool, we build a fictitious MovieDB App. Limitations with the current approaches are covered in Chapter 2.

Consider Peter, a mobile application developer, who is tasked with the development of the MovieDB app. MovieDB app shows a list of popular movies that, when selected, enable a user to navigate to a screen showing the details of that movie. This app consists of three screens a Popular Movies screen for displaying the list of popular movies, an About screen to display copyright information, and a Movie Detail screen to display the details of a selected movie. Screenshots for these screens are shown in Figure 6.1. The content to be displayed by this app will be provided by the freely accessible MovieDB API.¹ Peter’s app includes the following requirements:

1. A tabbar for navigating between the Popular Movies and About screens. Tabbar is a Mobile UI Pattern that displays tabs near the top of the screen for navigation (see Figure 6.1). The user can also swipe the screen to navigate between tabs.

2. Authentication with the Movie DB API e.g specify an authentication key.

¹http://docs.themoviedb.apiary.io/
Figure 6.1: The Popular Movies, About, and Details screens for MovieDB, an app based on the MovieDB API.

3. Display a list of the popular movies from the Movie DB API on the Popular Movies screen.

4. Navigate to the Details screen from the Popular Movies screen by selecting a list item.

5. Pass the identifier for a movie when clicking on a list item in the Popular Movies screen and pass it to the Details screen so the details for the correct movie can be fetched from the MovieDB API.

6. Fetch the details for a selected movie from the MovieDB API and render the results to the screen on the Details screen.

7. Display a copyright message on the About screen.

To build the app using conventional Android app development tools, Peter needs to write the code for the tabbar specifying the tabs for each screen, configure the navigation for each tab, handle navigation to the tab screen and create the animation for swiping between tabs. In order to connect to the MovieDB API and handle authentication, he must write code for the API requests, model the data returned by API calls, provide error handling, check network connectivity, authenticate with
the MovieDB API and ensure best practices for concurrency on a mobile platform. For the Popular Movies screen, Peter needs to create the following: layout files for the UI, connect the data returned from the API with the UI components, write event handlers for selecting a list item to navigate to the Details screen, and implement the navigation pattern tabbar. The Details screen needs to accept the parameters passed from the Popular Movies screen, pass that parameter to an API call to fetch that data for the selected movie and render the movie details to the screen. The About screen needs to display a static message to the user. In addition to these tasks Peter needs to configure the build system, add dependencies, add logging code, create styles, debug, follow software engineering best practices such as ensuring maintainability and performance, while meeting stringent deadlines. To capture this need we have our first key requirement:

- **R1.** Automate generation of the boilerplate code required for data-intensive mobile apps.

Many current app generation tools are aimed at inexperienced developers or non-technical experts [46, 56, 171–173]. These approaches hide many of the implementation details described above from the end users enabling them to focus on higher level constructs. While this makes programming palatable, end users cannot customise how these apps address the technical domain concerns (See Chapter 4) of mobile apps [27, 72, 174]. For example, Peter has specific requirements about where to fetch data from (i.e. the MovieDB API) and how to authenticate the app with the API using an API key. As such these approaches are not suited for use by a professional mobile app developer. We capture these concerns in the following key requirement:

- **R2.** Design a tool for professional mobile app developers.

A number of Model Driven Development (MDD) approaches for building mobile apps have been developed [42, 44, 47, 50, 175]. These approaches commonly focus on MDD as a cross-platform approach to mobile app
development. A common emphasis of these cross-platform approaches is on modeling a common subset of features that can then generate code for multiple target platforms. The modeling approaches often present a single way of modeling a mobile app and do not provide multiple views of a mobile app as in our proposed solution. For example, describing every aspect of an app by using a DSL [42]. In addition, these MDD approaches tradeoff flexibility for productivity by abstracting away many of the implementation details. Data-intensive mobile apps share a lot of functionality yet have very unique UI elements and interactions. To support building unique mobile apps a tool should:

- R3. Generate apps that provide the full capabilities of the underlying platform i.e. provide flexibility in the generated apps.

Typically the app development process begins by designing the UI and User Experience with the client before these ideas are refined into mockups by a designer. Tools for prototyping [176–178] can greatly help with the evaluation of ideas especially concerning the navigation flow through an app. Once this process has been completed developers have to start from scratch to implement the agreed upon navigation flow – current tools produce throwaway prototypes. In addition, implementing a prototype that can be reused by developers is a time intensive and costly process. What is needed is a tool that can be used for rapid prototyping but generates apps that can be built upon by developers when realizing the final app. We have broken these needs into two key requirements:

- R4. Enable rapid prototyping of a fully functioning mobile app.
- R5. Produce prototypes that can be refined into the final app.

User Experience plays a crucial role in the development of mobile apps due to the influence of user reviews and rankings of the app store [25]. Designers and developers need to pay careful attention to the usability of their apps. Modeling different components of an app such as the navigation flow is well suited to and frequently represented as a graphical
visualisation. Developers often spend a lot of time programming using text based languages and is a familiar interaction model for describing event handlers, data flows and UI bindings. To capture this concern we have our final requirement:

- R6. Provide multiple abstraction levels for modeling the different concerns of a mobile app i.e. navigation flow and UI composition.

### 6.2 Influence of the Technical Domain

With the requirements for a new tool clearly outlined we are ready to explore how the technical domain influences the tool's implementation. From our analysis of mobile app composition used in the creation of our meta-model and analysis of the technical domain we discovered that there are **Multiple Abstraction Levels in Mobile Apps**. These levels are important to understand when designing MDD based tools as this determines which abstractions are added to the tool.

We also used our technical domain model to consider the wide range of **Tool Stakeholders** who would use MDD based tools. This includes developers who would use the tool and those who would use the generated output. After considering the different stakeholders we realised that there are **Multiple Views of a Meta-model** which require a different subset of the meta-model.

#### 6.2.1 Multiple Abstraction Levels in Mobile Apps

The technical domain capture the concerns that arise during the implementation of a system such as Ecosystem which includes the underlying platform that is used to build a mobile app such as Android or iOS (see Chapter 4). These platforms influence the way the mobile app is developed and provide APIs for developers. App development involves different levels of abstractions that build upon the previous level and allow developers to reason about different aspects of a mobile app. Building tools at the appropriate level of abstraction requires making
trade-offs between flexibility and productivity as the higher levels are harder to modify – the more abstract the concept the less details can be specified. Each level of abstraction is discussed below from the lowest level to the highest and are necessary for addressing the technical domain concerns.

• **Low Level Platform APIs.** This is the lowest level of abstraction provided by a mobile platform. Being the lowest level, it permits the maximum flexibility as developers can modify every aspect of the platform. Typically the rest of the platform’s APIs are implemented using these low level APIs. Most apps do not need to use these APIs but they are available if the developer has to add a very specific and unique feature.

• **Base Components and APIs.** Platform vendors provide components from which app developers build their apps such as buttons, classes for creating screens and fetching data from webservises or for interacting with platform features such as sensors. The majority of a mobile app is currently developed at this level of abstraction as it strikes the appropriate trade-off between flexibility and productivity.

• **High Level App Concepts.** These are the concepts that non-developers can reason about or end users can experience. Implementing these concepts needs significant development work, requiring the use of multiple lower level APIs. Example concepts at this level are screen, call to webservice, maps etc. The concepts at this level can often be mapped across platforms.

• **App Concerns.** This is the highest level of abstraction and is focused on viewing and manipulating app wide concerns such as navigation flow and data modeling. UML based approaches to mobile app development often work at this level of abstraction. While these approaches are ideal for modeling high level concepts adding the necessary lower level details required for producing unique mobile apps can be challenging – an intrinsic limitation of model driven approaches.
While these abstractions have been split into four distinct levels, when developing mobile apps these levels get blurred. In addition, mobile platform vendors frequently update their design guidelines but may not update their APIs at the same time requiring developers to build custom components from scratch. Consider the example of using a UML tool (App Concerns) [179] to model a data-intensive mobile app which includes a new UI navigation pattern such as a Drawer. Implementing a Drawer requires using High Level App Concept of a list, Base Components and APIs for displaying text and images, and Low Level Platform APIs to handle custom animation and styles. Developers cannot always wait for the platform to implement a feature they need or for a 3rd party library to become available. Modeling approaches that cannot support this use case provide limited utility to mobile app developers who have to face a rapidly changing environment.

6.2.2 Tool Stakeholders

We also used our technical domain model to consider the wide range of Tool Stakeholders who would use a new MDD based tool. All software have stakeholders who influence how the software is built and the features it should have. Using the technical domain as a frame of reference to view the stakeholders for a new MDD based tool exposed three groups of stakeholders: Modellers, Recipients and Implementors. Modellers are concerned with using the tool to generate artefacts or to reason about a software engineering problem. They interact with the tool through an interface such as a textual DSL or a visual modelling language. A Recipient benefits from the artefacts created by the modellers. End users that provide feedback upon a generated prototype and a developer who has to work with the generated code are two examples of a benefactor. An Implementor is concerned with ensuring that the modelling tool can be developed and that captures the domain so that the modellers and benefactors have the appropriate abstractions. While these are three distinct groups a single individual may fulfil each group i.e. a developer builds a DSL for herself to simplify her own work.
Chapter 6. Rapid APP Tool (RAPPT)

Modellers

A developer is one of the most obvious stakeholders for an MDD based tool that generates source code. They need the tool to have the correct abstractions to model the requirements and specifications for new mobile apps. Developers also need appropriate abstractions that map to concepts in the target platform so that they can easily identify what parts of the mobile app will be generated. Finally the developer is concerned with completeness and the relationships between the concepts exposed to the user of the MDD based tool. This is to ensure that the abstractions that are exposed to the user map to the correct relationships required for building mobile apps.

Another aspect of a Modeller is that of the Data Modeller who is primarily concerned with how the data is stored, saved, updated, retrieved etc. Due to the domain concerns of network connectivity and the heavy use of web services in data-intensive apps, caching is very important. As such Data Modellers need to have access to concepts that let them tweak the caching strategy and that more of the implementation details are exposed to the user of the MDD based tool. A Data Modeller is also concerned with the structure of the data and where it will be stored. UI and UX are crucial factors to the success of an app and are the primary concerns of the Designer Modeller. This is primarily focused on the look and feel of the mobile app. They are concerned with how the concerns of the end user of the app will be catered for in the design of the app and how the tool will allow them to realise their designs.

MDD based tools are typically used by people but they can also be used by other tools such as project template generators being used by Integrated Development Environments. To capture this concept we come up with the Tool Modeller. In this situation, the output of another program may be passed to the input of our MDD based tool to generate a working mobile app. For example, a tool that exposed abstractions suitable for high school students may want to use our MDD tool as the generation blackened allowing the creators to focus on the design of the interface and abstractions. A Tool Modeller is concerned with how easy it is to integrate with other tools and if there is a standard input format.
Recipients

The output from our MDD based tool is ultimately going to benefit the mobile app End User. Although, End Users do not care how the app is developed they do care about the UI and UX of the final app as well as how well their use cases are satisfied. As such it is important to ensure that the tool can be used to describe solutions to problems that real end users have.

Developers are also another main Recipient of MDD based tool as they will not have to write so much code from scratch. They are concerned with the quality of the generated output. Quality of the generated code is important as our code needs to be designed to be modified as the developer will have access to all aspects of the code – none of the code will be abstracted away. While this makes round trip engineering nigh on impossible it provides developers with maximum flexibility as they are not restricted in anyway. The code that gets generated resembles that written by a professional developer to ensure that it is easy to read and maintain. Another aspect of the generated code is that it is generally applicable. Many different types of apps from many different business domains will be generated from our MDD tool so the code needs be suited to a range of situations. This may require the tool to provide abstractions that customise the code templates in some manner.

Implementors

Implements are the tool builders of the MDD based tools. In a previous chapter (Chapter 5) we presented how to design a language and a meta-model based on the technical domain. This is a crucial step for the Modellers who are concerned with ensuring the correct concepts are present in the model. Their job is to ensure that all the concepts for each of the Modellers is present and can adequately express solutions to the problems each Modeller faces. The Model Designer needs to be wary of the relationships between the concepts and that different Modellers may have conflicting requirements. As such, trade-offs may need to be made and there is the possibility that multiple interfaces
or tools will be needed later on. For example, consider a Designer who wants a drag and drop based tool that enables them to customise the UI compared with a developer who wants clean abstractions for handling concurrency.

An Implementor is the Architect who is concerned with how the code templates will evolve over time, what the architecture of the generated output will be and how the meta-model will map to the concepts in the architecture. These decisions involve having a thorough understanding of the domain as well as a solid grasp of software engineering principles. These implementation decisions made by the Architect Implementor influence the utility provided to the Recipient and influences how the Modellers describe their solutions.

### 6.2.3 Multiple Views of a Meta-model

Each of the stakeholders mentioned above influences the design of our MDD based tool in some manner and by extension how the meta-model is viewed, manipulated and extended. While the concepts in the meta-model for each of the stakeholders may be slightly different there will be a core set that will be the same – each of the stakeholders is still concerned with data-intensive mobile apps and addressing the technical domain concerns. A better way of considering the impact each stakeholder has on the meta-model is by considering a view of the meta-model. A view represents a window of the meta-model from a particular viewpoint. This is analogous to the concept of View and Viewpoint from software architecture [8, 102, 107] except the object of analysis is the meta-model rather than software architecture.

Another distinction between a meta-model view and an architecture view is that a meta-model view may also include a view of the output from using a tool that is based on the meta-model i.e. display the generated code. As discussed in Chapter 5, a meta-model and the tool that realises the model are intricately linked so a view that represents the output is part of our view hierarchy.
Each viewpoint may have multiple views of the meta-model or a view may be of interest to multiple viewpoints. A viewpoint is a stakeholders view of the meta-model typically showing the concepts and relationships that are of interest to them in a manner that is relevant. For example, a Designer would prefer a visual representation while a developer may opt for a textual representation of their respective views. There are a number of different view types that will show different aspects of the meta-model. The hierarchy of view types are shown in Figure 6.2.

![Figure 6.2: The different view types of a meta-model all of which extend the concept of a view.](image)

- **Full View** A Full View contains all of the information present in the meta-model. This view can be used for synchronisation between the various partial views. For systems as complex as mobile apps the full view is often based on a textual based language as the relationships are complex and not suited to being displayed visually. Although a meta-model that has a few number of concepts and relationships can still be represented visually. A full view is typically
the master representation of the model being constructed typically in XML, JSON or another tree-like data structure. In the context of the technical domain concerns for mobile apps, discussed in Chapter 4, a full view can aid in understanding how all of the domain concerns influence one another.

- **Partial View** A Partial View only contains information on a particular aspect of the meta-model. Multiple Partial Views in combination may contain all of the information present in a Full View but runs the risk of containing redundant information. A Designer is not concerned with how data is shown to the user just that it looks good and is correctly formatted. In a Partial View targeting a Designer the information for how the data is fetched would be omitted but constructs would be included for describing what the data would look like i.e. style, format, structure, size etc. There are two types of Partial Views: Detailed View and Summary View.

  - **Detailed View** A detailed view contains all of the information about one aspect of the full meta-model. An example of this is a Screen view that contains all of the information for an individual screen but not all of the screens like the full model. Due to the high degree of detail shown, these types of views are of use to the Modellers. To address the technical concern of Lifecycle, a detailed view of a screen can show all of the interactions at each stage of the Lifecycle. This allows the developer to reason about the different stages and identify problems.

  - **Summary View** A Summary View contains the cross cutting information of a particular viewpoint. An example of a Summary View is a Navigation View that contains all of the screens and the links between them but does not contain any information about the theme of the app. Summary Views can be useful to all stakeholders as it represents crosscutting concerns that influence many parts of the system. A Summary View can assist in dealing with the technical domain concerns of Hardware Constraints and Network Connectivity. Hardware Constraints such as battery usage require analysis of all parts
of an app to identify areas that drain resources significantly. In the case of Network Connectivity, all services that access a network are affected and different approaches may be used depending on the desired functionality but should be analysed as a whole.

- **Output View** Output View is a view of the output of the model transformation process. Often this view is used to show source code of the generated output or the structure of the output for consumption by an additional tool. These types of view are aimed at the Recipient stakeholders. The output view of a meta-model can help developers learn how to address the technical domain concerns by generating best practices. Developers can interact with the core concepts of a mobile app and then see how the technical domain concerns are addressed.

Depending on the scope of an MDD based tool multiple views of the meta-model may need to be exposed for different users or use cases. Modellers may switch between interfaces to manipulate the meta-model using the most appropriate view. However, supporting multiple views vastly increases the complexity of the tool as now these views need to be synchronised.

### 6.3 The RAPPT Approach

RAPPT is a tool that we built to validate our meta-model, satisfy the needs of stakeholders and provide different abstraction levels for developing mobile apps. Our approach is a tool that provides multiple views for specifying the concepts of a mobile app and generates working prototypes that form the scaffolding of the final app. The focus of our work has been on building a tool that targets professional app developers and assists them in the early stages of development by providing them with a Domain Specific Visual Language (DSVL) to specify high level app features and then a Domain Specific Textual Language (DSTL).

RAPPT begun as a simple Domain Specific Language used to evaluate
our meta-model as discussed in Chapter 5. As the tool developed we realised two separate languages were needed a DSVL and a DSTL, see Section 6.2.3.

After the specification has been completed RAPPT generates the source code for a single platform, a working Android app, to which the developer adds the final polish. The major steps involved in using RAPPT to generate a new project are shown in Figure 6.3. The core aspects of our approach are described below and include a Partial Model, Partial Synthesis, Multiple Views with Overlapping Abstraction Levels and Generate an App for a Single Platform.

**Figure 6.3:** RAPPT assists software developers by generating the initial architecture for their app.

### 6.3.1 Partial Model

MDD use models to abstract away implementation details that is well suited to address our requirement *R1. Automate the boilerplate code required for data-intensive mobile apps*. MDD based approaches abstract away some of the details allowing developers to focus on using higher
level abstractions. In practice, this trade-off between productivity and flexibility is not always feasible as users often need to modify the details that are abstracted away [180]. As such we opted for the use of a partial model that allows us to provide key abstractions for modelling the scaffolding and still permit developers to edit any part of the code after generation. In a previous chapter we discussed how this meta-model was constructed and analysed the technical domain concerns that satisfies our requirement: **R2. Design a tool aimed at professional mobile app developers.**

At an implementation level, mobile apps contain source code that address a number of concerns which are listed below. Where applicable we have mapped the concerns to constructs in the meta-model.

- **Styles and Theme** - This refers to the custom styles and theme applied to an app. Meta-model constructs: *Theme, View* and *Style*.
- **Business Logic** - Refers to the unique business logic of a mobile app.
- **Animations** - Visual animations such as sliding a screen in from the left hand side.
- **Interactions** - User initiated interactions such as navigating to a screen. Meta-model constructs: *Event* and *Instruction*.
- **Sensor Usage** - Accessing the devices sensors such as the GPS to locate a user’s position. Meta-model constructs: *Sensor, Service, Event* and *Instruction*.
- **Platform Integration** - Integration of frameworks that provide services such as ads, analytics etc.
- **User Interface** - Visual elements displayed to the user. Meta-model constructs: *View* and *Style*.
- **Data Access** - Access the data from a source such as a database. Meta-model constructs: *Storage, Model, Data* and *Instruction*.
- **Data Model** - Data representation used in the mobile app. Meta-model construct: *Model*
Table 6.1: The level of support for the major components of a mobile app in RAPPT’s DSL

<table>
<thead>
<tr>
<th>Parts of a mobile app</th>
<th>No support</th>
<th>Partial Support</th>
<th>Full support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business Logic</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Animations</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Styles and Theme</td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Interactions</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sensor Usage</td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Platform Integration</td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>User Interface</td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Data Access</td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Data Model</td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Network Access</td>
<td></td>
<td></td>
<td>x</td>
</tr>
</tbody>
</table>

• Network Access - Send and receive data on the network. Meta-model constructs: Instruction, Service, Data, Model and Event.

The support for each concept provided by RAPPT is shown in Table 6.1.

The parts of a mobile app that are not supported by RAPPT vary greatly from app to app. For example, no two apps make use of exactly the same business logic as it depends upon business processes and workflows that vary from company to company and animations are only limited by a designer’s creativity. This high degree of variability make these concepts unsuitable for abstracting away as the details are needed for an implementation. Adding this degree of flexibility to the tool will start to replicate the low level functionality provided by the mobile app Software Development Kits. In our approach, developers would manually add these features to the mobile app after the rest of the app had been generated. This approach provides developers with fine grain control over the app specific features and still provides the benefits of automation.

The partially supported components are where the overlap between hand written and synthesised code takes place. Initially the code for these components can be generated but are going to need manual modifications to tailor them to the specific app. For example, RAPPT gen-
erates a User Interface containing the basic UI elements (e.g. labels, buttons, input fields etc.) and connects them to the appropriate controllers and data elements. However, layouts and styles are added by the developer after generation as these are unique for each app and thus not suited for generation. These UI elements are necessary for showing information retrieved from a web service which influences part of the app architecture as it shows how data is passed through an app to the screen. This is in contrast to the layout of the elements which is a usability concern that does not influence the software architecture.

Concepts from the meta-model such as Style and Theme provide a way to markup the generated code for further development. In this way, developers can specify things that should be styled in a similar manner but the actual styles themselves have to be created manually in the generated code.

Finally, the full supported components can be expressed by the high level abstractions in the DSL and should not need to be modified in the synthesised code. These aspects of a mobile app are core to the way app functionality is specified using the DSL, however they cannot be bundled into a library or otherwise hidden from a developer as they need to be extended in the future. For example, additional data models may need to be added to the app as the functionality grows.

6.3.2 Generate an App for a Single Platform

One of the key motivations of our approach is the requirement R3. Generate apps that provide the full capabilities of the underlying platform i.e. provide flexibility in the generated apps. To achieve this we used Partial Synthesis which refers to the process of generating the core software artefacts, pertaining to software architecture. Synthesised code represents a complete app – the app compiles and can be deployed to a device. This distinction is important as the developers are intended to modify and extend the synthesised code in order to finish the app. As a consequence, the generated code must be of a suitable quality to ensure that developers can get started with the modifications quickly. Thus, we
have strived to synthesise high quality code by producing code that resembles that written by a professional, made use of popular libraries, implemented well established design patterns, followed platform specific UI guidelines and adhered to software engineering best practices such as separation of concerns. We also focused on generating code for a single platform which meant we did not need to handle discrepancies between platforms and could generate code that adhered to the platform’s UI guidelines.

This was achieved through one way generation from model to code and only using well known 3rd party libraries in the generated output i.e. popular Android libraries on Github. We decided to use one way generation for the following reasons:

- We strove to ensure that the generated code resembled code written by a professional with no unnecessary artefacts for round trip engineering. This was based on the idea that developers will maintain the code not programs so the code needs to be readable and understandable.

- One way generation ensured that we could keep the number of concepts in the meta-model to a minimum as we only needed to focus on concepts used to express the mobile app at a high level. Concepts that captured low level details need to be added to the meta-model to ensure that round trip engineering was possible.

- RAPPT was designed to be a bootstrapping tool to enable developers to quickly get started without any setup and the generated apps could be imported directly into existing IDEs.

We designed our code generator to produce mobile apps that could be compiled and deployed to a device without modification satisfying the requirement of R4. Enable rapid prototyping of a fully functioning mobile app. This enables developers to produce the first prototype quickly and can gather feedback on the navigation flow for the app during the initial client meeting. As mentioned above app development begins with the UI and UX and then moves onto the development tasks.
smooth transition from the prototyping stage and to ensure there is no wasted effort we designed the generated apps in a way that it forms the scaffolding for the final app. Once the initial prototyping stage was complete developers take the generated app and build the rest of the app on top of what was generated enabling RAPPT to satisfy requirement \( R5 \). Produce prototypes that can be refined into the final app.

### 6.3.3 Multiple Views with Overlapping Abstraction Levels

Software engineers utilize higher level abstractions to hide the unnecessary details and hence focus on the problems at hand. As such we needed to ensure that RAPPT could satisfy requirement \( R6 \). Provide multiple abstraction levels for modeling the different concerns of a mobile app i.e. navigation flow and UI composition. For each of the views discussed in Section 6.2.3, different abstractions are needed. These abstractions could conflict with each other. Navigation and UI layout views both need a screen but require different information. The navigation view is concerned with how a screen is connected to other screens whereas the UI layout view provides precise configuration of UI components.

Our approach provides three views for developing mobile apps consisting of overlapping levels of abstraction to address these conflicting scenarios: A DSVL for high level app functionality, a DSTL for providing additional details not available in the DSVL and access to the target platform for creating custom app functionality. Figure 6.4 provides an overview of these abstraction levels, their composing elements, and their relations.

The DSVL includes concepts for modeling the high level concepts (App Concerns) of data-intensive apps such as navigation flow and the Data Model. Included in the DSVL are a number abstractions that are categorized as High Level App Concepts, and can be enhanced using the DSTL. For example, the concept of a screen is a High Level App Concepts that is present in the DSVL so that the navigation flow can be modeled
but the DSTL is required to specify what will be displayed on the screen or which webservices it calls. A summary of the concepts present in the Visual Language is shown in Table 6.2. The rational for choosing visual notations was to choose the representations that closely relate to the concepts in the meta model and the target platform (the Android operating system). Only a few concepts have been added to the Visual Language as our focus was on evaluating the concept of a multi-view approach to mobile app development, rather than a complete visual app builder.

The DSTL also includes abstractions from the Base Components category. These abstractions are added to enhance the specification for screens and to be able to model webservices for fetching data to render to the screen. Examples in the DSTL are input fields, images and keywords for specifying webservices. For highly custom features and concepts that cannot be modeled by any of the previous stages developers have to use the Low Level Platform APIs in with the generated code.

Figure 6.4: Levels of abstraction in a mobile app.
Table 6.2: Example visual elements that make up RAPPT’s Visual Language

<table>
<thead>
<tr>
<th>Concept</th>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Screen</td>
<td><img src="image" alt="Screen" /></td>
<td>Represents a screen displayed on a mobile device as seen by the end user.</td>
</tr>
<tr>
<td>Button Navigation</td>
<td><img src="image" alt="Navigation" /></td>
<td>Represents navigation from one screen to another by clicking on the UI component Button.</td>
</tr>
<tr>
<td>Map</td>
<td><img src="image" alt="Map" /></td>
<td>Displays a Google Map3</td>
</tr>
<tr>
<td>Tabbar</td>
<td><img src="image" alt="Tabbar" /></td>
<td>Represents the Mobile navigation UI pattern, Tabbar.</td>
</tr>
</tbody>
</table>

6.3.4 Partial Synthesis

Partial Synthesis [181] refers to the process of generating part of a system from a model. In a partial synthesised system decisions are needed to determine which parts of a system will be formally modelled and which parts will remain hand written. For example, Android Studio, an IDE for Android app development, has a getting started wizard when creating a new project. This wizard guides the developer through the main navigation type, project name and type of app before generating a project. Behind the wizard is a very limited model of an Android project which the user configures by completing the questions. The generated project is created though the process of partial synthesis i.e. that can be deployed to a device but does not represent the final product. Our approach uses partial synthesis and an enriched model of a mobile app by using a DSTL and DSVL to specify graph like components of the app that is navigation flow, data flow etc. This allows a more complete app to be generated than that from an IDE wizard.
These systems are also required to handle the issue of integrating synthesised code with hand written code ensuring that no modifications are lost. We take a different approach by stipulating that after the code has been synthesised the modelling stage is complete and the rest of the project requires hand written code to finish – we support forward engineering only. For mobile app development, where there are many different concerns from the technical domain that need to be addressed by an appropriate architecture, synthesising the initial architecture can provide a significant benefit, see Chapter 7.

Bagheri et al. describe Partial Synthesis as an approach that is designed to be used by professional developers who both “produce and consume concern-specific modeling notations” [12]. In our approach professional developers use the modelling notations through the DSL and interact with the synthesised code but they do not create the modelling notations. Abstractions are lifted from code prior to use by a practitioner.

6.4 Architecture and Implementation

6.4.1 Development Architectural View

An overview of the main components that make up RAPPT’s software architecture are shown in Figure 6.5. This shows the core components required to take in either the DSTL or DSVL and generate a working mobile app from the description. Each of the components and its relationship with other components will be described below.

**DSL Parsers.** As input, RAPPT accepts a description of a mobile app written in a DSL and then parses this description to create an App Model. There are two DSL Parsers in RAPPT one for the DSVL and one for the DSTL.

**App Model.** The App Model is a model of the app that needs to be built. This model is a concrete instance of our meta-model tailored for the app that needs to be generated. It is this single “source of truth” for the
app being constructed that enables developers to switch between the different views i.e. switch between editing the DSTL to using the visual language.

**Validators.** Providing developers with helpful error messages to guide them in the construction of a valid description of a mobile app is crucial to the success of RAPPT by practitioners. The Validators role is to validate the App Model to ensure that the valid app can be generated. These rules contain technical domain specific information that is required for building valid apps i.e. there should always be at least one screen and the first screen to show to the user needs to be specified.

**Intermediate Model.** As an intermediate step to generating a model of the target platform we needed to create an Intermediate Model. This model is primarily used to store values extracted from analysing the
App Model.

Transformer. It is the role of the Transformer to extract out information from the App Model required for building an app. This stage consists of a number of transformations that assist in the model transformation process. There is a crucial component that aides the Transformer in determine which transformations need to be run, the Instruction Builder.

Instruction Builder. Included in the description of the mobile app is a set of instructions that the developer wants to execute when certain events are handled. To determine which events need to be set up, how the events are handled and making sure the correct tasks get executed is the role of the Instruction Builder. It is also used to infer what the behaviour of the app is.

App Translator. After the rules have been applied to the App Model by the Transformer both the App Model and the Intermediate Model are passed to the App Translator. From these two models the App Translator constructs a model of the target platform through another set of model transformation steps. These transformations determine which code templates will need to run and what the values will be for that template.

Android Model. Our platform specific model is a model of an Android app. the Android Model. This model contains all of the code templates required for building a working mobile app. It is a code representation of an Android app complete with a project directory structure and build scripts.

Code Generator. The final component needed for generating mobile apps in RAPPT is the Code Generator. The Code Generator is responsible for executing the Android Model to produce a concrete Android project. Once the Code Generator has completed its task the Android app is ready to load into an IDE or deploy to a device.


6.4.2 Model Transformation

RAPPT aides developers by automating some of the decision making processes at the start of a new app. Many of these decisions are addressed through a sensible set of conventions that each of the generated projects adhere too. Some of these decisions are described below.

- A project structure for a mobile app follows platform specific conventions but only provides a guideline and does not specify how the entire app should be structured. Based on the components specified in RAPPT, an appropriate and consistent project structure is created to simplify maintenance of the generated code.

- Select the right platform APIs to handle performance issues and ensure that the user interface remains responsive. There are often multiple ways to construct an app using a platform’s API Professional app developers are familiar with the appropriate API’s required for creating performant apps but they still need to implement them for each app. An example for Android development would be the use of a View Holder pattern when creating a list of elements. When creating a list the View Holder pattern is used to reuse the data for each of the row elements so that the data does not need to be reloaded when the user scrolls rapidly through the list.

- Design the software architecture to be extensible for the future and integrate the other features of a mobile app – apps are more than a single screen showing a list of data. Quality requirements are achieved through an appropriate software architecture [1] which needs to be built into the system. RAPPT generates a robust architecture which allows developers to focus on the functionality rather than on architectural decisions.

These concerns are addressed by RAPPT through model transformations (by the Transformer and App Translator components), appropriately designed code templates and the flexibility to modify every aspect of the code. These transformations use standard model transformation
techniques from MDD literature [182]. RAPPT consists of over 70 transformations that undertake tasks such as format class names, specify files to generate, wire up dependencies, specify imports and configure the project. These transformations are ordered as they build upon the output of previous transformations.

We have displayed a small subset of the transformations in Figure 6.6: (1) The outline for a row is described using the DSTL, (2) this description is modelled as a row in the App Model, (3) from this modelled row there are multiple transformations that need to be run to extract all the required information only a subset are shown and (4) each transformation rule influences multiple code templates in multiple places in the generated output. Each aspect of the DSTL has to undergo this process and rules can conflict with each other that is a rule may specify a screen as the first screen to show to the user whereas another rule may indicate that that screen is part of a more complex interaction such as a tabbar. These rules need to be reconciled and handled by the transformation rules.

![Diagram of the transformation process](image)

**Figure 6.6:** An example of our approach used to describe a list item that displays data fetched from a webservice.

Transformations for a particular part of the model can depend on other transformations before being run. An example of a transformation required to synthesise a controller class for a row element in a list is shown in Figure 6.7. Transformations can operate on part or all of the Partial Model, in this example the transformation operates on a single node representing a row in a list (1). Some operations performed by a trans-
formation are internal to the class such as formatting the identifier for a private variable. In the situation where a transformation results in a file in the synthesised code a data object is created to hold data for the template generation stage (2). Creating a data object often depends on the results of previous templates in our example of creating a controller class we need to know the package name and output directory (3). Subsequent transformations can extend the controller class so we need to expose the data object to later transformations to add extra information (4). Once all of the transformations have been performed, data objects are matched with and then passed to code templates for code synthesis (5). Code templates for RAPPT define the UI patterns, design patterns, language idioms and architectural properties of the generated code.

Figure 6.7: Example transformation showing how a row controller class is synthesised from the Partial Model.

Model transformations are an important part of RAPPT and required careful engineering to ensure that they were applied in the correct order, compatible with one another and ensured a working mobile app once executed.
6.4.3 Deployment Architectural View

RAPPT adheres to a client-server architecture where the parsers for the DSTL and DSVL are split across server and client. The primary reason for this is that when interacting with the model in a browser, client side parsers respond faster which was ideal for the DSVL. At the time of the project there were no suitable client side parser generators to use for the creation of our DSTL so the parser for our textual language had to be hosted serverside. The deployment for RAPPT’s processors for the DSTL and DSVL can be seen in Figure 6.8.

![Diagram of RAPPT's key components](image)

**Figure 6.8:** The deployment of RAPPT’s key components.

**Client**

The Client is responsible for enabling the user to interact with a model of the app being built through either the DSTL or the DSVL. Our DSVL has a separate meta-model which contains information about the nodes, links, layout positions and presentation. This meta-model is updated in real time when the user interacts with the DSVL. Once the user switches to the DSTL or wants to generate an app control passes to the DSVL processor where the meta-model is transformed into an App Model. The DSVL processor is also responsible for transforming an
App Model into a DSVL meta-model without layout information. An automatic layout algorithm is employed to present the DSVL on the screen.

At the core of RAPPT is the App Model which contains all of the information about an app to generate our DSVL and our DSTL. From the App Model new languages and tools can interact with RAPPT which simplifies integrating RAPPT into an IDE. The App Model is passed between the client and server, and it is structured as a JSON object.

**Server**

On the Server, the Model Processor is responsible for synchronising the DSTL Instance with the App Model produced by the DSVL Processor. This is to ensure that changes in either the DSTL or the DSVL get reflected in the other model. Another important task of the Model Processor is to create an Android model which describes the details for generating an Android app. The App Generator takes this Android Model, code templates and creates a working mobile app. At the completion of this step is the source code that gets sent to the client so that the user can review what gets generated. Once the user approves of the code the generated project is made available to download.

The other major component on the Server is the DSTL Processor which is responsible for converting between the App Model and the DSTL. By making the DSTL editor client side users do not need to download, install or configure any software to get started, just navigate to the website. This enables users to rapidly prototype a new app and try multiple versions of their app before starting the full development process.

**6.4.4 Technology Choices**

RAPPT was built following an iterative and incremental process that involved building the internal processor, DSTL and DSVL in distinct phases. From the start we ensured that the model could be used to generate working apps so the internal processor, an early version of the DSTL and the code generator were developed together. This meant that
we could generate early versions of an app and modify the model using real life examples. Our first version of the DSTL was a JSON file that was a declarative language to minimise the cognitive load on developers. As the code generator matured we started to focus on improving the DSTL.

Generating real world mobile apps revealed to us that a pure declarative approach would not work for our DSTL. For example, we needed developers to be able to specify what would happen when events were triggered such as a button press. While we found that many interactions can be inferred from the code it is difficult for developers to understand what has been generated so we made sure that the details of event handlers were made explicit by the developer when writing the DSTL. During this phase we almost started to modify the syntax into a language that developers would be familiar with by taking inspiration from Groovy and the C family of languages (C#, Java, C, C++ etc.).

As the DSTL evolved we realised that the early phases of a mobile app development involved an iterative prototyping process which was typically done by designers. To facilitate this process we added a DSVL to RAPPT. Designers would come up with the designs for the app and then use the DSVL to realise as much of the app as they could. Then the developers would take over and flesh out the design of the app using the DSTL and by modifying the generated code.

RAPPT was designed from the start to be a web based editor to minimise friction users have with using the tool. The client editor was developed using the Angular\(^4\) web framework and based on web technologies such as HTML 5, CSS and Javascript available here.\(^5\) The DSVL Instance was implemented using the Javascript AVG library in D3 which includes algorithms for the auto-layout of graphs. Auto-layout for the DSVL was an important feature for RAPPT as developers may cause updates to the chart by editing the DSTL and these changes would need to be shown when switching to the DSVL editor. Instances of the DSTL are a string representation of the DSTL.

\(^4\)https://angularjs.org/
\(^5\)http://rappt.io/
The App Model was implemented as a JSON object and was shared between the server and the client. Using JSON as the model format meant that it could easily be sent to and from the server as many web apps use JSON as a data transmission format. In addition the JSON file format is easy to ready for both humans and machines which simplifies the debugging process. On the client side when the developer manipulated the DSVL, the DSVL Processor updated the JSON representation of the App Model. When the user switched between tabs this App Model was sent to the server and converted into the DSTL so that the developer could view the text representation.

The DSTL Processor was implemented using the ANTLR\textsuperscript{6} compiler which compiled the DSTL. Errors that were generated on the server where sent to the server and displayed to the user in the error pane. The generated Android app is made available to the client app so that the developer can download the source code. We implemented the server side code in Java and the code templates using String Template\textsuperscript{7}.

\section*{6.5 Summary}

In this chapter we answered the question \textit{How can the technical domain influence the design and implementation of tools for automation?} showing how the technical domain can influence the design and implementation of a MDD based tool in three areas: UI/UX, internal model representation and code generation. We also presented RAPPT a prototype for the generation of scaffolding for data-intensive mobile apps inspired by findings from the technical domain. Building RAPPT helped \textit{improve our understanding of how the implementation details influence software architecture} by \textit{1)} revealing technical domain specific abstractions and \textit{2)} showing that multiple views are needed for different stakeholders. In the next chapter we provide a comprehensive evaluation of RAPPT including user evaluations, developing existing apps and interviews.

\footnotesize\textsuperscript{6}http://www.antlr.org/  
\footnotesize\textsuperscript{7}http://www.stringtemplate.org/
Chapter 7

Evaluation and Discussion

In this chapter we evaluate our approach taken to improve our understanding of how the implementation details influence software architecture. Namely we evaluate our meta-model, the technical domain model and our prototype, RAPPT. We interviewed 10 developers and conducted 2 case studies on: the Google I/O app and the Malaysian Bus Tracker, to evaluate the technical domain model and domain tactics covered in Chapter 4. We built Multiple exemplar apps from different domains were built to evaluate the completeness of our meta-model presented in Chapter 5. We conducted an industry case study spanning 2 years to determine the affect of evolution on the generated code and to evaluate the output from our prototype, RAPPT (Chapter 6). A User Evaluation with practitioners was also conducted to evaluate the learnability of the concepts in the meta-model.

From our evaluations we conclude that the technical domain model captures relevant domain specific knowledge and the domain tactics capture well known solutions. However, domain tactics need concrete code examples to guide developers in an implementation. We also found that the meta-model contains a sufficient number of concepts for describing a wide range of apps and the code generated by RAPPT stands the duress of software evolution.

Each evaluation in this chapter addresses one of the three research
questions presented in Chapter 1. Below we described the evaluations aimed at each research question.

**RQ1. What are the concerns of the technical domain that influence architecture and how can this information be captured?**

We held interviews with 10 software engineers and conducted 2 case studies helped. We chose to use interviews so that we could identify what practitioners recognised as architectural influences. This also meant that we could assess if the information we captured would be of use to engineers. The case studies demonstrated how developers made tradeoffs between the technical domain model and the architectural choices they then had to make.

**RQ2. How do we empirically build a meta-model that contains the core abstractions a developer uses to build mobile apps?**

We built multiple exemplars and conducted an industry case study. The industry case study was run to ensure that the principles that the meta-model could be used in the development of a real app. By recreating multiple exemplars with RAPPT we showed that the concepts in the meta-model are widely applicable to data-intensive apps.

**RQ3. How does the technical domain influence the design and implementation of tools for automation?**

We created a meta-model and a tool implementation that was evaluated with 20 end users. We chose to evaluate our tool with end users to ensure that the concepts in the meta-model could be understood by a developer and that the tool was suited for use by professionals.

This chapter has the following sections: Section 7.1 presents the interviews that were designed to evaluate the technical domain model. We present our exemplar apps in Section 7.2 that were built to evaluate the meta-model followed by our User Evaluation with developers in Section 7.3. Section 7.4 describes two case studies that demonstrate how the technical domain is used to analyse software architecture. We conclude the chapter with Section 7.5 which presents an industry case
study, Prompa, that spans two years.

7.1 Interviews and Questionnaire: Evaluating the Technical Domain Model

In this section we evaluate the mobile app technical domain and domain tactics from Chapter 4. We used semi-structured interviews and a questionnaire to answer the question *RQ1. What are the concerns of the technical domain that influence architecture and how can this information be captured?* Our goal was to determine if the technical domain model benefited novice developers by capturing knowledge from experienced developers. We also wanted to assess whether the domain tactics reflected knowledge used by professional app developers.

The interviews were run as follows. Each participant was shown the concerns from the technical domain model and the problems associated with each concern. Then the participant was given the 18 domain tactics for mobile app development to read through. Following that the participant completed a questionnaire for each of the domain tactics followed by an informal interview about the problems and the tactics.

7.1.1 Preparation

Prior to running the interviews we developed an interview guide shown in Appendix D. By preparing an interview guide we made sure that the interviews were consistent and repeatable. All of the questions for the demographics and interviews are shown in Appendix D. Below we outline the major steps in our interview and briefly describe each one.

- **Pilot Studies.** We conducted pilot studies with 3 mobile app developers to refine the interview process.

- **Organisational and Individual Approval:** Before starting interviewing each participant we made sure to obtain approval from both the organisation, individual and to record the interview.
• **Show slide deck**: To introduce the research to the participants we created a slide deck shown as part of Appendix E. This slide deck communicated where the research was intended to go and where the interviews fitted in with the research goals.

• **Present Technical Domain Model**: At this point the participants were given the concepts in the technical domain model, the common problems and the domain tactics related to mobile app development.

• **Questionnaire**: After reading through the domain tactics the participants were asked to complete the demographics questionnaire and to complete the survey questions. Each domain tactic had 5 Likert scale questions and 1 open ended question to capture any additional information the participants would like to provide.

• **Interview**: The final step of this evaluation was to have an unstructured interview with the participant. Questions asked here were designed to provide feedback on the domain tactics as whole.

The core activities that we followed for completing the interview were adapted from Hove et al. [183]. One of the changes that we made was the inclusion of a questionnaire to obtain qualitative feedback, on each of our domain tactics.

**Questionnaire and Interview Design**

For each of the domain tactics we asked a set of questions designed to determine the suitability and benefit of the domain knowledge documented. The domain knowledge was assessed based on whether the appropriate: quality attributes, domain concerns and problem context were assigned to the tactic. These questions were on a 5 point Likert scale from strongly disagree to strongly agree and are shown below:

1. The quality attributes of this Domain Tactic are appropriate.
2. The Domain Concerns for this Domain Tactic are appropriate.
3. This Domain Tactic addresses the Problem Context that it has been assigned.

4. This Domain Tactic is useful for novice developers to know.

5. The Solution description provides enough details to provide a high level overview for creating a solution.

6. Other notes about this Domain Tactic.

We designed our interview questions to cover three key areas, the technical domain model, mobile development problems and domain tactic. Questions differed depending on whether the participant was an app developer. These questions were intended to augment the responses from the questionnaire and qualitatively assess the technical domain model from the perspective of a developer. They were split into questions about the Domain Conceptual Model and Mobile Development Problems followed by either App Developer or Non-App Developer specific questions.

### 7.1.2 Participants

Our participants were from a development team from the Deakin Software and Technology Innovation Laboratory and Outware\(^1\). The Deakin Software and Technology Innovation worked on projects in a number of different domains including defence, transportation, health and finance. Outware specialises in building Android and iOS apps in the domains of government, finance, health, lifestyle and sport. We had a total of 10 participants take part in the study. All of the participants were male with 6 participants non-mobile app developers and 4 mobile app developers. Out of our participants 6 had worked on 3 or more mobile apps and 3 had worked on 1-3 mobile apps showing that non-mobile app developers occasionally work on mobile apps. An overview of our participants’ responses to our demographics questionnaire is shown in Figure 7.1.

---

\(^1\)http://www.outware.com.au/
Figure 7.1: Participant responses to the demographic questionnaire.

7.1.3 Results and Discussion

After participants had completed the questionnaire on the technical domain tactics we held an interview to gather qualitative data. Interviews were held at the offices of the company or university where the development teams were located. Each interview run for approximately 20 minutes and were recorded with a smart phone. The transcripts from each interview took 1 hour to transcribe and can be found in Appendix D.

Interview questions were designed to explore the completeness of the captured domain knowledge, assess the completeness of the domain knowledge and to evaluate the importance of the captured knowledge.
Table 7.1: Interview questions asked to all participants

<table>
<thead>
<tr>
<th>Number</th>
<th>Question</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Is there anything missing from the domain model? Why?</td>
</tr>
<tr>
<td>2.</td>
<td>Are there concepts that you would remove from the domain model? Why?</td>
</tr>
<tr>
<td>3.</td>
<td>What are the top 5 most important development problems? Add other questions if need be.</td>
</tr>
<tr>
<td>4.</td>
<td>How did you rank &quot;importance&quot;?</td>
</tr>
<tr>
<td>5.</td>
<td>How would you like the information in domain tactics presented to you? wiki, tool etc.</td>
</tr>
<tr>
<td>6.</td>
<td>Useful for novice developers?</td>
</tr>
</tbody>
</table>

For each interview we asked a set of core questions to all participants which are shown in Table 7.1. We also asked a specific set of questions based on whether the participant was a mobile app developer (Table 7.4) or a non-mobile app developer (Table 7.5).

General Questions

Below we will discuss the responses to each of the questions starting with the general questions asked of all participants.

1. **Is there anything missing from the domain model? Why?** Out of the participants, 7 indicated that there was no concepts missing from the domain model. The concept of security was mentioned as missing from the conceptual model and represents a core concept that should be present. There is overlap between the concept of security and Social and Personal as personal information needs to be secured. For this reason we decided not to add security to the model.

2. **Are there concepts that you would remove from the domain model? Why?** Social Interactions was mentioned by 2 participants to remove as on Android data can be shared without integrating a social media platform and most apps are not social media apps. However, 6 of the participants indicated that they would not remove any concepts from the domain model.
Table 7.2: Problem Contexts for mobile app development

<table>
<thead>
<tr>
<th>Number</th>
<th>Problem Context</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Multiple platforms need to be supported.</td>
</tr>
<tr>
<td>2.</td>
<td>App depends heavily on data stored in the cloud.</td>
</tr>
<tr>
<td>3.</td>
<td>Must support many devices with different hardware capabilities.</td>
</tr>
<tr>
<td>4.</td>
<td>The battery drains quickly.</td>
</tr>
<tr>
<td>5.</td>
<td>Users need to be made aware of fresh content (or new event).</td>
</tr>
<tr>
<td>6.</td>
<td>App needs to run on devices with varying screen sizes.</td>
</tr>
<tr>
<td>7.</td>
<td>Data from the app needs to be shared with other people or integrated with other apps.</td>
</tr>
<tr>
<td>8.</td>
<td>App relies on the current user’s context i.e. location, time etc.</td>
</tr>
<tr>
<td>9.</td>
<td>Users need to authenticate with the App.</td>
</tr>
<tr>
<td>10.</td>
<td>App frequently freees or crashes.</td>
</tr>
<tr>
<td>11.</td>
<td>User needs to customise the behaviour of the app.</td>
</tr>
</tbody>
</table>

3. What are the top 5 most important development problems? Add other questions if need be. A set of problems were presented to the participants as shown in Table 7.2. Each participant was then asked to rank these in order of importance. The results for this are shown in Table 7.3.

Problem Context number 2 “App depends heavily on data stored in the cloud” was mentioned in the top 5 by 9 participants. This highlights the importance of dealing with data when developing a mobile app. The next most common problems were “Multiple platforms need to be supported.” and “Must support many devices with different hardware capabilities.” with 7 participants including these problems in their top 5.

4. How did you rank "importance"? Importance was measured in a variety of ways by the participants including 3 that used personal ex-
Table 7.3: Importance of problems as indicated by the participants.

<table>
<thead>
<tr>
<th>Participant</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1.</td>
<td>1</td>
<td>2</td>
<td>9</td>
<td>11</td>
<td>8</td>
</tr>
<tr>
<td>P2.</td>
<td>2</td>
<td>3</td>
<td>8</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>P3.</td>
<td>10</td>
<td>4</td>
<td>6</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>P4.</td>
<td>10</td>
<td>4</td>
<td>9</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>P5.</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>P6.</td>
<td>10</td>
<td>4</td>
<td>1</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>P7.</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>P8.</td>
<td>1</td>
<td>6</td>
<td>3</td>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>P9.</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>P10.</td>
<td>3</td>
<td>2</td>
<td>5</td>
<td>8</td>
<td>6</td>
</tr>
</tbody>
</table>

Experience, 4 that measured impact on usability. Others used frequency of occurrence, difficulty to fix and when the problem occurs. One user highlighted the importance of addressing user facing issues saying “I uninstalled the facebook app specifically because it drains my battery”.

5. How would you like the information in domain tactics presented to you? Wiki, tool etc. Asking participants this question provided insight into how the individual likes to learn technical content. The most common answer (3 participants) was to provide this content in a wiki. Two of the participants indicated that the current content can be improved by adding working examples to each of the domain tactics with one participant saying “For me it would be some kind of documentation accompanying working examples with low overhead to run”.

6. Useful for novice developers? All of the participants agreed that the information in the domain tactics was useful to novice developers. One reason for this is that “there are things in there that you want to be thinking about when you start designing apps because they are the kinds of things that are painful to add in later on”. This shows that the technical domain model captures the essential elements of building a mobile app. One participant pointed out that a lot of their learning occurred from making mistakes. This raises an interesting question, do developers need to learn about concerns that tooling/machines can take care of for us? We have taken the approach that automation can greatly assist in addressing those concerns.
Table 7.4: Interview questions for participants who indicated they were app developers

<table>
<thead>
<tr>
<th>Number</th>
<th>Question</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Have you been faced with these development problems at least once when developing a mobile app?</td>
</tr>
<tr>
<td>2.</td>
<td>Which ones have/haven't you come across?</td>
</tr>
<tr>
<td>3.</td>
<td>What other development problems have you encountered that are not listed here?</td>
</tr>
<tr>
<td>4.</td>
<td>Are there any development problems that you would remove from the list?</td>
</tr>
<tr>
<td>5.</td>
<td>Are there any other tactics to these development problems that you have used?</td>
</tr>
<tr>
<td>6.</td>
<td>Do you use these domain tactics in current apps?</td>
</tr>
</tbody>
</table>

App Developer Questions

1. **Have you been faced with these development problems at least once when developing a mobile app?** All of the answers to this question indicated that developers had faced these development problems, “Yeap, pretty much all of them to some extent or another.” and “I think I have encountered all of them.” From this we can see that we have identified problems that are familiar to experienced mobile app developers.

2. **Which ones have/haven’t you come across?** The most frequently encountered problem mentioned by 3 of the app developers was “Data from the app needs to be shared with other people or integrated with other apps”. One explanation is that this feature is not available for all mobile platforms so apps that leverage this feature can only be made available to a limited number of customers.

3. **What other development problems have you encountered that are not listed here?** There were only two problems mentioned that were not already in the list of problems: Security and Timeliness. While security is an important quality attribute it is covered by the domain concern of Social and Personal as this considers the requirements of managing personal data. To handle Timeliness or Performance a quality model for mobile apps could be added to the domain model.

4. **Are there any development problems that you would remove**
Participants agreed that they are all relevant but they were dependent on the app being built with one quote saying “I mean they are all valid problems. Again I think they are very app or problem specific.” That is, not all problems were relevant to the app being built but that it was dependent on the features of that app.

5. Are there any other tactics to these development problems that you have used? A variation of the thin client tactic was suggested that involved sharing code for the data access layer across multiple platforms. Based on this finding we will add an additional domain tactic to the next version of the tactic catalogue that describes this approach.

An enhancement was also proposed for the tactic “Determine User Context” by a participant who said “Developing multiple algorithms, there are already algorithms from various people that use multiple sensors.” This indicates that technical domain tactics change over time as the platform matures and requires constant updates to provide developers with the most recent best practices.

6. Do you use these domain tactics in current apps? Out of the app developers, 4 of them used tactics in their current apps with Limit Sensor Usage, Settings Screens and Notifications as the most popular (See Appendix B for a full list of domain tactics). Settings screens was mentioned despite several earlier responses indicating that the customisation of an app was not a frequent occurrence. An explanation for this is that developers view customisation of an app as disabling features rather than configuration. To overcome this misunderstanding the domain tactic can discuss configuration in the context of controlling the app rather than just styling an app.

Non-app Developer Questions

1. What do you think of these development problems in relation to other development problems you have come across? Three of the responses to this question indicated that the problems were generally applicable to other domains One participant stated “I’ve been doing stuff in distributed computing so I’m sympathetic to issues like network
Table 7.5: Interview questions for participants who indicated they were not mobile app developers

<table>
<thead>
<tr>
<th>Number</th>
<th>Question</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>What do you think of these development problems in relation to other development problems you have come across?</td>
</tr>
<tr>
<td>2.</td>
<td>Are there any development problems that you can think of that would be handy to know about when getting started?</td>
</tr>
<tr>
<td>3.</td>
<td>Is there anything that you have learned about mobile app development that you didn’t know before this interview?</td>
</tr>
<tr>
<td>4.</td>
<td>Would you use these domain tactics if you had to start building mobile apps?</td>
</tr>
<tr>
<td>5.</td>
<td>Does the conceptual domain model help you better understand mobile app development?</td>
</tr>
</tbody>
</table>

connection problems” which indicates the concerns of the mobile domain can be better understood if they have been encountered in another domain. This finding also indicates that the problems capture domain knowledge which is obtained through experience.

2. Are there any development problems that you can think of that would be handy to know about when getting started? The responses to this question included specifying what to do to test the app with one participant saying “quick deployment of the app for testing” was missing. Another participant desired to find out how a login module could be quickly developed. These two suggestions show a desire for novice app developers to have access to solutions for common problems. Both ideas are something that can be taken into consideration by the mobile platforms when designing their getting started documentation.

3. Is there anything that you have learned about mobile app development that you didn’t know before this interview? When asked this question about mobile app development one participant exclaimed that “It’s terrible!” Which highlights the importance of considering the technical concerns when building a mobile app. Other participants indicated they had been made aware of important concepts saying “definitely the hardware constraints and the platform constraints were made
more obvious to me.” and “I just think it has been framed in a nice way. I can see a representative spectrum of things.”

4. Would you use these domain tactics if you had to start building mobile apps? Each participant indicated that they would use these tactics when building a mobile app with one individual answering with “yes, of course, pretty handy you don’t have to waste your time when you just realise you could have done that.” This highlights the value of the domain tactics at the start of development.

5. Does the conceptual domain model help you better understand mobile app development? All of the non-app developers agreed that the conceptual domain model did increase understanding about mobile app development. Two participants indicated that our documentation can be improved by adding a developer checklist to act as a guide for the developer as they are building an app. This checklist would then help ensure that developers avoid the problems as presented in this study.

While developers acknowledged that the technical domain model was comprehensive with nothing to add or take away, all domain tactics could be improved with examples. Overall the responses indicated that novice developers would benefit from understanding the technical domain model before building an app especially if tool automation was provided to address the concerns.

7.1.4 Threats to Validity

In this section we discuss the construct, internal and external threats to validity.

Construct validity could be missing based on the bias in experimental design. When answering lots of questions where there are multiple choices candidates can tend to mark all questions the same. In our experiment the strongly disagree was located on the left and strongly agree located on the right. If a participant marked all the questions down one side it bias the results. To detect when this was happening we switched the order of strongly agree and strongly disagree for one of
the questions.

One threat to internal validity is that participants were only shown the domain model and the problems associated for 1 hour which is not very long to find gaps. Showing the concepts in a short time frame means that developers are less likely to state that a concept is missing. To address this issue a follow up questionnaire at a later date could be completed by the participants to see if they still agreed with their assessment of the domain tactics and problems.

Another challenging component of this study is to determine the impact of the knowledge that the non-app developers gained from reading through the technical domain concerns. We asked qualitative questions about whether they learned something from our study but we did not include a quantitative measure. This shows the perceived benefit of the domain knowledge. In the future, another study can be conducted that looks at two groups as they build a small app one that has the domain tactics available and one that does not. This would enable us to compare the impact domain tactics have on the quality of an app.

A threat to external validity is the number of participants. As only 10 people were interviewed we cannot draw any statistical significant results. From these results we can indicate findings that warrant further investigation and provide insights into how the technical domain model can be improved. To rectify this problem another questionnaire would be run with more participants to ensure statistical significance can be obtained.

Our results show that domain tactics are useful for novice developers to understand architecture concerns when building a mobile app. Indicating that the technical domain model and domain tactics capture appropriate architectural knowledge.
7.2 Replicating Existing Apps

We replicated existing apps to evaluate our meta-model for completeness. The intention was to capture the commonality between data-intensive apps and provide a breadth of coverage of features across apps. By doing this we planned to address the research question **RQ2. How do we empirically build a meta-model that contains the core abstractions a developer uses to build mobile apps?** We focus our evaluation on the concepts in the meta-model and determine if they are sufficient for modeling apps independent of the business domain. We performed our evaluation by taking 10 mobile apps from different business domains and generating the architectural scaffolding for each app. The generated apps are hosted online here\(^2\) along with the code used to create them. Screenshots from the Centrelink app are shown in Figure 7.2. The focus of this experiment was completeness – that is, we wanted to identify the concepts that could not be expressed using the meta-model, as implemented by RAPPT\(^3\) and hence identify limitations with the meta-model.

![Example screens generated by RAPPT for the Centrelink app.](https://github.com/ScottyB/rappt-eval/tree/master/gen-apps)

---

\(^2\) [https://github.com/ScottyB/rappt-eval/tree/master/gen-apps](https://github.com/ScottyB/rappt-eval/tree/master/gen-apps)

\(^3\) [http://rappt.io/](http://rappt.io/)
7.2.1 App Selection

All the apps we selected were data-intensive mobile apps as defined in Chapter 2. Apps were selected that were a thin data-driven client, relied on web services for content, context aware, focused on UI/UX and addresses the concerns associated with data storage. We selected apps from the top ranked list as provided by Google and available on the Google Play Store\(^4\) on the 1st of March 2014, in Australia (which is our default store).

Apps were selected from different categories however, we omitted the following categories for the reasons outlined below:

- **Personalization**: Apps in this category are designed to customise a users phone. Examples include apps that change the style of the launcher, icons used on the device and additional ringtones. These apps typically do not fetch content from a web service and thus do not have a good representation of data-intensive apps.

- **Widgets**: Widgets are the type of apps that run on the background of a users device. They show relevant information such as the time, weather or share prices. Widgets could technically be added to the supported apps that we generate but remain an area of future work. These types of apps are specific to the Android platform.

- **Games**: Games seek to develop unique interactions with great visual affects enhanced with sound effects – they have a high degree of custom functionality which is unsuitable for MDD techniques. Games have highly customised UI elements that are outside of the guidelines provided by the platform vendors. As such these types of apps fail to qualify as data-intensive apps that focus on a specific business domain.

- **Live Wallpaper**: Live Wallpapers are similar to a traditional desktop wallpaper except that they are animated. Live Wallpapers are not real apps as they are backgrounds for the device.

\(^4\)https://play.google.com/store/apps/top?hl=en
Table 7.6: Apps selected from the top ranked apps on the Google Play Store

<table>
<thead>
<tr>
<th>App</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ebay</td>
<td>Shopping</td>
</tr>
<tr>
<td>Netflix</td>
<td>Entertainment</td>
</tr>
<tr>
<td>Gumtree Australia</td>
<td>Lifestyle</td>
</tr>
<tr>
<td>CommBank</td>
<td>Finance</td>
</tr>
<tr>
<td>Seek</td>
<td>Business</td>
</tr>
<tr>
<td>Domain Real Estate &amp; Property</td>
<td>Lifestyle</td>
</tr>
<tr>
<td>TripView Sydney Lite</td>
<td>Travel &amp; Local</td>
</tr>
<tr>
<td>Express Plus Centrelink</td>
<td>Productivity</td>
</tr>
<tr>
<td>Clean Master</td>
<td>Tools</td>
</tr>
<tr>
<td>Spotify</td>
<td>Music &amp; Audio</td>
</tr>
</tbody>
</table>

The 10 apps that we selected along with their category are shown in Table 7.6. We built each of these apps using RAPPT which is available online. We developed four screens for each of the apps that included the navigation flow, fetching data, and rendering content to the screen. All of the generated apps can be downloaded and run on an Android device (version 4.0 or higher).

7.2.2 Findings

RAPPT was able to model and generate 3-4 screens from all of the 10 selected apps. This demonstrates that there is functionality that is common between data-intensive apps that can be modelled using RAPPT. Apps were selected from a range of categories indicating that the features of a data-intensive app are widely used. RAPPT was able to support both of the common UI navigation patterns as well as navigation from lists, buttons and menu items. Through RAPPT’s ability to mock the API, realistic data was able to be displayed in the generated apps. RAPPT was also able to model the data displayed to the user in the apps and replicate it in the generated apps.

Partial features represent concepts in an app that can be modelled by a variation in either the Domain Specific Visual Language or Domain

5http://rappt.io/
Specific Textual Language present in RAPPT. That is, the underlying abstraction is present in the meta-model but not all of the variations for that feature are supported. RAPPT has the concept of a Marker for a Map but not a Marker that has the location data loaded from an API call. Abstractions exist for lists but not cards which are a visual variation of a list, form elements such as input fields are part of RAPPT but they cannot be set to hide the input for a password field. Another example is a series of introduction screens that the user swipes through, the concept of a screen exists but the swipe navigation does not.

The main features that RAPPT was unable to generate were custom UI components and animations, fine grain styling information and business logic. All of these characteristics are required to produce a mobile app and need to be added to the generated code by the developer. These features require highly specific functionality that is tailored to each app which RAPPT does not support by design, see Chapter 6. RAPPT is intended to automate the mundane work, boilerplate aspects of the app, and more importantly architectural/structural aspects that are well suited to model based tools.

### 7.2.3 Threats to Validity

One of the internal threats to validity is selection bias. For the purpose of this evaluation we selected 10 mobile apps to reproduce. As there are many categories of apps, increasing the number of apps produced would provide a broader understanding of the limitations of RAPPT. Select bias is also a concern when it comes to the selection of screens to mock-up. For each app 3-4 screens were selected and a subset of the functionality was chosen to be replicated.

An external threat to validity that needs to be addressed is that of generalisability of the findings. This threat manifests itself in two ways. First, generalisability across all data-intensive apps as there are only a subset of the total apps are used in the experiment. Second, generalisability within a single category as certain functionality may be more relevant in certain categories.
The variety of apps that we were able to generate suggests that our meta-model captures the core elements for data-intensive mobile apps.

7.3 User Evaluation

The meta-model we developed was evaluated through the construction of a DSL that exposed the concepts to developers. Later, this DSL informed RAPPT’s DSTL and DSVL (see Chapter 6). By creating a DSL the abstractions in the meta-model can be manipulated to construct apps in a form familiar to developers - i.e. a text based language. This enabled developers to use the meta-model to address problems they commonly face when developing mobile apps. Thus, the DSL and supporting tool became the focus of an User Evaluation that was designed to examine how mobile app and non-mobile app developers can benefit from using RAPPT. In the User Evaluation details described in Appendix E the DSL is referred to as the App Modeling Language (AML).

We also wanted to gauge the acceptance of our approach among professional developers. Our User Evaluation was designed to answer the main research question of RQ3. How does the technical domain influence the design and implementation of tools for automation? by addressing three sub research questions:

- **RQ3.1. Does the meta-model contain the necessary abstractions for data-intensive mobile apps?** Abstractions that developers use to construct mobile apps may not be apparent from analysing existing apps – the meta-model lacks necessary concepts.

- **RQ3.2. Would RAPPT’s tool and approach be accepted by developers?** RAPPT is targeting developers who build mobile apps and needs to fit in with their mental model. Developers also need to find RAPPT useful in their day to day work in order to invest the time to learn a new tool.

- **RQ3.3. How can RAPPT be improved to be suitable for use in industry?** RAPPT is ultimately intended to be used in a commercial en-
environment. To achieve this goal we need to ensure that all research problems preventing adoption have been identified and solved.

- **RQ3.4. What aspects of RAPPT should future work focus on? (i.e. gaps in current tool and method)** RAPPT has been designed with the end user in mind from the beginning. To ensure that RAPPT’s development remains practically focused it is essential that the end users help steer the direction of future research.

These questions were designed to evaluate the suitability of RAPPT for producing apps in the mobile app technical domain. Questions RQ3.3 and RQ3.4 are focused on the developer and environment in which RAPPT will operate. These aspects are an important part of the technical domain as shown in previous chapters both aspects influence the software being developed. RQ3.1 ensure that concepts from the metamodel can accurately model apps from the mobile app technical domain. RQ3.2 looks at how RAPPT can be further developed to address the needs of the technical domain.

Resources for performing the User Evaluation including screenshots of the app to build and results are available online the raw results have also been included in Appendix G. This user study has been approved by Swinburne University Human Research Ethics Committee (SHR Project 2014/313). A copy of the documents including ethics clearance letter and an approved addendum are provided in the Appendix E. The following subsections provide details of our user evaluation.

### 7.3.1 Experiment Setup

Our experimental setup consisted of a laptop and mouse. As RAPPT is hosted online participants were free to use their own computers to complete the experiment. By using learning videos we were able to reduce bias and run multiple sessions with different participants ensuring repeatability. The participants included staff from Swinburne University and members from our research group. All participants had

6[https://github.com/ScottyB/rappt-eval/tree/master/user-eval](https://github.com/ScottyB/rappt-eval/tree/master/user-eval)
a background in software or information systems.

- **Initial Setup** – Participants were asked to fill out a demographic survey, watch a learning video of how to use RAPPT to reduce bias and complete a survey to ascertain their confidence level with building Android apps with RAPPT.

- **Pilot Study** – Prior to running the full experiment we also conducted a pilot study with a participant to refine the process and questions.

- **Key Task** – Participants were then asked to build a small three screen data-intensive mobile app based on the freely accessible MovieDB API. Each participant was provided with instructions, an API key and a link to the API documentation. Screenshots for the completed app are shown in Figure 7.3 which is replicated from our tool motivation in Chapter 6. There was no time limit and participants could leave at anytime. The task required participants to implement two mobile app design patterns that were common to app developers, the tabbar and the drill-down. These components were based on our analysis of apps as described in Chapter 5.

- **Installation and Running** – Participant apps were downloaded and run on the instructors device to insure the app had been completed correctly.

- **Final Questionnaire** – At the end of the task, participants were asked to complete a questionnaire that was composed of 17 questions with a 5 point Likert scale ranging from *Strongly Disagree* to *Strongly Agree*, and 8 open-ended questions capturing their experience of using RAPPT to build Android apps. We followed the positive questionnaire design approach as suggested by Sauro et al. [184]
7.3.2 Results and Discussion

### Demographics

For this study we asked 20 (3 female and 17 male) developers and researchers from our research group to participate in our evaluation of RAPPT. There were no age limitations although we did limit the participants to people working in Information Technology. To capture the demographics, we asked participants to fill out a short survey before starting the main experiment. The results of the demographics survey can be seen in Figure 7.6 and in Figure 7.10. Out of 20 participants, only one was not able to finish the experiment successfully. This was due to the participant’s lack of interest in mobile app development and has been reflected in the questionnaire responses. It took the participants approximately 90 minutes to finish the tasks and answer the questions.

From the results in Figure 7.11 we can see that 50% of participants had less than 1 year experience developing mobile apps and from Figure 7.12 that 20% have never developed an app. This indicates that abstractions required for addressing the components in the evaluating
task (drill-down, authentication and tabbar) maybe unknown to the participants. As such the concepts in the meta-model have to be taught before they can be used.

**Figure 7.4:** D1 - How many years experience do you have in mobile app development?

![Bar chart showing distribution of years experience.]

**Figure 7.5:** D2 - How many apps have you built?

![Bar chart showing distribution of number of apps built.]

**Figure 7.6:** User demographic questions 1 - 2 and participants’ responses.

To analyse the participants’ responses, Likert scale scores were collected from questionnaires as well as the text answers to open-ended questions. We collected data from three separate questionnaires all made available to participants using Opinio. In order to answer our research questions we analysed the textual answers using a Thematic analysis process [185]. A different coding was applied to the data depending on the research question of interest.

---

8[http://www.objectplanet.com/opinio/]
Learning Tasks

After participants finished the demographics questions they were given a set of learning tasks that were designed to gain familiarity with syntax and to learn how to use RAPPT. The learning tasks were as follows:

- **Hello World**: The first learning task consisted of writing a “Hello World” example using RAPPT followed by a task that involved nav-
avigating between two screens.

- **Navigation**: This learning task demonstrated how to navigate between two different screens using RAPPT.

- **Navigation Patterns**: RAPPT supports a number of popular UI design patterns for handling top level navigation such as tabbars and navigation drawers. The third learning task showed participants how to use RAPPT to create tabbars.

- **Fetching Content From an API**: One of the key tasks for a data-intensive mobile app is to fetch content from a server and to render the results to the screen. The fourth learning task showed how to do this using RAPPT’s DSTL.

- **Displaying a List**: This learning task built on the previous task by fetching content from a server and rendering the content as a list rather than populating individual fields.

- **Passing data between screens**: The final learning task tied all of the previous tasks together and showed participants how to pass content between two screens. At the completion of this learning task participants could build an app that used the drill down UI pattern as well as create a high level navigation pattern.

The learning task and the user tasks are part of Appendix E.

**Post-Learning Task Questionnaire**

After the learning tasks had been completed the participants were asked to complete a post-learning questionnaire (Appendix G). These questions, shown in Figure 7.14, were designed to assess how confident the participant was to build an app using RAPPT.

45% of participants indicated that they were confident to build a small data-driven app using RAPPT after completing the learning tasks and another 45% were indifferent. 60% of participants felt that they understood the concepts used in RAPPT and 50% indicated rated their skills as strong Android developers. These results show that the majority of
Figure 7.11: Q1 - How confident are you to build a small data-driven app with RAPPT? Scale ranked: 1 (Not very confident) to 5 (Very confident)

Figure 7.12: Q2 - How well do you understand the constructs in the RAPPT? Scale ranked: 1 (Not very well) to 5 (Very well)

Figure 7.13: Q3 - How would you rank your abilities as an Android app developer? Scale ranked: 1 (Poor) to 5 (Excellent)

Figure 7.14: Post learning task questions and participants’ responses. All questions were on a scale 1-5

participants felt that they could use RAPPT to build a small mobile app after the learning tasks.
Table 7.7: Survey questions and their associated research questions.

<table>
<thead>
<tr>
<th>Research Question</th>
<th>Survey Questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. RQ3.1</td>
<td>What concepts would you like to see included in the visual model?</td>
</tr>
<tr>
<td>2. RQ3.1</td>
<td>What concepts would you like to see included in RAPPT’s DSL?</td>
</tr>
<tr>
<td>3. RQ3.2</td>
<td>What would you like to see in the visual model that is already in RAPPT’s DSL?</td>
</tr>
<tr>
<td>4. RQ3.1, RQ3.3</td>
<td>In your opinion, which types of apps are not suited to be developed with RAPPT?</td>
</tr>
<tr>
<td>5. RQ3.2, RQ3.3</td>
<td>What did you like the most about RAPPT?</td>
</tr>
<tr>
<td>6. RQ3.2, RQ3.3</td>
<td>What did you dislike the most about RAPPT?</td>
</tr>
<tr>
<td>7. RQ3.3, RQ3.4</td>
<td>What needs to be added or fixed for you to use RAPPT for all your data intensive Android app projects?</td>
</tr>
</tbody>
</table>

Final Questionnaire

For RQ3.1 we followed a deductive coding approach using the concepts from the meta-model as the start codes. In this case, our interest was in text that could not be labelled as this would give an insight into what was missing from the meta-model. For RQ3.3 we followed a Grounded Theory approach [186] to extract a result from the data we had collected through classification. RQ3.2 used a combination of both approaches as the first author was aware of some possible research directions.

The open-ended questions are shown in Table 7.7 and were designed to predominantly address RQ3.1, RQ3.2 and RQ3.3. Each of the research questions will be discussed with the results in further detail below.

RQ3.1. Does the meta-model contain the necessary abstractions for data-intensive mobile apps? Many of the responses indicated that they desired to have more UI components such as checkboxes and date pickers. These concepts while not present in the tool are modelled in the meta-model by the concept of a UI component. From the results we identified the following concepts missing from the meta-model: support
for video and audio, background services, dialogues, local databases and interactions with other hardware or apps. In the context of this research question, a meta-model for data-intensive apps, the following need to be added to the meta-model: dialogues as they are a base UI component, background services for long running operations and local databases for local storage. Multimedia capabilities and interaction with other apps and hardware are not concepts widely applicable to data-intensive mobile apps.

**RQ3.2. How should future work on RAPPT be spent?** The purpose of this research question was to expose desired attributes for a tool that would synthesise mobile app architecture and guide the future research direction. From the results we extracted 3 areas of future work. A number of answers indicated there were inconsistencies in the DSL associated with scopes and syntax. Fixing these inconsistencies and improving the learnability of the DSL is the first area of research. Another area of research from our findings is looking at how to display a preview of the generated app during development. The third and final research area identified would study how to extend RAPPT to support components.

**RQ3.3. How can RAPPT be improved to be suitable for use in industry?** Participants indicated that for adoption by industry RAPPT needs to fix the technical issues, include features of an Integrated Development Environment (IDE) and provide a better feedback cycle. Technical issues include UI enhancements to improve documentation and visual designer, improve implementation and provide better error reporting. IDE features mentioned in the responses included code completion, auto code formatting and automatic saving. A better feedback cycle refers to not being able to view the app until downloaded and run on a device. One way to overcome this issue would be to provide a link for installing the app directly to the device reducing the feedback cycle. RAPPT’s forward engineering only approach was also mentioned as a drawback. As a web application RAPPT is not suitable for continual development of a project located on the desktop computer.

**RQ3.4. Would RAPPT’s tool and approach be accepted by develop-**
ers? To answer this we asked two sets of questions designed to extract their impression of using RAPPT. Each question used the Likert score of 1 to 5 where 1 represented *Strongly Disagree* and 5 *Strongly Agree*. The first set of questions focused on the usability of RAPPT and the ease of use. The second set focused on the developers impression of the tool for building data-intensive mobile apps. Both sets of questions can be seen in Table 7.8. It also depicts the frequency of participant responses to each question.

**Table 7.8:** Sample questions of the questionnaire. Likert points have been given score of 1 to 5 representing *Strongly Disagree* to *Strongly Agree*.

<table>
<thead>
<tr>
<th>No.</th>
<th>Question</th>
<th>Frequency (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td><strong>g.1</strong></td>
<td>It was easy to use RAPPT.</td>
<td>0</td>
</tr>
<tr>
<td><strong>g.2</strong></td>
<td>It is easy to understand what each icon represents.</td>
<td>0</td>
</tr>
<tr>
<td><strong>g.3</strong></td>
<td>It was easy to load data from an API and render it to the screen.</td>
<td>0</td>
</tr>
<tr>
<td><strong>g.4</strong></td>
<td>It is easy to avoid making errors or mistakes.</td>
<td>5</td>
</tr>
<tr>
<td><strong>g.5</strong></td>
<td>RAPPT makes you more productive when starting a new project than using templates from an IDE.</td>
<td>0</td>
</tr>
<tr>
<td><strong>g.6</strong></td>
<td>The concepts in RAPPT are sufficient for modeling a mobile app.</td>
<td>0</td>
</tr>
<tr>
<td><strong>g.7</strong></td>
<td>RAPPT is useful for mobile app development.</td>
<td>0</td>
</tr>
<tr>
<td><strong>g.8</strong></td>
<td>Using RAPPT is more efficient than starting with a raw Android project.</td>
<td>5</td>
</tr>
<tr>
<td><strong>g.9</strong></td>
<td>You are satisfied with using RAPPT.</td>
<td>0</td>
</tr>
</tbody>
</table>

The overall response was positive with 80% of participants agreeing or strongly agreeing to 7 out of the 9 questions. These results indicate that professional developers can use RAPPT and that our approach is inline with the needs of modelling a data-intensive mobile app. The first 4
questions pertained to usability of the tool and the last 5 related to the tool approach. Q2 received the strongest result with 65% of participants Strongly Agreeing to the icons in the DSVL being easy to understand. This confirms our current choice of icons for the concepts present in a data-intensive mobile app.

An interesting finding from the survey was the results to question Q4 on RAPPT enabling users avoid making mistakes. It shows an almost normal-distribution spread. The answers and comments on the questionnaire indicated needs for improving error handling mechanism as most participants have experience with major software development IDEs. Some participants mentioned that they would have preferred a more real-time error handling mechanisms. A possible way to improve the error handling is to move more of the concepts from the DSTL to the DSVL – visual languages are not as susceptible to developer error as textual languages are. Improving the overall robustness of the tool is another way to address the issue of poor error reporting.

Answers to Q6 were also non-committal with 40% of participants being neutral. The concepts in RAPPT cover the main concepts required for modeling data-intensive apps but are not exhaustive. For example the concept of a navigation pattern is present although not every navigation UI pattern is supported. This is one reason why the results may not be strongly one way or another. Clarifying the question by indicating data-intensive apps rather than all mobile apps would also remove some confusion.

Participant responses to question Q7 indicate the overall acceptance of the approach. 95% of the participants had positive views on the usefulness of the approach (60% Strongly Agree and 35% Agree). Same is true for responses to question Q9.

Participants found that RAPPT greatly improved their productivity. Here are a few sample responses to the survey question: What did you like the most about RAPPT? “the ease and simplicity saves a LOT of time (seriously!)”, “Easy to use, save time!” and “It’s able to produce ready to use app instantly”. These results show that our approach has potential
and further work is needed to evaluate the ongoing development of an app created by RAPPT.

### 7.3.3 Threats to Validity

The User Evaluation involved 20 participants which is sufficient for our purposes but not for statistical analysis. In future work we will evaluate RAPPT with more professional developers. Another threat to validity is best summed up with a comment from one of the participants “... I need more time to play around with RAPPT...”. Participants built one app taking approximately 1 hour to complete and are likely to give different responses after using RAPPT extensively. It would be interesting to see how the answers from the participants changed after using RAPPT for an extended period of time on multiple projects. The participants were asked to complete one task. An improvement to the evaluation task would involve selecting a range of apps with different functionality and purposes. Evaluating RAPPT on additional apps would also help expose gaps in the model and the language which could be used to inform the implementation.

In our evaluation we asked the participants to build an app which the authors verified against the criteria outlined in Appendix H. The participants were not asked to import the generated projects into an IDE which would be necessary in a commercial environment. The simple task did not require the participants to modify the generated code. Thus, not every step required for using RAPPT was evaluated. Future work would require participants to download and extend the generated output by adding an additional feature.

### 7.4 Case Studies: Technical Domain Analysis

In this section we show how the technical domain model in Chapter 4 can be used to analyse software architecture. The following two case studies address the research question RQ1. *What are the concerns of the technical domain that influence architecture and how can this infor-
How can we determine if the concerns in the technical domain model are sufficient for describing the tradeoffs between competing quality attributes?

In the first case study, Malaysian Bus Tracker, we wanted to see how analysing requirements using the technical domain model guided an implementation. We found that the technical domain concerns provided additional quality requirements that had not been made explicit in the requirements document. The second case study, Google IO 2014, was designed to analyse an app against our technical domain model. Specifically we wanted to see 1) what issues the technical domain concerns created, 2) what tradeoffs needed to be made between technical domain concerns and 3) analyse how these tradeoffs and issues were addressed. Our case studies focused on analysing source code.

A third case study was conducted on Prompa, an app for job scheduling which is available in Appendix K.

### 7.4.1 Malaysian Bus Tracker

The Malaysian Bus Tracker case study demonstrates how the technical domain model can be used to identify additional requirements from existing requirements. This is feasible due to the technical domain model containing a link between quality attributes and the domain concerns.

Malaysian Bus Tracker (MBT) is an app designed to provide up-to-date bus tracking information to customers. The UI designs for 4 of the screens of MBT are shown in Figure 7.15. In this section we will analyse the requirements for the app before analysing the architecture of the solution. Before development on MBT started a requirements document was created with the following requirements grouped into cate-
CATEGORIES: Functional, Non-Functional and Device. Below we have listed the requirements for each group and analysed them with respect to our technical domain model to extract additional quality attributes.

**Figure 7.15:** UI screen designs for the Malaysian Bus Tracker Customer app.

**Functional Requirements**

- **R1.** Display the bus stops closest to the user.
- **R2.** Enable users to search for a bus stop.
- **R3.** Provide a means for users to add and remove favourite bus stops.
- **R4.** Display the closest bus for each route for a selected bus stop.

Requirement **R1** requires the app to access the built in Global Positioning System (GPS) to determine the users current location, **Sensing.** Showing the users location only when the look at the device means the users' location does not need to be continuously polled. This makes the app more efficient as it places less demand on the battery, **Hardware Constraints.** Making use of the GPS raises concerns about reliability of sensor values i.e. Will the sensor values enable the closest bus stop to be calculated when the user is indoors? Developers also need to address the concern of timeliness of the closest bus stop and raises questions about how often and when the calculations should be performed.
Requirements **R1**, **R2** and **R4** are likely to involve making a call to a web service, *(Network Connectivity)* and involves an understanding of the *Threading Model*. In the case of **R2**, where the bus stop information is updated infrequently, data could be shipped with the app to improve the availability of the app in areas of poor network connectivity. Decisions about where the processing of the nearest bus will be performed will impact upon the architecture of the app.

As soon as private information is stored as in **R3** the concern of security is raised. Decisions need to be made about where the personal information will be stored and how it will be accessed. Favourite app information *(Social and Personal)* could be stored in a remote database which would require user authentication and calls to a web service to obtain that information, *(Network Connectivity, Threading Model and Lifecycle)*. This creates issues around availability of the information, when and how the information should be fetched.

**Solution** The developers of MBT make two calls to web services to obtain the closest bus stops and busses for those routes. These network calls are called on a thread that could be suspended and not on a background service. This could create issues with performance if there is poor network connectivity as the whole UI would hang.

**Non-Functional Requirements:**

- **R5.** The app must be available for download from both Apple and Google’s app stores. *(Availability)*

- **R6.** Update a bus’s location every 30 seconds when there is an internet connection. *(Performance)*

- **R7.** The app must be able to handle peak time loads. *(Scalability)*

**R5** has two implications, first either an app has to be developed for each platform or a cross platform app needs to be built. Second, the developed app must follow the guidelines outlined by each platform *(Framework and Ecosystem)*. Choosing to develop a hybrid app requires developers to build the majority of the app as a website to achieve the
benefits of a cross-platform solution. Web apps do not have the same responsiveness or features of a native app and complex UI interactions will need to be developed as a native app (Usability). The type of app to build impacts upon the language, framework, libraries and UI patterns which are all architectural concerns. It also impacts upon the re-usability of the code as a pure native app can not be ported easily to another language and platform.

**R6** indicates that there needs to be a continuously running background task to handle the constant updates, *Threading Model*. Tradeoffs need to be made between performance and power consumption (Efficiency), *Hardware Constraints*. As this requirement needs to run in real time for it to be useful to the user a loss in *Network Connectivity* needs to be handled appropriately with a suitable warning shown to the user. Developers face an architectural decision here to decide whether they push the updated bus information to the devices or if the devices poll a webservice every 30 seconds. These decisions centre around the timeliness of the data, does it need to be shown to the user every 30 seconds or only when they are interested in it.

Requirement **R7** directly impacts the usability of the app and is a requirement for the mobile app. To fulfil this requirement the web services for this web app need to be designed to handle peak loads at certain times. This is an example of how a requirement of a mobile app impacts the architecture and design decisions of not only the app itself but also the server layer.

**Solution** The developers of MBT have opted to build two separate apps, one for each of the platforms. A design decision like this strongly influences the architecture of the app as a significant portion of the business logic should be hosted in the cloud in order to avoid code duplication. Moving the business logic to the cloud influences the concerns of *Sensing* as sensing values need to be sent to the server layer (*Network Connectivity*) as well as influencing issues around using the *Threading Model*.

From our analysis of the source code **R6** has yet to be implemented.
Performance is one of the main quality attributes that need to be addressed when building an app so it is vital that decisions pertaining to how long to set updates needs to be tested. In this case, where the significant portion of the logic is hosted in the cloud meeting this requirement depends heavily on the design and implementation of the server layer.

**Device Requirements:**

- **R8.** Android app must work on devices with Android 4.1 and up.
- **R9.** iOS app must work on iOS 7 and up.

These requirements directly pertain to the *Hardware Constraints* and *Framework and Ecosystem* concerns. Hardware specifications for any functionality developed must work on these required devices. The mobile platforms constantly update their SDKs adding new features and deprecating others. Depending on the functionality desired in the app developers might to create new widgets or work around limitations of the selected SDK which may impact upon the architecture. These issues relate to the availability of functionality and hardware specifications.

**Solution** The solution here was to simply build two separate mobile apps on each platform, Android and iOS. As the app did not use any of the latest features available on a single platform there were no issues of portability.

**Key Findings**

From this case study we demonstrated how the requirements can be analysed using the technical domain model to reveal additional concerns. We found the following key findings:

- Using the technical domain model reveals additional questions that need to be solved before an architecture for the system can be put in place. For example, using the GPS to monitor the bus means that developers will have to plan handle the timeliness of
the system and confirm whether the location can be calculated when the user is indoors.

• Quality requirements for a system map to concerns in the technical domain model. For each of the non-functional requirements we were able to identify technical domain concerns that needed to be considered in order to satisfy that requirement i.e. the requirement \textit{R6. Update a bus’s location every 30 seconds when there is an internet connection.} maps to Threading Model and Hardware Constraints.

• In some instances the technical concerns influence the work done but cannot be worked around. Although the Device Requirements mapped to the Hardware Constraints and Framework and Ecosystem concerns, the final solution involved factoring in additional time to build two separate apps.

7.4.2 Google IO 2014

The Google IO app is released by Google every year and serves “as a practical example of best practices for Android app design and development.”\textsuperscript{9} – essentially it is a Reference Architecture for the community. Google I/O is a developer conference held every year that showcases many technical demonstrations and talks. The Google I/O app is an app for that conference. The main features of the app are as follows:

• Presents the conference agenda and permits the creation of your own schedule.

• Synchronises your schedule between multiple devices.

• Displays information about the speakers and sessions.

• Integrates with the social media platform, Google+.\textsuperscript{10}

• Triggers reminders before a sessions starts.

\textsuperscript{10}https://plus.google.com/
Table 7.9: LOC for each of the major packages in Google IO 2014

<table>
<thead>
<tr>
<th>Package</th>
<th>LOC</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>service</td>
<td>996</td>
<td>4</td>
</tr>
<tr>
<td>io</td>
<td>1352</td>
<td>6</td>
</tr>
<tr>
<td>sync</td>
<td>1710</td>
<td>7</td>
</tr>
<tr>
<td>provider</td>
<td>1833</td>
<td>8</td>
</tr>
<tr>
<td>others</td>
<td>2075</td>
<td>9</td>
</tr>
<tr>
<td>utils</td>
<td>2880</td>
<td>13</td>
</tr>
<tr>
<td>ui</td>
<td>11999</td>
<td>53</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>22845</strong></td>
<td><strong>100</strong></td>
</tr>
</tbody>
</table>

- Allows information to be passed between devices using NFC technology.
- Enables user feedback for each session.

To ensure we selected a non-trivial app to study, we counted the LOC using CLOC.\(^\text{11}\) We only counted lines of code excluding all comments and blank lines. Android apps have XML files to represent layout and presentation and use Java code for the controllers and business logic. For this study we only analysed the Java files of the application. The results of this analysis is shown in Table 7.9.

Below we map functionality from the Google I/O app to concepts in the technical domain to demonstrate how the developers have addressed the technical domain concerns. By showing how these concerns are addressed in an actual app we can show that developers need to consider the technical domain model when building a mobile app.

In this study we illustrate the following technical concerns:

- Network Connectivity and Threading Model
- Social and Personal, Sensing, Lifecycle and Hardware Constraints
- Sensing, Lifecycle and Threading Model

\(^{11}\)http://cloc.sourceforge.net/
Figure 7.16: Main data sources for Google IO 2014 app.

**Description:** Data-intensive apps are heavily dependent on their data sources for fetching and interacting with information. The data sources used in the Google I/O 2014 app are shown in Figure 7.16. There are four major data sources in use in the Google I/O 2014 app: a bootstrapping file shipped with the app, a local cache, a local database and a remote database.

**Issues:** Poor *Network Connectivity* can severely impact the performance of an app as data takes a long time to load. This issue is important to consider when loading data as soon as the app starts as there is no guarantee that a device will have a steady network connection. In data-intensive apps the *Threading Model* needs to be well understood so that 1) the UI thread is not blocked and 2) the UI is not updated outside of the UI thread to ensure a good user experience.

**Tradeoffs:** To overcome performance issues of network requests developers can opt to only make API calls when on a wireless network – wireless networks are generally more reliable than other connections. Another approach is to fetch all the data the app needs when the app is opened for the first time. However, both approaches will fail to show content to users if there is no connection when the app first opens.
To avoid blocking the user when a network connection is made developers have a choice of threading options they can spawn a thread that will get suspended when the app is placed in the background or they can create a background service that is running all the time. Spawning a separate thread from the UI thread will cause it to be shutdown once the app is closed and will need to be restarted. In traditional software development this issue does not arise as the choice is simply which thread to run on as operating systems do not place apps in the background in quite the same way as on a mobile device.

**Solution:** To address the issue of *Network Connectivity* the Google developers have implemented two strategies. First, the Google I/O 2014 app comes with bootstrapping data to load when the app first starts up. This ensures that the app is responsive the first time the user opens the app and is fully functional irrespective of the quality of the connection. Unlike typical web applications, mobile apps allow for data to be shipped with the app and can avoid displaying a loading screen to the user. The downside of this approach is that the data is slightly out-of-date. Second, there is an aggressive caching strategy in place in the Google I/O 2014 app. The app first checks that the data it has in the cache is out of date before making further network requests improving the user experience of the app. Although caching strategies are implemented in many software systems to improve performance they are of paramount importance when developing mobile apps due to the issues around network connectivity.

The Google I/O app uses a constantly running background service to synchronise data between the server and the mobile device. Users can trigger a manual update if they wish but are not presented with long running loading screens. The choice to perform the synchronisation in a background service means that it will not be interrupted when a higher priority app runs, such as an incoming phone call due to the *Single Threading Model* of mobile apps. This avoids the issue of requiring a server to respond to requests for sending data part way through a transmission. The downside is that care needs to be taken to stop and start the service as appropriate so as not to be a significant drain on the battery.
Example 2: Social and Personal, Sensing, Lifecycle and Hardware Constraints.

**Description:** The Google I/O app is intended to be used by attendees to the Google I/O conference. The app takes advantages of a device’s sensors to send information about a session to another user (Social and Personal) using Near Field Communications (NFC).

**Issues:** The latest mobile devices have certain sensing capabilities such as having NFC chips. One draw back of this is that the Hardware Constraints of older devices means that they do not have NFC chips requiring developers to consider this eventuality. When the Sensing capabilities of a mobile device are used continuously contention arises between performance and accuracy. Continuously polling sensor values, such as nearby devices, provides up-to-date information but is a sure fire way to drain the battery. Poor energy efficiency impacts upon the usability of the app in a social setting which impacts the concern of Social and Personal.

**Tradeoffs:** To overcome the Hardware Constraints of older devices a requirement can be that all mobile devices downloading this app must have NFC chips. While this ensures that all the functionality works correctly it does restrict the number of users who can access the app. Another approach to tackle this issue is to provide similar functionality without using the advanced features and often less convenient. When using the Sensing capabilities of mobile devices, decisions about usability need to be made. Scanning other attendees badges even when the app is not in the foreground is very convenient but drains the battery. To improve upon this method a notification can be triggered informing the user that they have just scanned a badge. The downside of this is that a user could potentially scan many badges when they did not intend to.

**Solution:** The solution taken in Google I/O 2014 to address the Hardware Constraints was to simply display a message to the user saying that this functionality required NFC. This is less than an optimal solution! A fall back method could have been to scan a QR code to access the same information which would have functioned almost the same as
using NFC. Degrading gracefully is an important issue to consider as some users may end up missing out on some functionality. Therefore, the concern of degrading gracefully needs to be understood in the context of mobile app development and considered as a first class citizen when designing an app.

To address the issues of when to enable Sensing the Google I/O required developers to navigate to a screen before enabling the scanning. This assumes that users were well aware of the functionality that they were using. Care had been made to start and stop the NFC scanning in the appropriate times in the app Lifecycle. This meant that the app consumed as minimal resources as possible. Deciding where and how the Sensing will take place influences the architecture of the app which starts to overlap with other concerns in our model such as Threading Model.

**Example 3: Sensing, Lifecycle and Threading Model.**

**Description:** The Google I/O 2014 app uses Sensing to detect other nearby devices at the conference. Once a device has been found the app makes a network request to look up more information about that device. Nearby devices are displayed as markers on a map.

**Issues:** In this case the Sensing aspect of this functionality presents the option to do processing on the mobile device or on the server. The user interface must be updated with the response from a network which requires dealing with the mobile app Threading Model and its state model, Lifecycle.

**Tradeoffs:** To perform the processing on the device, the data for performing the lookup needs to be downloaded to the device prior to any lookups. This creates the need for more data synchronisation and local storage directly impacting upon the architecture of the mobile app. Opting to download the device data also affects the server API design. If the option is to do the processing on the server then sensor values can be sent individually or in bulk which also influence the API design.

**Solution:** The Google I/O 2014 app scans for nearby devices in a back-
ground service which requires careful integration with the rest of the app. Developers have to decide when and where in the app Lifecycle the background service should be started. If the service should be stopped at anytime to conserve power consumption. The developers have opted to make batch requests for more information so as not to make too many requests. In a conference setting with many vendors and attendees the number of network requests could easily escalate. This raises the question “how often should requests to the server be made?”, too few and the information about a vendor will no longer be relevant as the attendee has moved away from them, too many and the benefit of batching requests is lost. A decision like this has implications on the architecture of the server as well – it must be designed to process batch requests rather than single requests.

Users of the Google I/O 2014 app get the most out of the Bluetooth integration when using their phone to navigate through the conference. As such, this functionality does not need to be running as a background service, Threading Model. The user is assumed to be interacting with their device as they use this feature so when a call comes through, or another high priority app starts, the Bluetooth scanning (Sensing) can be put on hold. Running the scanning task on a separate thread to the UI is sufficient.

This example shows that mobile developers often need to address multiple concerns from our technical domain model at once. In this case developers needed to be aware of the implications of Sensing, Lifecycle and the Threading Model to develop the architecture for this feature.

**Key Findings**

In this section we showed how developers address the concerns from the technical domain model and highlighted the need for tradeoffs to be made between competing concepts. We provided examples from Google I/O for the following technical domain concerns:

- Network Connectivity and Threading Model
• Social and Personal, Sensing, Lifecycle and Hardware Constraints

• Sensing, Lifecycle and Threading Model

From this case study we found the following key findings:

• The Google IO 2014 app had four major data sources to deal with Network Connectivity and Threading Model: bootstrapping data to show content to the user when there is no network connectivity, cache to store data from the web services, local database to store content created on the device and a web service to store data in the cloud for synchronisation between devices.

• The developers showed a message for devices that did not support their NFC feature rather than provide an alternative for older devices. Supporting older devices may not be worth it as new phones frequently come out and the mobile ecosystem frequently changes.

• Sensing, Lifecycle and Threading Model go hand in hand when dealing with sensors and performance. The developers took care on where in the Lifecycle they used the NFC scanning and where the sensors are to be used influenced the overall architecture.

• Handling data requirements for Google I/O 2014 app required approximately 21% of the code (provider + io + sync). This amount of code takes a significant amount of effort to develop and would be costly to change at a later point if the wrong design choice was made and technical concerns not well understood.

7.4.3 Threats to Validity

In this section we discuss the construct, internal and external threats to validity for the case studies.

First a threat to construct validity, inferences are made based purely on observation of the source code.

1. What issues the technical domain concerns created?
2. What tradeoffs need to be made between technical domain concerns?

3. How are these tradeoffs and issues addressed?

By studying just the source code inferences can be made as to the answers of these questions but there may be other factors that we are unaware of. For example, question 2 involves developers making a tradeoff that includes other influences that are not in the source code such as timing constrains and developer experience. This creates the problem of experimenter bias. That is the tradeoffs can be reasoned about by the solutions that they implemented but there may be additional concerns that were taken into consideration. One way to address this concern is to analyse the code base and interview the original developers.

One of the threats to internal validity is experimenter bias as the researchers know what to look for and how the technical domain model is to be used. It is unclear as to whether the technical domain model can be used to identify the same solutions, problems and tradeoffs in a repeatable fashion. To overcome this issue the experiment can be repeated with different participants where they use the technical domain model to asses each mobile app.

### 7.5 Industry Case Study: Prompa

We conducted a commercial case study on the output of RAPPT over 2 years to explore the affects of software evolution on the generated code. This evaluation was designed to address the research question, RQ3. How does the technical domain influence the design and implementation of tools for automation? by looking at how 3 main design patterns that we built into the code changed over time. In addition we examined how RAPPT will be used in the development environment of the mobile app technical domain. Both paths seek to identify key areas for future work. This is to ensure that RAPPT addresses the concerns of the technical domain and can be used by practitioners.
The aim of this case study is to assess how a rapid app generation tool would be used in a professional app development environment and how the generated code will evolve. To do this we analysed how our tool was used to build a new app and how the developers integrated the generated code with their hand written code. We also looked at the structure of the generated code and compared it to the code after 2 years of software evolution.

Our case study was conducted on a start-up project, Prompa. Prompa is a mobile app designed to simplify the process of rostering staff on for a work shift. There are two aspects of the mobile app: 1) the app is used by shift managers to roster workers on for a shift and to see who is currently working and 2) workers can see which shifts they have, who the manager is and where they have to be. The API to provide the services to the mobile client had not be developed at the start of our case study.

Prompa\(^\text{12}\) was selected for our case study as it fulfilled the typical requirements of a data-intensive app as described in Chapter 2 namely it satisfied the following:

- **Thin Data-Driven Client.** The plan for Prompa was to build the Android client first and develop apps for other platforms in the future. This meant that the business logic for Prompa was developed on the server so that it could be shared between the different apps resulting in thin mobile clients for each platform.

- **Reliance on Web Services.** Prompa is an app that relies extensively on access to a network to roster staff for a shift, notify staff of a new shift, view available jobs etc. Dealing with technical concerns of Network Connectivity was important as information lost could mean staff would not show up for a shift.

- **Context Aware.** One of the core features of Prompa was the ability for the manager to be notified when staff arrived for a shift through the use of Geofences - the triggering of an event once the user

enters a specific geolocation. The app was also used by staff to register when they had break time and when they started work again. Both of these features required the app to understand the current context of the user.

- **UI and UX Focused.** As this was a new app the owners were keen to make a positive first impression. Complicating the design was the need for certain business workflows to be completed which needed to be met by the design and implementation of the app.

- **Data Storage.** Local storage was of upmost importance for the Prompa app which had a number of features that operated offline but required a network connection for synchronisation. One feature that Prompa had allowed both staff and managers to rate each other. This rating could be done offline but needed to be saved on the device for uploading at a later date.

### 7.5.1 Study Questions

Based on the research question **RQ3** we came up with two sub research questions to focus on for this case study:

- **RQ3.1. How will the generated code evolve over time?** We wanted to explore how robust our code templates were and to see if the generated code captured the core aspects of the software architecture. By answering this question we hope to validate our implementation principle of generating code that resembles code written by a professional developer.

- **RQ3.2. How is RAPPT used in the development environment of the mobile app technical domain?** RAPPT is intended to be used by professional developers so we need to find out how our tool will operate in the correct context of use. Answering this question seeks to validate our assumption that RAPPT can be used in an agile development environment and provides developers with a benefit than starting the app from scratch.


### 7.5.2 Study Propositions

When considering answers to RQ3.1 we came up with three possibilities 1) the generated code provides no benefit to the developers and had to be removed from the code base, 2) the generated domain tactics formed the basis for a robust implementation - generated classes still present with additional methods and removed methods, and 3) the domain tactic addressed the problem sufficiently - the generated code was unmodified. We answered these questions by identifying code that had been generated and comparing with the modified code. Ideally we would find that the tactics had been unmodified (option 3) but option 2 is also a satisfactory result. Although we strove to generate code resembling that of written by a professional business domain requirements may require additional code to be added to the domain tactic classes.

There are a number of facets to RQ3.2 that we wanted to explore in this case study:

- **Integrating Designs.** RAPPT is designed to support the major data-oriented features of a mobile app and does not address the UI and UX concerns. We want to see if RAPPT can be used as the intersection between designers and developers by allowing the later to scaffold the necessary architecture for realising the designs.

- **API Design.** Developers often have to build the backend web services as well as the mobile clients when building a new app. This raises the question would RAPPT be used before the API was finished or would the API be developed first and then the mobile client?

  Our hypothesis for using RAPPT is that the API would be developed first along with the designs. This information would be given to a developer who would use RAPPT to produce the first iteration of the app. From this point the developers would build upon the app adding design details and the final polish.
7.5.3 Case Study Design

Units of Analysis

This research method is as an embedded case study consisting of two units of analysis, source code and method.

To answer RQ3.1, we implemented two domain tactics in RAPPT and generated the first versions of RAPPT using those tactics. We then analysed the source code 3 years later to look at the changes in the implementation of the tactic and to see how the code had been modified.

For RQ3.2, we analysed the app development process used in RAPPT. The primary focus was to use RAPPT as part of the development process and determine how it would be of maximum use. We needed to have the core concepts for a data-intensive mobile app so that the developers could leverage the tool straight away. We collected data through informal observation of the process.

Data Collection

Data collected to answer RQ3.1 consisted of gaining full access to the Prompa repository from 12/09/13 to 15/04/15. This included all versions of the source code, git logs, build tools and final product. By comparing the early versions of the software, the generated app, with the final versions we can see the impact of the evolution of software on domain tactics.

Collecting data for RQ3.2 was more involved as it consisted of participant-observation. The role of the researcher was to use RAPPT and generate an app when the developers required it. This observation consisted of working with the development team and noting how they developed the app for one week.
7.5.4 Findings and Discussion

Impact of Software Evolution

After 2 years of on-going development on Prompa we went back to analyse the initial code base to look at the affects of software evolution on the synthesised code. We found that the pattern and idioms implemented in the code templates and part of the initial generation were still present in the code base. In addition to this we found that our idioms and patterns had been copied to other parts of the code base. Finding the idioms of the generated code still in place is inline with research on software evolution that tells us that architecture does not change significantly over time [64]. The key takeaway from this is that care needs to be taken in specifying the code templates as developers will use the synthesised code as a reference architecture.

Participant-Observation

During the first few iterations the development was of an exploratory nature as the team were designing the web services at the same time as the mobile app. RAPPT was able to mock up Prompa based on the designed web service and the developers were able to validate their design decisions such as the information to display on each screen. A consequence of these rapid iterations meant that RAPPT had to regenerate the app each time there were major changes. To maximise RAPPT’s utility the app developers focused on creating the styles and themes for Prompa independently from the main app during the iterations for the navigation flow and information density per screen.

The developers faced no significant difficulty in integrating the synthesised code with the manually written styles and templates. Also, modifying and adding new functionality to the generated code provide no major challenges to the developers either despite not being familiar with the choice of libraries selected for use in the templates. These examples of working with the generated code indicate that the output from RAPPT is easy to extend. Another finding was that the developers still desired
to use RAPPT after the initial generation which shows that RAPPT is beneficial in a commercial environment but needs to be adapted for use in an iterative development process.

7.5.5 Threats to Validity

Internal Validity: One threat is that of an inexperienced team of developers that were working on Prompa. Throughout the course of the evaluation the developers’ experience and mobile app development skills improved. Experienced developers may use RAPPT in a different way or approach the development of the API in a different manner. Repeating the case study with experienced developers would address this issue.

A consideration to the findings of the impact of software evolution is that of historical events. In a start up like Prompa many different things are happening at once and the product direction can change at anytime which impacts the amount of development work completed: the number of changes impacts software evolution. To address this issue multiple case studies need to be conducted and the results considered along with the number of changes made over the course of the study.

External Validity: By conducting only one industry case study with a start-up it is difficult to replicate the results. One of the problems with this is there is no control case study for any one project to which we can compare results against. In our case, we do not know how Prompa would have been built or what the impact of software evolution would have been without using RAPPT. This is difficult to address in an industrial setting where time is money and developing two versions of the same app is not feasible.

By conducting a case study on using RAPPT in a start-up, building a new API and mobile app from scratch, it is unclear as to whether RAPPT can be generalised to other scenarios. Data-intensive apps often have existing systems to which they connect to provide the data that they need where as a start-up does not have that existing system. To generalise our findings further case studies would be run in different
scenarios: 1) a project that was rebuilding an app, 2) an app that was being developed for an existing system and 3) an app that was being ported to another platform.

7.6 Summary

This chapter described the evaluation of RAPPT and our approach for improving mobile app quality. Our evaluation consisted of building apps from different business domains, a user evaluation with 20 participants, informal interviews with 10 practitioners, case studies and industry case studies. We found that our meta-model captured the core elements required for mobile app development but that the interface to these concepts plays a significant role in the developers perspective.

From our evaluation we found the following key findings:

• RAPPT was beneficial to developers in a commercial environment as they copied the templates that were generated throughout the rest of the code base.

• RAPPT is useful for generating the initial scaffolding for a project but needs to be adapted to work in an iterative development process.

• The Technical Domain Concerns are interrelated and cannot always be satisfied in isolation. For example using the GPS in a mobile app influences the concerns of Sensing, Lifecycle and Threading.

• Technical Domain concerns influence a large part of a system. For example, Google I/O had four major data sources and 21% of the code base was for handling data requirements which relate to Network Connectivity and Threading Model.

• Information captured in our Domain Tactics are beneficial to novice mobile app developers.
Our meta-model for data-intensive mobile apps contains the core abstractions for modelling and generating the scaffolding for new apps.

From the industry case study we found that our architecture scaffolding and domain tactics were still present after 2 years of development. In addition we found that the developers had copied our templates for use in other areas of the software. Conducting the interviews revealed that developers found the information in the technical domain model relevant and of particular benefit to novice developers. Building existing apps showed that we had core concepts required for scaffolding mobile apps but UI and UX related concepts were of upmost importance. These concepts are difficult to include in a tool as they frequently requiring updates to the code templates. In the next chapter we summarise the findings of this thesis and conclude.
Chapter 8

Conclusions and Future Work

8.1 Conclusions

The main contribution of this thesis is a novel, comprehensive technical domain model for mobile apps that guides developers in making informed design decisions.

Mobile app developers now have a new way of thinking about software architecture – viewing architecture in terms of technical implementation concerns. By considering the technical concerns of software, developers can make informed design decisions that otherwise is gained through experience.

We presented the concept of the technical domain as a major influence for achieving quality in software through the use of domain tactics. We looked at how the technical domain influences software architecture and how it influences early design decisions. This enables developers to leverage tried and tested solutions when building new apps preventing the need to rediscover domain specific solutions.

Throughout this thesis, we focused on the domain of mobile app development as a case study for how the technical domain can be defined and analysed by identifying the key concerns of the domain. Based on this foundation we then looked at how the key abstractions of mo-
Mobile app development could be mined into a meta-model. Creating a meta-model is the first step in building MDD based tools for software automation.

Finally, we looked at how the technical domain influences the design and implementation of a tool built for automation. This was done by developing a working prototype, RAPPT, that enabled developers to leverage the technical domain knowledge and generate architectural scaffolding. We also performed an extensive evaluation of our approach that consisted of an user evaluation, commercial and opensource case studies, and interviews with experts. From this research we presented a set of methods and models for understanding the technical domain using the mobile app domain as a case study throughout.

This thesis research has made the following key research contributions:

**Major contributions**

- *Technical Domain Model* – We presented a conceptual domain model for modelling the influences the mobile app domain has on software development. Concepts from a technical domain model may influence other software in the same technical domain depending on requirements. For example, an app that does not make calls to a webservice will not be influenced by network connectivity issues.

- *A method for constructing a Technical Domain Model* – This method enables developers to construct a technical domain model for other domains and analyse the impact the domain has on quality requirements. Furthermore, software reuse and adherence to best practices is encouraged as the concepts are extracted from existing systems. By using this method, developers can clearly see the attributes of the technical domain that will need to be addressed by software architecture.

- *A method for mining reusable domain specific architectural knowledge* – Building upon the method for domain specific influences, we designed a method for mining and documenting reusable architectural domain knowledge. The result of this stage is a set
of Domain Tactics for the technical domain. With this method, researchers can capture architectural knowledge that will assist developers in building software in other technical domains.

• **Domain Tactics** – We presented a novel structure for documenting reusable domain specific architectural knowledge. A Domain Tactic is similar to a pattern except it addresses a specific problem that is present in the technical domain model and includes how the technical domain influences software architecture. We also presented a new tactic catalogue for a set of tactics specific to the technical domain of mobile apps. Domain Tactics provide practitioners with another tool for capturing architectural knowledge in a reusable manner.

• **A meta-model for data-intensive mobile apps** – Our meta-model was derived by analysing existing apps to identify common concepts of data-intensive apps. This ensures that our meta-model contains concepts that describe developed apps. The meta-model forms the basis for a new tool to improve the level of automation afforded to developers.

**Minor contribution**

• **Toolchain** – We developed a new prototype toolchain to realise the meta-model that is used to generate architectural scaffolding for a new project. Developers then build upon these generated apps using standard development tools and methods to add the final polish to the app. The implementation of this toolchain included a number of minor contributions such as:

  - RAPPT – an online tool for developers to describe and generate a working mobile app.
  - A Domain Specific Textual Language (DSTL) used to describe the core features of a data-intensive app.
  - A Domain Specific Visual Language (DSVL) that developers use to specify the high level constructs such as the navigation flow.
These contributions also provide the necessary processes to mine the technical domain specific knowledge for constructing models or building tools.

8.2 Future Work

In this section, we suggest a set of possibilities for future work in the area of domain specific tooling for mobile apps, meta-model construction and domain tactics.

8.2.1 Generalisation

Throughout our study we focused on the domain of mobile app development seeking to go in-depth into a single domain rather than perform a shallow analysis of multiple domains. Our approach has laid the groundwork for analysing a domain’s influence on software architecture. One limitation of our approach is the lack of consideration to the specifics of different technical domains and how they will influence our findings. To overcome this limitation, we need to study other domains and cataloguing domain specific models. These models can then be used to develop advanced domain specific tools that improve development, provide a way for stakeholders to reason about the influence a domain has on software architecture and provide a set of Domain Tactics for reusing domain specific architectural knowledge. Early development work on additional technical domain models for the domain of smart homes and systems that use artificial intelligence are documented in Appendix L.

8.2.2 Automatic Layout and Theme Extractor

One of the key steps in the development of a mobile app is the UI design and navigation flow throughout an app. This process is typically done by designers who are unfamiliar with developer tools and practices. Currently RAPPT does not simplify the transition between design
artefacts such as images of screens to the concrete artefacts that make up a mobile app. As described in Chapter 6, RAPPT provides multiple views for generating a mobile app and structure for specifying the foundations of the app. Future work would seek to increase the level of automation provided by RAPPT by exploring the possibility of extracting layout and theme information from images of the designs and generating developer friendly code that is easy to modify. While UI generation approaches exist these designs are subject to change frequently as UI trends change over time and this will need to be taken into consideration.

8.2.3 Reverse Engineering

After code synthesis and changes have been made to the code it is often desirable to go back to the model, preserving changes in the code, to make modifications at a higher level as highlighted in the user evaluations, Chapter 7. This, roundtrip engineering, feature of MDD based tools is one of the most challenging to get right and maintain flexibility [187]. Our focus has been on forward engineering but significant benefit to developers can be provided once forward engineering has been incorporated into RAPPT.

Incorporating forward engineering into RAPPT presents a few challenges and the needs of the developers editing the code will need to be considered. Forward engineering can involve abstracting code away using abstractions which means developers cannot access all the code that gets generated. Another common approach is to use annotations to mark out what code gets generated which also limits what code can be modified.

8.2.4 Cross Platform

Due to the nature of our research we focused on tool support for a single platform, the Android platform as discussed in Chapter 6. The primary reason for this was to explore the depth of a technical domain and how
these findings influenced software architecture. However, cross platform tooling is of interest to practitioners and developers due to the cost of building an app for each of the two major platforms, Android and iOS. Further work is needed to see if the technical domain needs to be expanded to handle the characteristics unique to each platform.

RAPPT is a model based tool that can be adapted to support multiple platforms but is left as future work. One way to achieve support for multiple platforms would be to create a meta-model for another mobile platform such as iOS. Due to the focus on ensuring the code that gets generated resembles that written by a developer separate transformations will need to be written.

### 8.2.5 Live Visualisation

A consistent theme in the feedback from evaluating RAPPT with professional developers, was the need to see a live preview of the app while they were using the tool, see Chapter 7. This presents some challenging problems both of a research and engineering nature. For example, RAPPT made use of a language which is typically used for ongoing development rather than a throwaway use case like a form. Engineering problems involve representing a mobile app emulator in the browser with sufficient performance to provide live, real-time feedback to the developer. Issues around performance relating to showing updates also needs to be considered as the changes will require the code to be recompiled as not all changes are UI focused.

### 8.2.6 3rd Party Frameworks

Mobile apps rely on a wide range of 3rd party frameworks such as ad networks, analytics, crash reporting etc. These frameworks are essential for developers to build and monitor their apps which is required for the mobile platform to gain popularity. Advanced features such as in app purchases, generating lite or free versions would also provide additional benefit to developers. Future work would involve adding these
features to RAPPT including updating the meta-model and toolchain.

8.2.7 Extensibility

Another theme that came across in the RAPPT evaluation was the need to be able to extend the whole platform, see Chapter 7. There are a multitude of reference architectures for mobile apps online all with different approaches and philosophies. App generation tools need to be able to handle this variability of user preference as well as provide ways in which the tool output can be extended. Over the period of this thesis, the author has observed that mobile UI design patterns change frequently and RAPPT needs to be extendable to stay up-to-date. With every new platform release we found that developers wanted to use the latest libraries and therefore tools need to be update-able by developers to stay relevant.

In conclusion, the technical domain represents concepts that influence software quality and thus the software architecture. As such further study of the technical domain for different systems can greatly enhance our understanding of software architecture. In addition, as our understanding of the technical domain increases, better tools and models can be developed to help developers create high quality software systems.

8.3 Closing Thoughts

In this thesis we have shown the importance of considering the technical domain concerns and the impact the technical domain has on software architecture. We chose to focus on the domain of mobile apps due to their widespread growth and use, ability to concentrate on a single domain, and limited prior work that focuses exclusively on mobile in the literature.

At the start of our research we focused on improving mobile app quality but over the course of our work we realised a much greater need – the identification and study of the technical domain concerns and do-
main tactics. By analysing the characteristics of a mobile app that contributed to software quality we found that the implementation details played a vital role. Addressing these implementation details required skill and experience which up to this point had not been holistically categorised or analysed in a systematic manner.

By understanding the technical domain architects can make informed design decisions and reason about the potential tradeoffs specific to the technical domain. While having a tactic catalogue for a specific technical domain can encourage reuse and adherence to best practices. It is the authors hope that the methods described in this thesis get applied to other technical domains and assist in improving software architecture. To sum up our major contribution, when striving to achieve quality in software the technical domain matters!
Appendix A

Building Data Intensive Mobile Apps

Covered in this appendix is the necessary background information on data-intensive mobile apps and what is required to build an app. This appendix is referred to from Chapter 5 and Chapter 2.

Data-intensive Mobile Apps

A data-intensive mobile app is a) a thin data-driven client that contains minimal business logic, b) relies on webservice for providing content, c) is context aware, d) focuses on providing a high quality UI and UX, and e) store data locally for fast retrieval. These attributes are adapted from the work done on md² [47]. Many of these apps are designed to complete a business workflow such as making a transaction in an e-banking app.

In contrast, games share many of the same concepts as data-intensive apps such as calling webservice and relying on the user’s context. However, they typically have highly customised UIs that do not use standard UI components. Games also have rich interaction models that involve different gestures such as using different gestures to fly an airplane. As such Games are distinct from data-intensive mobile apps that focus on filling out forms and navigating through content. Another type
of app distinct from data-intensive apps are live backgrounds which are intended to display animations on the background of the device. These types of apps do not enable business workflows.

Data-intensive apps typically fetch data from a single web service which is responsible for making requests to other parts of the system. This is a typical client-server architecture style. Therefore, a non-trivial part of a data-intensive app is code responsible for making network requests and handling the data flow through an app. A data-intensive app performs the following set of tasks – format, render, read, send, store, edit, synchronise, validate and delete data. These tasks can be performed either at the server layer and accessed through a web service or be implemented on the device.

**Thin Data-driven Client**

Typically a data-intensive app integrates with a larger system through a web service which may serve content to other apps on different mobile platforms or a web app. In this scenario the business logic is implemented on the server side so that the logic does not need to be re-implemented on each of the client apps. By using this strategy the client mobile apps become a thin-client that is, they focus on user interactions and passing data back and forth between the web service. This presents a set of unique challenges for mobile apps as network connections are inherently unreliable and subject to variable signal strength i.e. a dropped connection when the user is travelling through a tunnel.

**Reliance on Web Services**

Issues around network connectivity strongly influence the usability of the app and therefore the success of the app. Mobile apps are heavily impacted by making network requests, especially when needing to synchronise data across many devices [27]. As such developers need to carefully consider the data in their apps when to make network requests, how often and what to send are all crucial design decisions that can be addressed with a solid robust software architecture. Mobile apps
are also inherently event based which influences the way in which the software app is implemented. For example, the phone can ring at any time or the operating system might trigger an app to shutdown to re-claim resources. All of these events need to be addressed and supported by the architecture.

**Context Aware**

One of the key characteristics of mobile devices is their mobility – users have their mobiles on them all the time where ever they are. This ubiquitous nature of mobile apps has lead to the development of many different apps that can present up-to-date information to users depending on location and time i.e. train timetable app. Displaying fresh content in an app is great for usability but challenging for developers to implement due to the unreliable network connection. For a train timetable app up-to-date information is crucial for the functionality and has different data requirements than a learning app that is not time critical. An appropriate software architecture can be used to address each app’s unique data requirements.

**UI and UX Focused**

Data-intensive apps are also heavily focused on creating a great User Experience and User Interface. These two attributes strongly influence the reviews of an app. During the architecture design process both attributes need to be considered as first class citizens and their impact upon the design considered. In mobile app development the UI design language changes and new trends emerge. As such, data-intensive mobile apps have to be frequently updated to avoid looking out dated which means the UI needs to be easy to modify and change.

**Data Storage**

The inherent non-functional requirements pertaining to storing data are particularly important for data-intensive apps. At some stage data may need to be stored on the device which leads to security concerns.
This is especially true for apps that access private data such as calendar, contact list and account information for other apps on the device. Privacy is another major concern of data-intensive apps particular for social media apps that contain private information and enable content to be shared with other users.

**Building a Mobile App**

Now that we have defined what a data-intensive mobile app is we will go on to describe what is required to build a mobile app.

![Features of a screen from the Eventbrite app that need to be implemented by the developer.](image)

**Figure A.1:** Features of a screen from the Eventbrite app that need to be implemented by the developer.

To illustrate what is involved with developing a modern mobile app we will describe the key features behind a screen from the Eventbrite\(^1\) app shown in Figure A.1. Eventbrite is a data-intensive app that lets people organise and publish an event online. Users of the app can browse

\(^1\)https://play.google.com/store/apps/details?id=com.eventbrite.attendee&hl=en
and register for events that they are interested in attending. Eventbrite also handles the payment for an event. To illustrate what is involved with developing a mobile app we will use the screen shown in Figure A.1 which shows a list of events that are coming up in Melbourne, Australia. Users must sign up for an account with Eventbrite before they can register for events and data can be shown based on the users preferences Figure A.1 (1). To build the list element to hold all of the nearby events (Figure A.1 (2)) developers need to create an adapter to hold the data to display in the element, handle paging of content to improve usability by reducing the load time and create custom view elements to render the content for each row.

Data and Web Services

Data displayed in the screen shown in Figure A.1 (3), is fetched from an online web service provided by Eventbrite. This web service provides a means to authenticate a user, offer search functionality with content stored on the cloud, synchronise user data, register and pay for an event, and display the list of events to a user. Making network requests requires using concurrency so that the user interface is not blocked and the network request takes place in the background. Results have to be sent across thread boundaries and listeners setup for when the results are ready. Error handling is another concern as the user needs to be notified when an error has occurred in the background service. A robust architecture needs to ensure that communication can seamlessly flow between the user interface and the background services undertaking long running operations.

Styles

Apps implement custom styles to provide a unique user experience and make use of themes to integrate seamlessly with the platform see Figure A.1 (4). Each of the major app platforms releases their own style guidelines\(^2,3\) which in some cases are enforced – ignoring these guide-

\(^2\)http://developer.android.com/design/index.html
\(^3\)https://developer.apple.com/library/ios/design/index.html
lines can lead to an app not being released! Mobile platforms such as Android releases support libraries to facilitate the adoption of new design languages\(^4\). These design guidelines are fashionable and change periodically requiring developers to adjust their designs to meet user expectations (Android had Light/Dark, Holo and Material design languages in the passed 5 years). In order to handle these changing styles it is paramount that an architecture be in place that isolates these changes from the rest of the application.

Mobile devices arose from the need to communicate first by text and phone calls and now through a plethora of chat, messaging, video conferencing and social media apps. Eventbrite has integrated with social media platforms by providing an option to share content with other people, Figure A.1 (5). Functionality such as “share” involves deciding what content to share and how to extract that.

**Context Aware**

Mobility is a large part of mobile apps and enable developers to provide content that is relevant to the users current location, time and context. The Eventbrite screen shown in Figure A.1 provides a filter (6) at the top of the screen for the user to select a list of predefined locations. In this instance Eventbrite has set the default location to the users current location. This functionality requires access to a devices GPS ensuring the access takes place in the background so as not to disrupt the user experience. Using sensors present the developer with a number of decisions regarding when to check location, how often and what to do if the sensor is not available on the current device. Ensuring that access to sensors do not disrupt the user experience and the app can degrade gracefully can be addressed by an appropriate architecture.

Eventbrite also provides functionality for users to search through all available events, Figure A.1 (7). Implementing search functionality requires storing data in a database, populating the database and constructing queries based on user input. All user input needs to be

\(^4\)https://www.google.com/design/spec/material-design/introduction.html
validated and checked to avoid injecting undesired content into the database. These queries need to be optimised so that the user does not spend too much time waiting for the results to return. Database design also plays an important role in the design of a mobile app. The relationships between entities and the type of data to store all decisions that need to be made by the architect.

Mobile apps have a primary navigation pattern (i.e. a slide out menu) that is generally accessible on all of the major screens. In our example the primary navigation method is provided by a set of scroll-able tabs shown at (8) in Figure A.1. Depending on the app requirements the navigation pattern may require animations for the screen transitions which have to be implemented by the developer. Navigation through an app is one of the key enablers for a good user experience that depends on the data of the app for deciding what to access and when.

**Social Media and Engagement**

Engaging with users is a key goal of mobile app developers as users play a crucial role in the success of an app i.e. user rankings. One way that developers engage with users is by integrating with 3rd party social media platforms to share content from the app, for free advertising and to provide social interactions. Figure A.1 shows a “Connect with Facebook” button (9) which enables Eventbrite users to see which events their friends are attending. Integrating with social media platforms also provides access to users interests and likes which enables app developers to tailor content displayed in their app on a per user basis.

Data fetched from a web service may need to be formatted specifically for display on a small mobile device. In addition, the data returned by a web service may not match to what needs to be rendered and additional logic may be needed. For example, in Figure A.1 we see a “Free” label at (10) which needs to be determined on the mobile client rather than at the server layer as it is dependent on the type of response returned i.e. is there a cost associated with this event.
Mobile apps are available all around the world for different people and languages. App developers need to be aware of the global reach of their apps and support multiple languages. The mobile platforms have a standard method for internationalising their apps by providing a file to contain translations for all the text in the app (i.e. strings.xml file for the Android platform.

The final aspect of the Eventbrite screen is the bookmarking functionality shown at (11). This requires developing a mechanism for storing user preference on the device so the next time the user visits the Eventbrite app they can quickly access events previously bookmarked. Persistence on a mobile app can be implemented in a number of different ways: internal storage, removable secure digital card, platform specific mechanism or a database. The choice of persistence medium depends on how the data is used, the format and how long the data needs to be available.

In summary, there are many aspects to developing a single screen for a mobile app. To build a quality app a developer needs to address each of these aspects: web services, styles, context aware and social media engagement. Each of these aspects requires understanding the technical domain, are independent of the application problem domain of the app and independent on the technology or platform choices.
Appendix B

Domain Tactics

In this appendix we include the set of Domain Tactics for the mobile app technical domain as discussed in Chapter 4. The tactics are grouped by the domain concern that the tactic helps to address. These tactics are intended to be used by developers when designing their mobile apps and to understand how they can be used together.

Problem 1. Must support many devices with different hardware capabilities.

1. Domain Tactic Name: Degrade Gracefully.

Technical Domain Concerns:

- Hardware Constraints – Not all devices support the latest hardware features i.e. back facing cameras in mobile phones when they first came out.

- Network Connectivity – Push notifications require a network connection to function correctly.

Benefits:

- Usability – Real-time data can be provided to the user when and if
they need it. This improves usability as notifications can be shown to the user as soon as the data is available even when they are not using the app.

- Availability – Content is made available to the user as soon as it is ready.

Liabilities:

- Usability – Abuse of the notification system by triggering too many notifications can cause devices to ring or vibrate which can irritate users. Notifications should be used only for the most important or most current information.

Solution:

- Integrate with a push notification service or use the platform specific APIs.

- Ensure that there is a screen in the app that can manage notifications. Common functionality of a notification screen includes viewing a list of recent notifications highlight which ones have yet to be read and being able to delete 1 or many notifications at once.

2. Domain Tactic Name: Avoid Legacy Platforms

Technical Domain Concerns:

- Hardware Constraints – Hardware features are not built into all devices

- Sensing – Hardware features often involve new sensors

- Ecosystem – New devices frequently come on the market

Benefits:
• Portability and Maintainability – Ecosystem moves along quickly so reduce the amount of legacy code

**Liabilities:**

• Reliability – App won’t be available for all users of target platform

**Solution:**

• Check availability of hardware features with manufacturers and platform vendor.
• Set a minimum supported platform version covering devices with desired features.

**Problem 2. App depends heavily on data stored in the cloud.**

3. **Domain Tactic Name:** Ship Data on the Device

**Technical Domain Concerns:**

• Network Connectivity – App may be opened when there is limited or no network connection

**Benefits:**

• Reliability – App is available even without a network connection

**Liabilities:**

• Suitability – Content showed in the app may be stale

**Solution:**
• Provide data with the app to load and show to the user while new data is being fetched.

4. Domain Tactic Name: Store data on the device (aggressively cache)

Technical Domain Concerns:

• Network Connectivity – Data for the app depends heavily on a network connection.
• Threading Model – Update data using a background service.

Benefits:

• Reliability – App content is available even when no network connection.
• Usability – The app has partial functionality available when there is no connection.

Liabilities:

• Maintainability – Increased complexity as code needs to be written to synchronize data between client and server.

Solution:

• Aggressively cache all API calls to the network.
• Show old content to the user when calls to the network are being made.
• Enable the user to interact with old content when there is no network connection but provide a visual cue to indicate that there is no connection.
• Update content in the background using background services.
• Only make request to the network where absolutely necessary.
Problem 3. The battery drains quickly.

5. Domain Tactic Name: Batch process

Technical Domain Concerns:

- Sensing – App’s algorithms operate extensively on values read from sensors.

Benefits:

- Portability – App is easier to port to other mobile platforms as processing done on a remote server.

Liabilities:

- Reliability – Processing of data is done once remotely which requires round trip to the server.

Solution:

- Send a bunch of sensor readings to a server together for processing.

6. Domain Tactic Name: Limit Sensor usage

Technical Domain Concerns:

- Sensing – App relies heavily on the device’s sensors.
- Hardware Constraints – Battery capacity is drained through excessive use of sensors.
- Lifecycle – Sensors do not always need to be running in every state of an app.
Benefits:

• Performance Efficiency – Prolonged battery life.

Solution:

• Where possible only access sensors when the user directly benefits from the result.

• Limit the time the sensors are enabled by responsibly enabling and disabling sensor usage in the appropriate states of the app’s life cycle.

Problem 4. App frequently freezes or crashes.

7. Domain Tactic Name: Respect single UI threaded model.

Technical Domain Concerns:

• Threading Model – Mobile devices have a single thread for the UI thread.

• Lifecycle – Performing the operations in the wrong state of an app can limit performance.

Benefits:

• Usability – Prevents the mobile operating system from shutting down the app.

Solution:

• Avoid running long running operations on the UI thread and run them in a background service when possible.

• Conserve resources when the user is not using the app.

• Check operations are running in the appropriate lifecycle state.
Problem 5. Users need to authenticate with the App.

8. Domain Tactic Name: Store Authentication Token

Technical Domain Concerns:

- Personal – Mobile devices have a single primary user.
- Lifecycle – Credentials saved and restored in the appropriate states in the Lifecycle.

Benefits:

- Usability – Convenience offered to the user by removing the need to enter text into the screen

Liabilities:

- Security – Once the phone’s lock screen has been passed there is free access to the app.

Solution:

- Store user credentials to avoid unnecessary form filling and skip the login screen when the user next uses app.

Problem 6. App relies on the current user’s context i.e. location, time etc.

9. Domain Tactic Name: Determine User Context

Technical Domain Concerns:

- Sensing – Context of a user depends upon a number of different sensors.
Benefits:

- Functional Suitability – Custom functionality can be provided given the user’s context

Liabilities:

- Performance Efficiency – Reading more sensor values increases power consumption.

Solution:

- Take readings from multiple sensors when developing algorithms for determining a user’s context.
- Track the history of a user’s context to inform the current context.

Problem 7. Multiple platforms need to be supported.

10. Domain Tactic Name: Thin clients

Technical Domain Concerns:

- Network Connectivity – Creating thin clients increases the dependency on the server layer.
- Ecosystem – A wider range of users can be reached by supporting multiple platforms.

Benefits:

- Portability – easier to port the application to additional platforms.

Liabilities:
• Availability – A greater dependency on a network connection.

Solution:

• Develop business logic in the server layer and keep the clients as thin as possible.
• Clients should focus on user experience and usability.

11. Domain Tactic Name: Hybrid approach

Technical Domain Concerns:

• Sensing – Hybrid approaches may have variable levels of support for different sensors on the different platforms.
• Ecosystem – Integration into each ecosystem the app will run on needs to be considered

Benefits:

• Portability – App will run on each of the supported platforms.

Liabilities:

• Portability – App may have variable levels of support on different platforms.
• Maintainability – Adding new functionality can be difficult as the layer between web and native code may need to be extended.

Solution:

• Select a hybrid based framework and design the app as a web application limiting the use of native features.
Problem 8. Data from the app needs to be shared with other people or integrated with other apps.

12. Domain Tactic Name: Integrate with social media

Technical Domain Concerns:

- Social – Content in the app needs to be shared with other people.
- Personal – People are now sharing personal content with others.

Benefits:

- Sociability – More people can engage with content created with your app.

Liabilities:

- Privacy – Content created in the app may appear to a wider audience.

Solution:

- Integrate with social media platforms providing ways to share, comment and like content.
- Ensure that what gets shared to social media platforms is made clear and provide options to disconnect.

13. Domain Tactic Name: Import/Export

Technical Domain Concerns:

- Personal – Personal content can be made available to other apps.

Benefits:
• Sociability – Content can be shared with other devices and vendor lock in is avoided.

• Usability – Apps that import content from another source are easier to get up and running.

Liabilities:

• Security (Privacy)– Content once confined to an app is now available on the device

• Maintainability – Complexity increases for handling different import formats.

Solution:

• Provide an option and suitable formats for exporting the content within an app.

• Import content provided by other apps or sources.

Problem 9. User needs to customise the behaviour of the app.

14. Domain Tactic Name: Implement a Settings screen

Technical Domain Concerns:

• Personal – App needs to provide user preferences to customise the app.

• Ecosystem – Mobile device platforms provide customisable functionality.

Benefits:

• Usability – Users feel more engaged with the app as they can express their preferences
Liabilities:

- Maintainability – user preferences need to be permeated throughout the app.

Solution:

- Add a settings screen to the app to allow users to set their preferences.

Problem 10. Users need to be made aware of fresh content (or new event).

15. Domain Tactic Name: Push Notifications

Technical Domain Concerns:

- Ecosystem – Mobile platforms provide a mechanism to push content to mobile devices.
- Network Connectivity – Push notifications depend on a network connection.

Benefits:

- Usability – Personal information can be made available precisely when the user needs it.
- Availability – Content is made available to the user.

Liabilities:

- Usability – Notification spamming can get annoying.

Solution:
• Integrate with a notification push system and ensure there is a screen to display recent notifications in the app.

**Problem 11. App needs to run on devices with varying screen sizes.**

16. *Domain Tactic Name: Specialised Assets*

*Technical Domain Concerns:*

• Ecosystem – There are many different screen sizes and pixel densities in the mobile ecosystem.

• Hardware Constraints – Screen sizes is strictly a hardware constraint.

*Benefits:*

• Usability – the app is optimally designed for every supported screen of the ecosystem.

*Solution:*

• Find out from the target platforms the recommended supported screen sizes and pixel densities.

• Ensure that each asset in the app conforms to each platforms recommendations.

17. *Domain Tactic Name: Alternative layouts*

*Technical Domain Concerns:*

• Ecosystem – There are many different screen sizes and pixel densities in the mobile ecosystem.

• Hardware Constraints – Screen sizes is strictly a hardware constraint.
Benefits:

- Usability – the app is optimally designed for every supported screen of the ecosystem.

Solution:

- Design different layouts for different screens to take advantage of the available space.
- Different layouts can also be applied to a mobile device in different orientations.

**18. Domain Tactic Name:** Resize Images

**Technical Domain Concerns:**

- Ecosystem - There are many different devices and screen sizes to support

Benefits:

- Maintainability - Reduces the number of images that have to be made and stored on the server.

Liabilities:

- Runs the risk of images not looking optimal for every platform as images need to be resized. This is most likely to occur for screen sizes that are at the extremes of standard devices i.e. very small of very large.

Solution:

- Provide a few image sizes for all devices and resize each image in the app for that device.
Appendix C

RAPPT’s Interface

Below is a description of RAPPT’s web interface describing the major screens and illustrate how the tool is to be used through an usage example based on a MovieDB. RAPPT consists of three screens a Designer for using the DSVL, a code editor for editing our App Modelling Language (AML) our DSTL, and a Code Browser that shows the project structure for the generated output. The rest of RAPPT is described in Chapter 6.

Designer

In order to out explain how RAPPT is intended to be used we will use Peter, our fictitious developer. The first task facing Peter is that of modeling the navigation flow and specifying the screens that make up the MovieDB app. When Peter opens up RAPPT he is prompted for the project name and package for the new app he is going to create. Once he has completed filling out these details Peter is presented with a blank Designer screen similar to the one shown in Figure C.1. The Designer interface consists of 6 sections numbered in Figure C.1 and described in more detail below.

1 This section contains the UI components for the Package, Project and Download. Peter can update the name of the project and pack-

age at anytime by clicking on the input fields contained in this section. Next to these fields is the Download button for downloading the source code for the generated Android app. These three fields are displayed for each of RAPPT’s screens.

2 Navigational Tabs enable the developer to navigate between the Designer, AML code editor and the Code Browser. After Peter has completed specifying the app using the DSVL he can add additional details by going to the AML screen or preview the generated source code before downloading the app by navigating to the Code Browser section.

3 An overview of the keyboard Shortcuts for adding elements from the DSVL to the Visual Editing Pane.

4 The Widget Pane contains the elements that make up the DSVL ready for the developer to drag onto the Visual Editing Pane. For the MovieDB app, Peter needs to drag three screens onto the designer one for the list of movies, displaying the details for a single movie and one for the app’s about screen. Peter also drags the
Tabbar component onto the list and about screens to specify the top level navigation pattern for the app. He then connects the list screen to the details screen specifying the navigation between these two screens.

5 The Visual Editing Pane is where the developer adds elements for describing the app and specifies the navigation flow by connecting screens together. This is the main area of interaction that Peter will use to specify the mobile app. In this area Peter can clearly demonstrate high level app concerns to non-technical stakeholders such as designers or clients.

6 An element’s properties can be updated in the Properties Pane once the element has been added to the Visual Editing Pane. Properties are attributes that are not represented visually such as the element’s ID. After Peter has updated the Visual Editing Pane with the three screens he then updates the title that will be shown for each screen as well as the screen’s ID. It is important that Peter uses clear and easy to understand IDs as these are used in the code generation stage.

Now Peter switches to the AML tab to enhance the description of the MovieDB app by specifying the Movie DB API and screen UI components.

**App Modeling Language (AML)**

The AML section is where Peter uses the DSTL to add details to the screens created on the Designer screen. This section is made up of three major panes: the editor where AML DSL code is written, a Samples Pane that contains source code examples for various tasks to act as a guide to the developer and a Compiler Errors Pane that pops up when the project has errors. These sections are shown on the right side of Figure C.2. We provide descriptions of each section below.

1 DSTL Editing Pane is where the developer edits the generated DSTL code. The developers uses the custom DSTL, AML, to enrich the
details of the app. Here Peter needs to describe the Movie DB API specifying the location of the data, the structure of the data returned, the UI components for displaying a list and details of a movie, configure the event handlers to pass data from the list screen to the details screen and handle authentication. On the Popular Movies screen, Peter describes the API call for fetching the popular movies, the list and the UI components to display the data using the samples as reference. Peter then describes the event handler for clicking on a list item and ensures that the ID for the selected movie is passed to the Details screen. This is needed to ensure that the Details screen shows the details about the correct movie that was selected in the list. In Android development this is known as the Master/Detail UI design pattern [170]. Next, Peter prepares the Details screen to make an API call using the passed Movie ID. Once done, he links the data in the response with the appropriate UI components for rendering to the screen. Peter does not need to worry about error handling, checking for a network connection, logging or dealing with concurrency, as RAPPT handles these by automatically generating the necessary code. Then Peter writes a short copyright notice on the About screen.

2 During the course of editing any syntax errors in the AML app
description will be shown in the Compiler Errors Pane. Semantic errors preventing the generation of an Android app are also shown in this pane. An example semantic error is navigating to a screen that does not exist. These errors guide Peter in specifying the MovieDB app ensuring that the generated code can compile and run.

3 The Samples Pane contains a list of working AML examples with different features such as Maps, Lists and UI components. These samples are intended to encourage reuse through copy-paste so that the developer can extend the DSTL code generated by the DSVL. Samples were created for each feature of AML to demonstrate how that feature should be used and how the different elements of the language can be combined. Peter copies the api block from the Api sample and updates the code to match the Movie DB API. From another sample Peter copies the auth block to the editor and configures the parameters to match the authentication details required for the Movie DB API. These samples provide multiple working examples that contain snippets of AML code and greatly help with development process by reusing already existing code. They will also be useful to beginner programmers, as they can view samples of different ways to utilize various capabilities of the tool.

Now that Peter has finished describing the MovieDB app he is ready to view the generated code before downloading the app and adding the final polish.

**Code Browser**

The Code Browser enables developers like Peter to browse the generated source code in the Code Browser tab for the app before downloading the project. We made a conscious decision not to support round trip engineering so that our approach would not require any artifacts in the generated code such as annotations or additional libraries specifically for using RAPPT. The primary reason for this is that the generated code
Figure C.3: Code Browser. 1. Project Directory Pane and 2. Generated Code Browser.

is the scaffolding for a new project and is intended to be heavily modified. Developers also have the freedom to switch tools and are not tied to using RAPPT once they have started using this tool. Developers extend and polish this scaffolding to produce the final app. A screenshot of the Code Browser is shown in Figure C.3. Different sections of this code browser are marked and described below.

1 The Project Directory Pane shows the directory structure for the generated Android app. This project structure can be imported directly into an IDE for editing code.

2 Uneditable Code Browser displays the files generated by RAPPT. This view enables developers to gain an insight into the generated files. Peter can view all of the generated files for the screens of the MovieDB app as well as the networking code, the choice of 3rd party libraries and the architecture used to ensure that the app satisfies the concerns specific to mobile apps [174].

Now the initial version of the app is ready to download and deploy to a device. Peter downloads the generated code and runs the app on his
device, thankful he does not need to manually write all of the boiler plate code automatically produced by RAPPT. The complete code for building a MovieDB app is shown in Figure C.4 and illustrates which parts of the DSL were generated from the DSVL, copied from the samples section and edited manually.

![Figure C.4: Complete source code for the MovieDB app showing code generated by the DSVL, code reused from copy-pasting AML samples and manually edited DSTL code.](image-url)
Appendix D

User Interviews

The following appendix includes all of the documents for the User Interviews including the demographics questionnaire, interview responses, guide used in the interview process and the transcripts. The User Interviews are discussed in Chapter 7.
Comment report
Lists all the questions in the survey and displays all the comments made to these questions, if applicable.

Table of contents
Report info
Question 1: How many years experience do you have developing (mobile apps)?
Question 2: What is your current role?
Question 3: Gender?
Question 4: How many mobile apps have you worked on?
Question 5: The quality attributes of this Domain Tactic are appropriate.
Question 6: The Domain Concerns for this Domain Tactic are appropriate.
Question 7: This Domain Tactic addresses the Problem Context that it has been assigned.
Question 8: This Domain Tactic is useful for novice developers to know.
Question 9: The Solution description provides enough details to provide a high level overview for creating...
Question 10: Other notes about this Domain Tactic:
Question 11: The quality attributes of this Domain Tactic are appropriate.
Question 12: The Domain Concerns for this Domain Tactic are appropriate.
Question 13: This Domain Tactic addresses the Problem Context that it has been assigned.
Question 14: This Domain Tactic is useful for novice developers to know.
Question 15: The Solution description provides enough details to provide a high level overview for creating...
Question 16: Other notes about this Domain Tactic:
Question 17: The quality attributes of this Domain Tactic are appropriate.
Question 18: The Domain Concerns for this Domain Tactic are appropriate.
Question 19: This Domain Tactic addresses the Problem Context that it has been assigned.
Question 20: This Domain Tactic is useful for novice developers to know.
Question 21: The Solution description provides enough details to provide a high level overview for creating...
Question 22: Other notes about this Domain Tactic:
Question 23: The quality attributes of this Domain Tactic are appropriate.
Question 24: The Domain Concerns for this Domain Tactic are appropriate.
Question 25: This Domain Tactic addresses the Problem Context that it has been assigned.
Question 26: This Domain Tactic is useful for novice developers to know.
Question 27: The Solution description provides enough details to provide a high level overview for creating...
Question 28: Other notes about this Domain Tactic:
Question 29: The quality attributes of this Domain Tactic are appropriate.
Question 30: The Domain Concerns for this Domain Tactic are appropriate.
Question 31: This Domain Tactic addresses the Problem Context that it has been assigned.
Question 32: This Domain Tactic is useful for novice developers to know.
Question 33: The Solution description provides enough details to provide a high level overview for creating...
Question 34: Other notes about this Domain Tactic:
Question 35: The quality attributes of this Domain Tactic are appropriate.
Question 36: The Domain Concerns for this Domain Tactic are appropriate.
Question 37: This Domain Tactic addresses the Problem Context that it has been assigned.
Question 38: This Domain Tactic is useful for novice developers to know.
Question 39: The Solution description provides enough details to provide a high level overview for creating...
Question 40: Other notes about this Domain Tactic:
Question 41: The quality attributes of this Domain Tactic are appropriate.
Question 42: The Domain Concerns for this Domain Tactic are appropriate.
Question 43: This Domain Tactic addresses the Problem Context that it has been assigned.
Question 44: This Domain Tactic is useful for novice developers to know.
Question 45: The Solution description provides enough details to provide a high level overview for creating...
Question 46: Other notes about this Domain Tactic:
Question 47: The quality attributes of this Domain Tactic are appropriate.
Question 48: The Domain Concerns for this Domain Tactic are appropriate.
Question 49: This Domain Tactic addresses the Problem Context that it has been assigned.
Question 50: This Domain Tactic is useful for novice developers to know.
Question 51: The Solution description provides enough details to provide a high level overview for creating...
Question 52: Other notes about this Domain Tactic:
Question 53: The quality attributes of this Domain Tactic are appropriate.
Question 54: The Domain Concerns for this Domain Tactic are appropriate.
Question 55: This Domain Tactic addresses the Problem Context that it has been assigned.
Question 56: This Domain Tactic is useful for novice developers to know.
Question 57: The Solution description provides enough details to provide a high level overview for creating...
Question 58: Other notes about this Domain Tactic:
Question 59: The quality attributes of this Domain Tactic are appropriate.
Question 60: The Domain Concerns for this Domain Tactic are appropriate.
Question 61: This Domain Tactic addresses the Problem Context that it has been assigned.
Question 62: This Domain Tactic is useful for novice developers to know.
Question 63: The Solution description provides enough details to provide a high level overview for creating...
Question 64: Other notes about this Domain Tactic:
Question 65: The quality attributes of this Domain Tactic are appropriate.
Question 66: The Domain Concerns for this Domain Tactic are appropriate.
Question 67: This Domain Tactic addresses the Problem Context that it has been assigned.
Question 68: This Domain Tactic is useful for novice developers to know.
Question 69: The Solution description provides enough details to provide a high level overview for creating...
Question 70: Other notes about this Domain Tactic:
Question 71: The quality attributes of this Domain Tactic are appropriate.
Question 72: The Domain Concerns for this Domain Tactic are appropriate.
Question 73: This Domain Tactic addresses the Problem Context that it has been assigned.
Question 74: This Domain Tactic is useful for novice developers to know.
Question 75: The Solution description provides enough details to provide a high level overview for creating...
Question 76: Other notes about this Domain Tactic:
Question 77: The quality attributes of this Domain Tactic are appropriate.
Question 78: The Domain Concerns for this Domain Tactic are appropriate.
Question 79: This Domain Tactic addresses the Problem Context that it has been assigned.
Question 80: This Domain Tactic is useful for novice developers to know.
Question 81: The Solution description provides enough details to provide a high level overview for creating...
Question 82: Other notes about this Domain Tactic:
Question 83: The quality attributes of this Domain Tactic are appropriate.
Question 84: The Domain Concerns for this Domain Tactic are appropriate.
Question 85: This Domain Tactic addresses the Problem Context that it has been assigned.
Question 86: This Domain Tactic is useful for novice developers to know.
Question 87: The Solution description provides enough details to provide a high level overview for creating...
Question 88: Other notes about this Domain Tactic:
Question 89: The quality attributes of this Domain Tactic are appropriate.
Question 90: The Domain Concerns for this Domain Tactic are appropriate.
Question 91: This Domain Tactic addresses the Problem Context that it has been assigned.
Question 92: This Domain Tactic is useful for novice developers to know.
Question 93: The Solution description provides enough details to provide a high level overview for creating...
Question 94: Other notes about this Domain Tactic:
Question 95: The quality attributes of this Domain Tactic are appropriate.
Question 96: The Domain Concerns for this Domain Tactic are appropriate.
Question 97: This Domain Tactic addresses the Problem Context that it has been assigned.
Question 98: This Domain Tactic is useful for novice developers to know.
Question 99: The Solution description provides enough details to provide a high level overview for creating...
Question 100: Other notes about this Domain Tactic:
Question 101: The quality attributes of this Domain Tactic are appropriate.
Question 102: The Domain Concerns for this Domain Tactic are appropriate.
Question 103: This Domain Tactic addresses the Problem Context that it has been assigned.
Question 104: This Domain Tactic is useful for novice developers to know.
Question 105: The Solution description provides enough details to provide a high level overview for creating...
Question 106: Other notes about this Domain Tactic:
Question 107: The quality attributes of this Domain Tactic are appropriate.
Question 108: The Domain Concerns for this Domain Tactic are appropriate.
Question 109: This Domain Tactic addresses the Problem Context that it has been assigned.
Question 110: This Domain Tactic is useful for novice developers to know.
Question 111: The Solution description provides enough details to provide a high level overview for creating...
Question 112: Other notes about this Domain Tactic:

Report info
Report date Sunday, July 30, 2017 4:22:17 PM EST
Start date Tuesday, May 3, 2016 2:13:00 PM EST
Stop date Monday, May 22, 2017 2:13:00 PM EST
Stored responses 10
Number of completed responses 10

Question 1
How many years experience do you have developing (mobile apps)?
<table>
<thead>
<tr>
<th>RespondentId</th>
<th>StartDate</th>
<th>CompletedDate</th>
<th>LanguageCode</th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
<th>Q5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1274281</td>
<td>2016.05.04 13:36</td>
<td>2016.05.04 13:48</td>
<td>en</td>
<td>5-Jul</td>
<td>Non-mobile App Developer</td>
<td>Male</td>
<td>5-Jul</td>
<td>5(Strongly Agree)</td>
</tr>
<tr>
<td>1274295</td>
<td>2016.05.04 14:19</td>
<td>2016.05.04 14:48</td>
<td>en</td>
<td>&lt; 1</td>
<td>Non-mobile App Developer</td>
<td>Male</td>
<td>N/A</td>
<td>3</td>
</tr>
<tr>
<td>1274308</td>
<td>2016.05.04 15:27</td>
<td>2016.05.04 15:39</td>
<td>en</td>
<td>1-Mar</td>
<td>Mobile App Developer</td>
<td>Male</td>
<td>1-Mar</td>
<td>5(Strongly Agree)</td>
</tr>
<tr>
<td>1274317</td>
<td>2016.05.04 15:53</td>
<td>2016.05.04 16:34</td>
<td>en</td>
<td>1-Mar</td>
<td>Non-mobile App Developer</td>
<td>Male</td>
<td>3-May</td>
<td>5(Strongly Agree)</td>
</tr>
<tr>
<td>1274324</td>
<td>2016.05.04 16:17</td>
<td>2016.05.04 16:48</td>
<td>en</td>
<td>1-Mar</td>
<td>Non-mobile App Developer</td>
<td>Male</td>
<td>1-Mar</td>
<td>4</td>
</tr>
<tr>
<td>1274893</td>
<td>2016.05.11 16:28</td>
<td>2016.05.11 17:5</td>
<td>en</td>
<td>&lt; 1</td>
<td>Non-mobile App Developer</td>
<td>Male</td>
<td>1-Mar</td>
<td>5(Strongly Agree)</td>
</tr>
<tr>
<td>1276592</td>
<td>2016.05.20 15:22</td>
<td>2016.05.20 16:5</td>
<td>en</td>
<td>1-Mar</td>
<td>Mobile App Developer</td>
<td>Male</td>
<td>3-May</td>
<td>5(Strongly Agree)</td>
</tr>
<tr>
<td>1276593</td>
<td>2016.05.20 15:28</td>
<td>2016.05.20 16:10</td>
<td>en</td>
<td>3-May</td>
<td>Mobile App Developer</td>
<td>Male</td>
<td>7 &gt;</td>
<td>5(Strongly Agree)</td>
</tr>
<tr>
<td>1276992</td>
<td>2016.05.24 16:11</td>
<td>2016.05.24 16:53</td>
<td>en</td>
<td>5-Jul</td>
<td>Non-mobile App Developer</td>
<td>Male</td>
<td>7 &gt;</td>
<td>4</td>
</tr>
<tr>
<td>1290881</td>
<td>2016.08.17 20:7</td>
<td>2016.08.17 20:48</td>
<td>en</td>
<td>1-Mar</td>
<td>Mobile App Developer</td>
<td>Male</td>
<td>5-Jul</td>
<td>5(Strongly Agree)</td>
</tr>
<tr>
<td>Q6</td>
<td>Q7</td>
<td>Q8</td>
<td>Q9</td>
<td>Q10 FreeText</td>
<td>Q11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>----------</td>
<td>----------</td>
<td>----------</td>
<td>--------------</td>
<td>----------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5(S)</td>
<td>5(S)</td>
<td>5(S)</td>
<td>3</td>
<td></td>
<td>5(S)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5(S)</td>
<td>5(S)</td>
<td>5(S)</td>
<td>4</td>
<td></td>
<td>5(S)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5(S)</td>
<td>1(S)</td>
<td>5(S)</td>
<td>5(S)</td>
<td></td>
<td>5(S)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5(S)</td>
<td>5(D)</td>
<td>3</td>
<td>4</td>
<td></td>
<td>5(S)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>5(D)</td>
<td>4</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5(S)</td>
<td>1(S)</td>
<td>5(S)</td>
<td>4</td>
<td></td>
<td>5(S)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5(S)</td>
<td>5(D)</td>
<td>4</td>
<td>3</td>
<td></td>
<td>5(S)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Some hardware features can change or become unavailable at runtime. For example NFC. Checking availability at startup doesn't help in this scenario.</td>
<td>5(S)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5(S)</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td></td>
<td>5(S)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>5(D)</td>
<td>5(S)</td>
<td>5(S)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>5(S)</td>
<td>5(S)</td>
<td>4</td>
<td>Missing 'Application interruptions' as a problem. For example, what happens when a phone call comes in while the UI is being updated?</td>
<td>5(S)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>5(D)</td>
<td>5(S)</td>
<td>4</td>
<td></td>
<td>5(S)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Not sure if reliability should be availability in relation to audience restrictions liability. What happens @ the intersection of 'Degrade Gracefully' and 'Avoid Legacy Platforms'? Do we have to rely on the delivery platform to notify users that they can't run the app?

Often the minimum supported platform is defined based on the user base coverage objectives. Supporting older platforms allows to reach a wider user base.

Avoiding legacy or older platforms is counter productive to being able to support many devices with different hardware capabilities.
<table>
<thead>
<tr>
<th>Q17</th>
<th>Q18</th>
<th>Q19</th>
<th>Q20</th>
<th>Q21</th>
</tr>
</thead>
<tbody>
<tr>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Disagree)</td>
<td>5(Strongly Agree)</td>
<td>3</td>
</tr>
<tr>
<td>5(Strongly Agree)</td>
<td></td>
<td>5(Strongly Disagree)</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>1(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
</tr>
<tr>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td></td>
<td>4</td>
<td>5(Strongly Agree)</td>
</tr>
<tr>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td></td>
<td>3</td>
<td>5(Strongly Agree)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td></td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td></td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
While generality is required, I feel the term 'data' does not provide enough guidance re: what to ship with an app.

The problem is that app is unusable until data is fetched, so this will still be prone to this issue on first launch at least which is what I think the problem is specifically about?

What data should be shown for apps that are highly-personalised, such as social media feeds?

This solution is probably specific to the type of data managed by the app. Often data can't be preloaded, especially if specific to the user.

I don't know if this relevant, but I think duration of synchronisation can also be a problem for this domain tactic. Doing background synchronisation when possible can help reduce the duration required to ship data

If the app is reliant on data in the cloud, the data shipped should be very limited.
Q27 FreeText

3

The context in which the tactic would be appropriate are unclear. Domain concerns should include ecosystem.

2

The proposed solution could be in conflict with other tactics, i.e. scenario: no internet. Other concerns are high bandwidth requirements.

5 (Strongly Agree) 5 (Strongly Agree) 5 (Strongly Disagree)

Reliability is a factor but so is availability. I'm not sure if the data (for identical inputs) only needing to be processed once is a matter of reliability or if another attribute should be used.

4

Should 'Network Connectivity' be considered here, or would this be used in conjunction with another solution that falls back on the device for processing?

4

Additional liability: sending data to a server means slower response times and possibly higher battery consumption.

5 (Strongly Agree) 5 (Strongly Agree)

4

5 (Strongly Agree)
<table>
<thead>
<tr>
<th>Q32</th>
<th>Q33</th>
<th>Q34</th>
<th>Q35</th>
<th>Q36</th>
</tr>
</thead>
<tbody>
<tr>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
</tr>
<tr>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
</tr>
</tbody>
</table>

Doesn't mention any downsides in the quality attributes, such as reduced granularity of data etc.

<table>
<thead>
<tr>
<th>Q32</th>
<th>Q33</th>
<th>Q35</th>
<th>Q36</th>
</tr>
</thead>
<tbody>
<tr>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
</tr>
</tbody>
</table>

No liabilities listed; what if quality of data is impacted by less frequent sensor access?

<table>
<thead>
<tr>
<th>Q32</th>
<th>Q33</th>
<th>Q35</th>
<th>Q36</th>
</tr>
</thead>
<tbody>
<tr>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
</tr>
</tbody>
</table>

This solution is only applicable in some scenarios. It also results in reduced dataset to work on.

<table>
<thead>
<tr>
<th>Q32</th>
<th>Q33</th>
<th>Q35</th>
<th>Q36</th>
</tr>
</thead>
<tbody>
<tr>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
</tr>
</tbody>
</table>

App can behave inaccurately because sensors are not sending data as often as it is required.

<table>
<thead>
<tr>
<th>Q32</th>
<th>Q33</th>
<th>Q35</th>
<th>Q36</th>
</tr>
</thead>
<tbody>
<tr>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Q32</th>
<th>Q33</th>
<th>Q34</th>
<th>Q35</th>
<th>Q36</th>
</tr>
</thead>
<tbody>
<tr>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Q32</th>
<th>Q33</th>
<th>Q34</th>
<th>Q35</th>
<th>Q36</th>
</tr>
</thead>
<tbody>
<tr>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Q32</th>
<th>Q33</th>
<th>Q34</th>
<th>Q35</th>
<th>Q36</th>
</tr>
</thead>
<tbody>
<tr>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Q32</th>
<th>Q33</th>
<th>Q34</th>
<th>Q35</th>
<th>Q36</th>
</tr>
</thead>
<tbody>
<tr>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Q32</th>
<th>Q33</th>
<th>Q34</th>
<th>Q35</th>
<th>Q36</th>
</tr>
</thead>
<tbody>
<tr>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Q32</th>
<th>Q33</th>
<th>Q34</th>
<th>Q35</th>
<th>Q36</th>
</tr>
</thead>
<tbody>
<tr>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Q32</th>
<th>Q33</th>
<th>Q34</th>
<th>Q35</th>
<th>Q36</th>
</tr>
</thead>
<tbody>
<tr>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Q32</th>
<th>Q33</th>
<th>Q34</th>
<th>Q35</th>
<th>Q36</th>
</tr>
</thead>
<tbody>
<tr>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Q32</th>
<th>Q33</th>
<th>Q34</th>
<th>Q35</th>
<th>Q36</th>
</tr>
</thead>
<tbody>
<tr>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Q32</th>
<th>Q33</th>
<th>Q34</th>
<th>Q35</th>
<th>Q36</th>
</tr>
</thead>
<tbody>
<tr>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Q32</th>
<th>Q33</th>
<th>Q34</th>
<th>Q35</th>
<th>Q36</th>
</tr>
</thead>
<tbody>
<tr>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
</tr>
</tbody>
</table>
Q37    Q38    Q39    Q40FreeText    Q41    Q42
5(Strongly Disagree)  5(Strongly Agree)  5(Strongly Agree)  5(Strongly Agree)  5(Strongly Agree)  5(Strongly Agree)  5(Strongly Agree)  5(Strongly Agree)

Unclear what to do when network is down but request is 'absolutely necessary'.

4  5(Strongly Agree)  4  5(Strongly Agree)  5(Strongly Agree)  5(Strongly Agree)  4  5(Strongly Agree)

1(Strongly Agree)  5(Strongly Agree)  5(Strongly Agree)  5(Strongly Agree)  5(Strongly Agree)  5(Strongly Agree)  5(Strongly Agree)  5(Strongly Agree)  5(Strongly Agree)  5(Strongly Agree)

Caching is not applicable for all types of data.

4  5(Strongly Agree)  5(Strongly Agree)  5(Strongly Agree)  5(Strongly Agree)  5(Strongly Agree)  5(Strongly Agree)  5(Strongly Agree)  5(Strongly Agree)  5(Strongly Agree)  5(Strongly Agree)
<table>
<thead>
<tr>
<th>Q43</th>
<th>Q44</th>
<th>Q45</th>
<th>Q46FreeText</th>
<th>Q47</th>
</tr>
</thead>
<tbody>
<tr>
<td>5(Strongly Disagree)</td>
<td>5(Strongly Agree)</td>
<td>4</td>
<td></td>
<td>5(Strongly Agree)</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td>'When possible' is pretty ambiguous.</td>
<td>5(Strongly Agree)</td>
</tr>
<tr>
<td>1(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td></td>
</tr>
<tr>
<td>1(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>5(Strongly Agree)</td>
<td>3</td>
<td>ensure all hi updates happen on the UI thread.</td>
<td>5(Strongly Agree)</td>
</tr>
<tr>
<td>5(Strongly Disagree)</td>
<td>5(Strongly Agree)</td>
<td>2</td>
<td>be more detailed. Also, this is probably two different solutions.</td>
<td>5(Strongly Agree)</td>
</tr>
<tr>
<td>5(Strongly Disagree)</td>
<td>4</td>
<td>5(Strongly Agree)</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td></td>
</tr>
</tbody>
</table>
Security is also another concern (i.e. securing credentials stored in memory, securing credentials on local storage)

Seems more of an accessory to its problem context rather than a solution

Probably need some emphasis on security concerns around storage and transmission of credentials. Should clarify that username and password should never be stored directly. A token could be stored locally to address the problem.

It is better to use token to communicate to server once user has been successfully logged in, rather than keeping user details in the device itself
<table>
<thead>
<tr>
<th>Q53</th>
<th>Q54</th>
<th>Q55</th>
<th>Q56</th>
<th>Q57</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

5 (Strongly Agree) 3 4 2 4

5 (Strongly Agree) 5 (Strongly Agree) 1 (Strongly Agree) 5 (Strongly Agree) 5 (Strongly Agree)

5 (Strongly Agree) 5 (Strongly Agree) 1 (Strongly Agree) 3 5 (Strongly Agree)

5 (Strongly Agree) 5 (Strongly Agree) 1 (Strongly Agree) 4 3

5 (Strongly Agree) 4 1 (Strongly Agree) 4 2

5 (Strongly Agree) 3 5 (Strongly Disagree) 3 4

5 (Strongly Agree) 5 (Strongly Agree) 3 3 3

4 5 (Strongly Agree) 5 (Strongly Disagree) 4 4

5 (Strongly Agree) 5 (Strongly Agree) 1 (Strongly Agree) 5 (Strongly Agree) 5 (Strongly Agree)
Q58 FreeText
Seems a bit disjointed and unclear. Not sure what implementation would entail.

Domain concerns should include 'Personalise User Experience'

Q59 Q60 Q61 Q62
5(Strongly Agree) 5(Strongly Agree) 5(Strongly Disagree) 5(Strongly Agree)

5(Strongly Agree) 5(Strongly Agree) 5(Strongly Disagree) 4

5(Strongly Agree) 5(Strongly Agree) 1(Strongly Agree) 5(Strongly Agree)
5(Strongly Agree) 5(Strongly Agree) 1(Strongly Agree) 4

5(Strongly Agree) 5(Strongly Agree) 1(Strongly Agree) 5(Strongly Agree)

5(Strongly Agree) 5(Strongly Agree) 1(Strongly Agree) 5(Strongly Agree)
5(Strongly Agree) 5(Strongly Agree) 5(Strongly Disagree) 4

5(Strongly Agree) 5(Strongly Agree) 5(Strongly Disagree) 5(Strongly Agree)

5(Strongly Agree) 5(Strongly Agree) 5(Strongly Disagree) 5(Strongly Agree)

5(Strongly Agree) 5(Strongly Agree) 1(Strongly Agree) 4
<table>
<thead>
<tr>
<th>Q63</th>
<th>Q64</th>
<th>Q65</th>
<th>Q66</th>
<th>Q67</th>
<th>Q68</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

Where is the balance between server/client?

- Strongly Agree
- Strongly Agree
- Strongly Agree
- Strongly Agree
- Strongly Agree

3  5

4  4

5  3

4  4

4  4

4  2

5  4

2  4

5  4

5  4

5  4

5  4

4  4

1  4

5  4

4  5

4  3
Q69  Q70FreeText

4

5(Strongly Agree) 5(Strongly Agree) 5(Strongly Disagree)

4 'Web application' - unclear if network access is expected. 5(Strongly Agree) 5(Strongly Agree) 5(Strongly Disagree)

5(Strongly Agree) 3

5(Strongly Agree) 5(Strongly Agree) 1(Strongly Agree)

5(Strongly Agree) 5(Strongly Agree) 1(Strongly Agree)

4 Could give examples of frameworks

5(Strongly Agree) 5(Strongly Agree) 1(Strongly Agree)

2

5(Strongly Agree) 5(Strongly Agree) 1(Strongly Agree)

5(Strongly Agree)

Hybrid frameworks are usually suitable only for really simple apps. Users of different platforms expect different UIs and flows from their apps. 5(Strongly Agree)

5(Strongly Agree) 5(Strongly Agree) 5(Strongly Disagree)

5(Strongly Agree) 5(Strongly Agree) 5(Strongly Disagree)

5(Strongly Agree) 5(Strongly Agree) 5(Strongly Disagree)

Limiting the use of native features may prevent apps from feeling that they belong in the eco system of the device 5(Strongly Agree) 5(Strongly Agree) 1(Strongly Agree)
<table>
<thead>
<tr>
<th>Q74</th>
<th>Q75</th>
<th>Q76FreeText</th>
<th>Q77</th>
<th>Q78</th>
<th>Q79</th>
</tr>
</thead>
<tbody>
<tr>
<td>5(S)A</td>
<td>5(S)A</td>
<td></td>
<td>5(S)A</td>
<td>5(S)A</td>
<td>5(S)D</td>
</tr>
<tr>
<td>5(S)A</td>
<td>4</td>
<td></td>
<td>5(S)A</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>5(S)A</td>
<td>5(S)A</td>
<td>5(S)A</td>
<td>5(S)A</td>
<td>1(S)A</td>
<td></td>
</tr>
<tr>
<td>5(S)A</td>
<td>5(S)A</td>
<td>1(S)A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5(S)A</td>
<td>5(S)A</td>
<td>1(S)A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5(S)A</td>
<td>5(S)A</td>
<td>1(S)A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5(S)A</td>
<td>5(S)A</td>
<td>1(S)A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5(S)A</td>
<td>5(S)A</td>
<td>1(S)A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5(S)A</td>
<td>5(S)A</td>
<td>1(S)A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5(S)A</td>
<td>5(S)A</td>
<td>1(S)A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5(S)A</td>
<td>5(S)A</td>
<td>1(S)A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5(S)A</td>
<td>5(S)A</td>
<td>1(S)A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5(S)A</td>
<td>5(S)A</td>
<td>1(S)A</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Portability may be affected by relying on social media integrations on each platform.
<table>
<thead>
<tr>
<th>Q80</th>
<th>Q81</th>
<th>Q82FreeText</th>
<th>Q83</th>
<th>Q84</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 (Strongly Agree)</td>
<td>5 (Strongly Agree)</td>
<td>Unclear how 'export' leads to 'wider audience'; is it implied the export is to a social app? Other apps: an ecosystem concern?</td>
<td>5 (Strongly Agree)</td>
<td>5 (Strongly Agree)</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td></td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>5 (Strongly Agree)</td>
<td>5 (Strongly Agree)</td>
<td></td>
<td>5 (Strongly Agree)</td>
<td>5 (Strongly Agree)</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td></td>
<td>5 (Strongly Agree)</td>
<td>5 (Strongly Agree)</td>
</tr>
<tr>
<td>5 (Strongly Agree)</td>
<td>5 (Strongly Agree)</td>
<td></td>
<td>5 (Strongly Agree)</td>
<td>5 (Strongly Agree)</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td></td>
<td>5 (Strongly Agree)</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td></td>
<td>4</td>
<td>5 (Strongly Agree)</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td></td>
<td>5 (Strongly Agree)</td>
<td>5 (Strongly Agree)</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td></td>
<td>5 (Strongly Agree)</td>
<td>5 (Strongly Agree)</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td></td>
<td>5 (Strongly Agree)</td>
<td>4</td>
</tr>
<tr>
<td>Q85</td>
<td>Q86</td>
<td>Q87</td>
<td>Q88FreeText</td>
<td>Q89</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------------------</td>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>Strongly Disagree</td>
<td>Strongly Agree</td>
<td>Strongly Agree</td>
<td>Concerns re: too many options? Also, most users won’t use a settings screen (I believe), so</td>
<td>Strongly Agree</td>
</tr>
<tr>
<td>Strongly Disagree</td>
<td>3</td>
<td>Strongly Agree</td>
<td>emphasis on sensible defaults might be useful</td>
<td>Strongly Agree</td>
</tr>
<tr>
<td>Strongly Agree</td>
<td>5</td>
<td>Strongly Agree</td>
<td>1 Strongly Agree</td>
<td>5 Strongly Agree</td>
</tr>
</tbody>
</table>

1 Strongly Agree

<table>
<thead>
<tr>
<th>Q85</th>
<th>Q86</th>
<th>Q87</th>
<th>Q88FreeText</th>
<th>Q89</th>
<th>Q90</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strongly Agree</td>
<td>Strongly Agree</td>
<td>Strongly Agree</td>
<td>Not really any information for solution except what is obvious from the title</td>
<td>Strongly Agree</td>
<td>5 Strongly Agree</td>
</tr>
<tr>
<td>Strongly Agree</td>
<td>3</td>
<td>Strongly Agree</td>
<td>not really any information for solution except what is obvious from the title</td>
<td>Strongly Agree</td>
<td>5 Strongly Agree</td>
</tr>
</tbody>
</table>

1 Strongly Agree

<table>
<thead>
<tr>
<th>Q85</th>
<th>Q86</th>
<th>Q87</th>
<th>Q88FreeText</th>
<th>Q89</th>
<th>Q90</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strongly Agree</td>
<td>Strongly Agree</td>
<td>Strongly Agree</td>
<td>Some preferences could be difficult to capture in a settings screen and would be better implemented throughout the workflow and/or</td>
<td>Strongly Agree</td>
<td>5 Strongly Agree</td>
</tr>
<tr>
<td>Strongly Agree</td>
<td>3</td>
<td>Strongly Agree</td>
<td>automatically based on the user's behaviour</td>
<td>Strongly Agree</td>
<td>3</td>
</tr>
</tbody>
</table>

1 Strongly Agree

<table>
<thead>
<tr>
<th>Q85</th>
<th>Q86</th>
<th>Q87</th>
<th>Q88FreeText</th>
<th>Q89</th>
<th>Q90</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strongly Disagree</td>
<td>Strongly Agree</td>
<td>Strongly Agree</td>
<td>Most times user don't even look at the settings screen. Ideally the app should be smart enough to adjust to the user behaviour.</td>
<td>4</td>
<td>5 Strongly Agree</td>
</tr>
</tbody>
</table>

4 5 Strongly Agree

<table>
<thead>
<tr>
<th>Q85</th>
<th>Q86</th>
<th>Q87</th>
<th>Q88FreeText</th>
<th>Q89</th>
<th>Q90</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strongly Disagree</td>
<td>3</td>
<td>Strongly Agree</td>
<td>Not all customisations should be defined within a settings screen, as not all users will go there. Important to allow users to customise within</td>
<td>5 Strongly Agree</td>
<td>5 Strongly Agree</td>
</tr>
<tr>
<td>Strongly Agree</td>
<td>3</td>
<td>Strongly Agree</td>
<td>other screens too.</td>
<td>5 Strongly Agree</td>
<td>5 Strongly Agree</td>
</tr>
<tr>
<td>Q91</td>
<td>Q92</td>
<td>Q93</td>
<td>Q94FreeText</td>
<td>Q95</td>
<td>Q96</td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-------------</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>5(Strongly Disagree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>Lifecycle concerns? How to react in foreground vs. background. Recent notifications screen</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
</tr>
<tr>
<td>5(Strongly Disagree)</td>
<td>4</td>
<td>4</td>
<td>A+++ good sell</td>
<td>3</td>
<td>5(Strongly Agree)</td>
</tr>
<tr>
<td>1(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>4</td>
</tr>
<tr>
<td>1(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>4</td>
<td>Should 'Lifecycle' be another domain concern for cases where processing takes a long time and the user needs to come back after some time?</td>
<td>3</td>
<td>5(Strongly Agree)</td>
</tr>
<tr>
<td>5(Strongly Disagree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>3</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5(Strongly Disagree)</td>
<td>4</td>
<td>5(Strongly Agree)</td>
<td>Content may also be generated by the app, and not always require network connectivity.</td>
<td>5(Strongly Agree)</td>
<td>4</td>
</tr>
</tbody>
</table>
**Q97**: Maintenance liability, makes changing style/branding difficult.

**Q98**: Limitations: Maintainability and storage implications of having different assets per screen variant?

**Q99**: Novice developers frequently miss this.
<table>
<thead>
<tr>
<th>Q103</th>
<th>Q104</th>
<th>Q105</th>
<th>Q106FreeText</th>
<th>Q107</th>
<th>Q108</th>
</tr>
</thead>
<tbody>
<tr>
<td>5(Strongly Disagree)</td>
<td>5(Strongly Agree)</td>
<td>4</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td></td>
</tr>
<tr>
<td>Again, maintenance liability; more effort to develop and alter.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5(Strongly Disagree)</td>
<td>4</td>
<td>5(Strongly Agree)</td>
<td>4</td>
<td>5(Strongly Agree)</td>
<td></td>
</tr>
<tr>
<td>1(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>1(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
</tr>
<tr>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td></td>
</tr>
<tr>
<td>Limitation on usability if the interface is laid out differently for different users, some who might be using multiple platforms simultaneously?</td>
<td>4</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5(Strongly Disagree)</td>
<td>4</td>
<td>4</td>
<td>May reduce maintainability</td>
<td>4</td>
<td>5(Strongly Agree)</td>
</tr>
<tr>
<td>Multiple layouts are usually difficult to maintain</td>
<td>4</td>
<td>4</td>
<td>5(Strongly Agree)</td>
<td>3</td>
<td>5(Strongly Agree)</td>
</tr>
<tr>
<td>5(Strongly Disagree)</td>
<td>4</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td></td>
</tr>
<tr>
<td>1(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
<td>5(Strongly Agree)</td>
</tr>
</tbody>
</table>
Limitations:
If images aren't being served, all apps need to have the highest-resolution images, regardless of whether they support them.
Can iMac performance if too much time is spent resizing images.
Resizing images on the device is usually a heavy operation. To be avoided.

Using vectors will resize and allow for high quality.
Interview Steps

Consent: PRINT Information Statement, Consent Form and Organisational Approval Form

1. Get company to sign Organisational Approval Form.
2. Start Timer
3. Ask participant to read Consent Information Statement.
4. Ask participant to sign Consent Form.
5. Clarify that data will be used for research purposes.
6. Ask for permission to record.
7. Show slide deck.
8. State Objective: To validate a set of Domain Tactics to ensure that the information they contain accurately models the domain of mobile app development and to determine whether they are of use to novice mobile app developers.
9. Show and explain concepts in Domain Conceptual Model.
11. Give Domain Tactics to Participants to read.
12. Clarify and ask if participant has any questions.
13. Ask participant to answer Demographic and Domain Tactic Questions.
16. Ask Problem questions and final questions.
17. Summarise and Clarify major findings.
18. Stop Timer and recording.
20. Thank Interviewee.

Guidelines for conducting and reporting case study research in software engineering

Experiences from Conducting Semi-Structured Interviews in Empirical Software Engineering Research

Concepts:

Sensing captures the characteristic of mobile apps that uses sensors such as GPS, accelerometer etc.

Threading Model represents the concerns of mobile apps single UI threaded model. This means that the user can't run long running operations on the UI thread.

Ecosystem this captures the concerns such as releasing through a 3rd party like Google and Apple. This also represents ad networks, analytics etc.

Network connectivity: going through a tunnel, moving around and intermittent network connectivity

Life-cycle: Apps transition through a lifecycle as the user uses them, into the background and foreground. Concerned with preserving resources.

Personalise User Experience: apps are personal devices that they like to customise, backgrounds. app have single user
Social Interactions: ability to share content, mobile devices for communicating and the integration with social media platforms.

Hardware Constraints: limited screen size, restricted resources and processing power.

Framework for analysing interview questions

Interview questions designed to address one of the following concepts:

1. Depth of Knowledge of participant
2. Importance of captured knowledge (How is the importance determined?)
3. Completeness of the captured domain knowledge
4. Value of captured knowledge (When and how is the knowledge considered?)

Mobile Software Design Problems

Derived from the Domain Conceptual Model.

Common situations that arise when developing mobile apps.

- Multiple platforms need to be supported.
- App depends heavily on data stored in the cloud.
- Must support many devices with different hardware capabilities.
- The battery drains quickly.
- Users need to be made aware of fresh content (or new event).
- App needs to run on devices with varying screen sizes.
- Data from the app needs to be shared with other people or integrated with other apps.
- App relies on the current user's context i.e. location, time etc.
- Users need to authenticate with the App.
- App frequently freezes or crashes.
- User needs to customise the behaviour of the app.

Questions

Domain Conceptual Model Questions

1. Is there anything missing from the domain model? Why?
2. Are there concepts that you would remove from the domain model? Why?

Mobile Development Problems
1. What are the top 5 most important development problems? Add other questions if need be.
2. How did you rank "importance"?

**App Developer**

1. Have you been faced with these development problems at least once when developing a mobile app?
2. Which ones have/haven’t you come across?
3. What other development problems have you encountered that are not listed here?
4. Are there any development problems that you would remove from the list?
5. Have you used these domain tactic before or a variation?
6. Are there any other tactics to these development problems that you have used?
7. Do you use these domain tactics in current apps?

**Non-App Developer Questions:**

1. What do you think of these development problems in relation to other development problems you have come across?
   Other domains
2. Are there any development problems that you can think of that would be handy to know about when getting started?
3. Is there anything that you have learned about mobile app development that you didn’t know before this interview?
4. Would you use these domain tactics if you had to start building mobile apps?
5. Does the conceptual domain model help you better understand mobile app development?

**Both**

Any comments or suggestions for improving Domain Tactics, Problem Contexts or the Conceptual Model: 1. How would you like the information in domain tactics presented to you? wiki, tool etc.

1. Useful for novice developers?
Participant 1

- r: I just have some questions.
- r: Ok, is there anything missing from the domain model or this (problems) and if so why?
- p: Not that I can immediately think of, hard to sort of look at something for the first time and immediately see a hole in it.
- r: yep, is there any of them that you would remove that you don't think should be there?
- p: No, I don't think so.
- r: Yeah, same problem as the first question I guess
- p: yeah
- r: But nothing jumps out so that is good. Ok, so with these (problems) could you pick the 5 most important development problems?
- p: Reuse these right?
- r: I'll give you a bit of paper.
- p: Cool.
- p: It might help to number these.
- r: Yeah sure. Then I can fold it over and give it to the next person.
- r: If you think of another more important problem or just insert it where you think it should be.
- p: sure. This one here "user needs to customise behaviour of the app" what does that mean?
- r: Perhaps they want to have a favourites screen in side the app that quickly navigates to parts of it, certain screens. Or they want to disable settings to make the GPS faster or slower.
- p: ah yep yep yep,
- r: Maybe different times of day the app is dynamic so you have different screens or something
- p: You don't need these in any kind of order just 5.
- r: Yep just top 5. Cool.
- r: How did you rank importance?
- p: I guess, by foremost by my perceived need across projects and how early you would encounter it I guess when building mobile app.
- r: ok, cool. On the form did you put yourself as an app developer or a non-app developer.
- p: I put myself as a non-mobile app developer. I'm not much of a developer now anyway. I can't really claim to be a developer if I haven't written any code in my current position.
- r: Ah fair enough. What do you think of these development problems in relation to other development problems you have come across? In terms of when you are build apps or software.
- p: How do you mean?
- r: Good question. Are they typical, do they apply to other domains or other areas. Or are they specific to mobile apps or have you encountered them in other situations.
- p: yeah I would say I have encountered most of those in non-mobile applications um. battery drains quickly is obviously of a concern for mobile web but not on a laptop that has a lot of resources to deal with heavier processing and is less likely to drain the battery. You know most of those are problems you encounter.
- r: In other situations?
- p: Yep, in like other non mobile app development.
r: yep. What about the solutions? Are the solutions things you think are specific to all apps? all software or do you think there is something in there that is mobile specific?

p: Um, I think a lot of them are fairly generalisable um, It depends I guess on what you consider as non-mobile. Does it include web on mobile that obviously has layouts and assets and certain concerns there. I would say most of these are generalisable um. With the hybrid approach, I mean is desktop web hybrid or maybe it is something worth doing. Otherwise I would say everything here applies.

r: Ok. Are there other development problems that would be handy to know about when getting started?

p: Um, probably the thing that sticks out to me would be um deployment of the app like quick deployment of the app for testing. So I guess immediate portability would be something I encounter offer, like how can we take this from idea to something we stick on this whether it be a mobile device or a web app with different components to it.

r: yep

p: how do we put this together quickly and get a turnaround on it. That's something that is less of a concern with mobile because you just stick it on the app. Then try it.

r: yep.

p: Provided there are no servers deployed as well or the servers are already deployed.

r: Then you can just hit it.

p: Yeah, Otherwise that would probably be the one that would stick out for me.

r: Ok, I'll consider adding it. Is there anything you have learned about mobile app before the interview? Just by reading the domain tactics and the domain model.

p: Um I don't think so. I just think it has been framed in a nice way. I can see a representative spectrum of things.

r: But nothing really new though.

p: nah.

r: Would you use some of these tactics if you had to start building mobile apps?

p: yeah absolutely.

r: Do you think this model can be used to help understand issues around mobile app development?

p: yeah definitely.

r: In what way?

p: Um even just being a reference point for starting projects, I mean you take a certain understanding into every project that you do but this is a disorganised model in your head unless you have something down on paper. From my point of view it would be good to have a sort of a bunch of different boxes coming off as checkpoints for your project.

r: So develop checkpoints?

p: Yeah absolutely.

r: Ok, how would you like this information and these tactics to be presented to you? What would be a good way to present this to say novice developers?

p: Um What are some options for that?

r: Could be wiki, tool, online documentation..

p: I guess for me it would be some kind of documentation accompanying working examples with low overhead to run. So something that is kind of one click setup that would be useful and something where you can throw in a delta and quickly see feedback based on that.

r: yeah that makes sense.

p: so it would probably depend on having an environment where you an reenact some of the problems you faced in here like connectivity issues with client server apps and you know have some control over sensors so you can some what limited what the OS gives you but something where you can say here are some of the problems you will face in
this scenario. I know there are tools that do some of that such as signal network latency and choppiness. Pretty good. So that is probably what I would do documentation with working examples.

- r: Cool. I think that is about it. Ok so before we pack up I'll summarise the main things. You also had feedback on these in general.
- p: I forgot, I think it was just regarding clarity for some of them. I guess in the context of a survey they are a good high level guide. I would certainly want more details to give to a novice developer. Like if I gave these solutions to a novice developer I wouldn't expect them to be able to confidently or certainly go off and implement them all. I would certainly add more details to them I think there was one in there like thin client vs fat server like what do you put on the server and what do you put on the client and how do you sort of consistently strike that balance like is there a protocol for it.
- r: So break it down?
- p: yep.
- r: This is consistent about what you were saying before about more detail with the working examples.
- p: yep.
- r: Ok, And do you think these could be useful for novice developers?
- p: yep for sure.
- r: Ok I think that is it.

---

**Participant 2**

- r: Ok so, questions...Is there anything missing that you can think of from this model (technical domain model)?
- p: Um nothing comes to mind straight away.
- r: Ok
- p: Nothing obvious. Not being an app developer.
- r: Ah ok, just the best that you can understand. Is there anything you would remove? Would you get rid of any of them?
- p: Oh, if I had to pick one of them to get rid of it would probably be threading model.
- r: The threading model?
- p: yeah.
- r: Why's that?
- p: To me it just sounds like it, I guess it reminds me of an implementation concerns the others seems like part of the design perspective.
- r: yep.
- p: The threading model seems like it is managed at a lower level we need to perform this task then whether you do it in the UI thread on in what seems like a low level task. But I'm not sure if you would remove it or if it is worth removing.
- r: Cool. That is good. Ok, Now the interactive stuff, without looking at this folded over piece of paper. Can I get you to list your top 5? Please.
- p: Top 5?
- r: Yep just the top 5.
- p: I'll just read these.
- r: Cool done?
- p: Yep.
- r: Ok, How did you rank importance?
p: Primarily, I was thinking about what I've had to deal with, so what problems have I run into. I don't usually run into freezes and crashes that is fairly but having things like open app and network connectivity not great and nothing works.

r: The app crashes and burns.

p: Yep. I can do literally nothing with this app because for some reason it needs internet. and then things like portability I think are my other concern. Especially if you are going to go down the route of multiple platforms, see like Android.

r: 2, 2?

p: Oh did I? I thought I put something else.

r: A donkey vote?

p: haha. I think it was supposed to be an 8. Yes an 8.

r: 2,3,8,1,6?

p: yep.

r: Non app developer questions. What do you think of these development problems related to other development problems you have had in other domains?

p: Oh in other domains?

r: yep.

p: Um, well yeah I've been doing stuff in distributed computing so I'm sympathetic to issues like network connection problems basically half the problem. What if everything fails? Keeping things alive like different machines synchronised things like that, things like hardware constraints and broader ecosystem are sort of outside of that previous development. Where hardware constraints such as you only have so much performance out of one machine but not like I have to run this on a whole bunch of devices with varying specifications. Using sensors is something I haven't really done before. Social stuff I think is something trivial to implement just need to work out how you are going to design it.

r: Yeah you are right there. Not mind blowingly complex.

p: yeah.

r: Are there other problems to add here that you think would be beneficial to know about when you are starting development?

p: Hmmmm, I think, I don't know how big a problem it is um. I been like, how do I do this or how do I avoid doing this. Anytime you do a login how you would manage that. It was sort of brought up in one of the tactics are we going to let people to login with their social media details and stuff like that.

r: Ok, is there anything you have learned about mobile app development that you didn't know before? Looking over all this stuff.

p: It's terrible! Um, I definitely liked going through the tactics that were more focused on mobile device doesn't support the thing you want to do or you can't do it so often because the battery will just die that kind of thing. Definitely the hardware constraints and the platform constraints were made more obvious to me. Seeing the tactics address these. yeah that is probably the main thing.

r: Cool cool cool, that is good. Would you use these domain tactics if you had to start building mobile apps?

p: yeah I think so. They are all fairly useful some of them are obvious others are yeah lets do that.

r: You would hope so after studying lots of apps! Do you think the model helps you understand mobile apps a bit more? What do you think? Could it be useful?

p: I think as things you need to be concerned about it makes sense. I don't know how much guidance by it in the tactics. But you know looking at it as a guide to apps like if I was to ignore the threading model and lifecycle one, you would need to know how to do it.

r: Yep
p: I've got my idea of how to implement but what are my constraints? I think it would be useful.

r: cool. How would you like this information to be presented? If we had to make this information available to developers.

P: The whole packet?

r: Yes the whole thing.

p: Um, best way is to present it is that everything happens magically and you never have to think about it. Um, possibly..

r: So does that mean you would like the platform vendors to take into account and say hey this is stuff you need to take into account.

p: possibly, just like, have the way you design the app and mandate this is how you have to deal with it. As you design the app it says figure out what to do in that situation make it the way to do it even if there is a hacky way to make it go away. Other than that I'm not sure. I'm not actually sure. I sort think text like this, I don't know probably not the best format. Textbook format is probably not the optimal format but I can't really think of anything.

r: wiki maybe.

p: You want the solutions to be provided to be fairly authoritative..

r: Like, this is the way to do it?

p: yeah, if you had a wiki, you had a place for people to go people might package it differently. Opening up to people or provide people's solutions may be ###.

r: If the solutions can be crap then they will be.

p: yeah

r: Do you think these are useful for novice developers this whole package?

p: yeah definitely, there are things in there that you want to be thinking about when you start designing apps because they are the kinds of things that are painful to add in later on.

r: do you have any examples of that?

p: Um, the obvious one is the network connectivity stuff, if you are going to go for a purely web app you got no pinning. Then this cannot be used without a connection like Telstra is having a bad day or something there will be times when it is not going to be usable at all. Like in terms of get developer to build for multiple platforms or whatever yeah where you store the data. But that might be easier to change later on.

r: Yeah. Just migrate I guess.

p: Yeah. Um stuff like that, even things like ecosystem you have to find your app or do you share the app with people or whatever specialised social services.

r: Yeah cool, I'm pretty much done. Any other comments on this?

p: Um, not really at all. I made some comments on questions in the survey.

r: To summarise: You seem to think this should be authoritative it shouldn't be optional?

p: Yeah pretty much like when you are opening your app store to all these developers a lot of stuff is going to be crap because they don't care and clone existing apps. Then probably end up with crap apps, many of the super popular apps are not going to have many of these problems. um or at least they don't crash most of the time because they are really invested in it I guess.

r: Thanks a lot.

---

**Participant 3**

- r: Alright, so in the domain conceptual model is there anything missing?
p: I don't think so. I mean when I was doing the survey I couldn't help but think that some of them like storing as much data on the device, I mean sending data to the server to deal with so the server deals with it instead of the local application that one I kind of want. No internet connectivity, high bandwidth requirements, maybe there should be a bandwidth bubble. Maybe a security bubble, you touched on security in some of them but I thought it was something that a novice developer should be aware of as a first class rather than just a hey look you should consider security because of this really should be here, at every stage security should be considered when there is private data. Otherwise I thought it captured everything that needed to be captured.

r: Cool, is there anything that you would remove? Would you remove any of them?

p: Remove? No. But if there was a prioritisation I would definitely say some were less of a priority than others. I don't think so, I don't think I would remove any of them.

r: cool.

p: I guess it depends on the app itself, not every app needs social interactions.

r: yep, there is always that, these are app specific. I should have specified this at the start.

p: cool.

r: Now these problems here could you number them 1-5 the most important just the top 5. Write down your top 5.

p: As in what I believe to be most important to least important. An only the top 5.

r: Yep only the top 5, with 1 being the most important.

p: cool. ok.

r: What were the numbers?

p: From most to least important 10,4,6,8,2

r: What did you use to measure importance?

p: Um, I guess previous experience like and personal expectations when I use an app. Like, so for example I wrote 10 as most important which is app freezes frequently or crashes. Obviously I would hate to experience this so to not have it freeze or crash is definitely going to be at the top of my list and then other things you know like draining the battery life. I uninstalled the facebook app specifically because it drains my battery. You don't have to have it open it just has background services that you can't stop. Um 6, vary screen sizes one that is important to me because I'm always amazed at other apps and how well they work on lots of different screen sizes and it is reasonably difficult to make content look, obviously an app is made for a phone and then we don't want to exclude people who use tablets but seeing as it was specifically designed for a phone you don't want the tablet interface to make it look like it was cobbled together you want to make it look like it was designed to work on it. So I think having actual designs that look like they were properly designed is important obviously not the most important. Which is the use the context location time my previous apps that I worked on was pretty complicated with time as data was stored in different time zone and users were in another time zone to make it acutely work consistently was excellent. 2. Was data heavily stored on the cloud so again recent experience everything was stored on the cloud so making sure that communication was clear and well handled. I feel that was important as well.

r: cool, thank you. Did you fill out that you were an app developer in the survey?

p: Yes I did.

r: Have you been faced with these development problems at least once when developing a mobile app?

p: yes

r: Which ones have/haven't you come across depending on? haven't if that is left?

p: Define multiple platforms you mean older versions for example older versions or other platforms like iOS.

r: It could be either.

p: Um I have come across that one. Hybrid capabilities yes, Fresh content I haven't really. I haven't really worked on anything with new content being created every so often. It isn't a multi user environment. I definitely haven't needed to integrate an app with social media, I definitely haven't done number 7, and I have never needed the user to customise
the behaviour of their app but I guess putting that in context the apps that I have developed very generally used in a very specific way and very controlled way. So there was I guess no need to customise the per user experience.

- r: What other development problems have you encountered that are not listed there?
- p: Security. Um specifically, making sure communication with a server is encrypted end-to-end. Um making sure that any information that could potentially be used in a malicious way is stored in such a way that other apps can’t get access still need to work my way if the phone has been root. Yeah, security I think.
- r: Ok, Are there any development problems that you would remove from the list?
- p: From that particular list?
- r: yup.
- p: I mean they are all valid problems. Again I think they are very app or problem specific. I can’t just say remove social media because I have never needed to build an app that uses social media. I can see why it is on there as quite a lot of apps do for example if I you were building a photo gallery app I imagine people would want to share it with facebook, twitter or whatever. There are none that I would remove.
- r: Cool. Have you used these domain tactics before or a variation?
- p: yeah.
- r: Are there other tactics that you have used that aren’t here?
- p: In the actual, would security be considered?
- r: Security is a broad, I think you were right when you said security needs to be put here (domain model).
- p: Ok.
- r: Then the problems related to security need to be added here (domain specific problems) and the solutions.
- p: um, none come to mind.
- r: Do you use these domain tactics in any current apps?
- p: Um, that I’m working on that have been shipped? both?
- r: yeah
- p: Yeah I’ve used them. At least a variation of them.
- r: ok How would you like the information in the model and in the tactics to be presented?
- p: I think there are two problems in that question or two particular scenarios that people would use this information. One of them would be the developer is aware of the problem but doesn’t know how to solve it obviously some sort of ability to sort and it spits out the information. But the other is also informal developers who are not aware of the problems and that one I think is more tricky in which case you obviously don’t want, developers could be seasoned developers who are aware of the problems or novice developers who are not aware. Not introduced to these problems while learning for those I think a very cut down version that inspires curiosity maybe hey I’m you might fine that not every phone has a gyro so you should probably handle that situation. Then they go away and get more information. It makes people think about it and obviously not all of the numbered points will apply to every app I guess you need to have the ability for people who know what they are doing to get the information at hand and also a short but sweet version so that people who just aren’t aware. I imagine those people just want to get into development and they aren’t aware of these problems they will never try to address them.
- r: Do you think this information is useful to novice developers?
- p: Packaged slightly differently, definitely.
- r: How would you package them then?
- p: Um, that is a very good question that I don’t have an answer for. I guess maybe more relatable problems like degrade gracefully I know what it means, but someone who is very knew would go I don’t know what degrade gracefully means just go. Then again if you were to provide a scenario that was even bigger. I guess the way that they are is good but I’m sure there are better ways.
- r: cool.
p: Not very useful.

r: Thanks for that, that's about it.

Participant 4

r: Questions, Is there anything missing from this model?

p: Um, not that I noticed

r: Is there anything that you would remove? That is perhaps not as relevant.

p: Um, no I don't think so.

r: Ok, So just on this bit of paper can you rank these? The top 5. Just the top 5.

p: yes.

r: Just in the importance whatever you define. Yes importance.

p: done.

r: cool, what are the numbers?

p: 10, 4, 9, 5, 2

r: What did you rank for importance? How did you rank importance?

p: Ah from most important to least important.

r: What .. How did you rank importance?

p: I tried to focus on things that would be an absolute show stopper for the app. For example app freezes or crashes you can't use the app so the same with authentication, battery draining quickly people aren't using the app at all. If you can't authenticate then the user can't use the app.

r: it is all over?

p: yep.

r: Cool, did you mark yourself as an app developer or non app developer?

p: Non app developer.

r: Ok.

p: I've done app development a bit of it but I don't currently do it.

r: What do you think of these problems in relation to other development problems you have come across? Like in other domains. How do they compare?

p: I say they would cover like most of the problems you generally encounter. I spend a lot of time in web so you have similar problems multiple platforms and screen sizes and stuff like that. Or even on embedded systems different software different ecosystems, different linux, different version of software installed. so I mean there is a lot of overlap there even outside the realm of app development.

r: So there is a lot of overlap?

p: yeah. I would say so

r: ok, are there any development problems there that would help you if you were getting started?

p: In what sense, like if you were a brand new developer.

r: yep if you were brand new.

p: like what might you need to know about just on this list?

r: no, in general it doesn't have to be just on this list.
p: oh, I suppose things like caching, I mean caching is one thing I don't think new developers would think about straight away. Um, minimise use of sensors and just how much of an affect they have on battery in mobile as well. not always obvious. yeah.

r: ok, is there anything that you have learned about app development before this process?

p: before I started study?

r: yeah before the questions.

p: no don't think so.

r: it was all familiar to you?

p: yeah.

r: Would you use the domain tactics if you had to build an app.

p: Yeah definitely.

r: Why?

p: I've used some already so. A lot of them are standard practice already. so.

r: Do you think this model and these problems helps you understand what is involved in building mobile apps? Or help better understand what is involved in building apps?

p: Yeah, I mean it gives a very broad overview of the different things that are involved. I think most importantly is having a read through of those all the factors that come into play. Obviously it is a lot more to read but this list is good it is a good summary but I think a lot of the quality attributes in there (domain tactics) are important read. I think they um especially for someone who hasn't done it before at least kind of yeah open their eyes to it and what they will encounter.

r: Ok cool, How do you think this information is best presented if we had to make it available to everyone?

p: That is a good question. Um. I don't know I mean a lot of these you could go through and do an exercise for each of them, try and implement a solution in a kind of context I mean that is probably the most effective way for people to actually know it. But if you want to introduce this to people probably just a video that introduces them one at a time talks about them.

r: Ok so video.

p: Or even if summarise for a bit. Maybe written in more um like an article that you would read through. I guess.

r: Cool.

p: It is the kind of content in a dev blog where it is going through all of these core things where it is putting them into the front of your mind. To make you think about them.

r: Ok, do you think this information is useful to novice developers.

p: Yeah.

r: Why?

p: Um just because they are not necessarily things that you think about when you start but they address a lot of problems that you will encounter.

r: ok so just common problems.

p: yeah common solutions to common problems with descriptions of them. And most of the solutions describe enough to get you thinking this is a good way of approaching that.

r: Cool ok, that is the formal questions. Any other thoughts on that?

p: I don’t think I have anymore thoughts.

r: haha ok thanks a lot.
Participant 5

- **r:** Ok so you put yourself down as a non-app developer, yeah?
- **p:** yeah.
- **r:** ok, So is there anything from this that you think is missing?
- **p:** Not that I can really think of. There are a couple of times in the survey where I became a bit fuzzy on the specific meaning of one or more of them or whether some of the other ones could have been included. IN some of the comments yeah, I said should this also be included in the domain concerns.
- **r:** Ah ok so as another additional?
- **p:** yeah,
- **r:** ok cool, can you remember any of them?
- **p:** If you go through I put it down when I found it. Whether or not that is totally relevant of not. Some of them might just be due to a lack of understanding of specific things.
- **r:** They will be interesting to read though, that will be really handy.
- **p:** yeah.
- **r:** Ok so this builds on top of it are there any concepts that you would remove? If you got anything you would add.
- **p:** With this map here?
- **r:** yes with the model
- **p:** Um, no I don't think so.
- **r:** Ok, um. Can you please order these from the most important to the least? Just the top 5 though.
- **p:** ok
- **r:** With 1 being the most important and 5 being the least
- **p:** ok. Yep.
- **r:** cool, 1,3,4,2,9?
- **p:** yep.
- **r:** What did you use as the measure of importance or how did you rank importance?
- **p:** Um. It was sort of a combination of difficulty of implementing or the amount of how time consuming they are to solve. Probably time consuming and frustration.
- **r:** Pain point basically.
- **p:** Yeah pain point. Um I guess, it was also a mixture of from the developers perspective and from the end users perspective. From the developers perspective it is annoying that multiple platforms need to be supported but the battery drain only becomes apparent when people start using it.
- **r:** Ah I see so a bit of both
- **p:** Yeah a bit of both, as a developer I guess I dunno your device is plugged in or running in a simulator or something like that so i guess I just tried to think of it from, overall importance. Hopefully that helps.
- **r:** yeah it does. What do you think of these development problems in relation to other development problems you have come across in other domains? Are they comparable? Are they unique?
- **p:** Given that I primarily do blackened kind of stuff that is usually in an attempt to solve multiple platforms. I guess form my perspective from a developer it is not a huge concern I have to deal with. Because I've picking a platform that solving that problem I don't tend to have that problem if you know what I mean. Data stored in the cloud, yeah I guess accessing data is always a bit of a problem or challenge or whatever. yeah the many devices with different hardware capabilities yeah like my games stuff that is certainly a big consideration especially when you are doing graphics stuff. battery is not something I have had to think about too much.
r: I guess that is because you are at the backend and not at the ...

p: Well yeah see because I'm in the backend or operating in parameters like if there is something in unreal that drains the battery really quick then there is nothing I can do as a developer about it um. I guess a lot of the stuff that I've kinda done projects I've worked on I guess, have not been hugely data driven. There have been some projects but there are many other projects that are operating real time in self contained session or um yeah like data is not central to the problem being solved ok.

r: ok.

p: I guess I've been developing stuff for entertainment or achieving a goal not really always like content consumption or creation. um. Yeah freezing and crashing, authentication is always an annoying problem that I seem to spend a lot of time on.

r: that is a pain then.

p: yesh.: Like there has been things I've worked on before where the authentication part has killed the project. It has been too hard and I've just given up.

r: Are there any other development problems that you think would be handy to know about before getting started?

p: I guess from my point of view it pretty much captures it the main concerns for approaching something. When you are dealing with mobile things like network connectivity, battery and yeah like where the data is coming from going to. Would there be anything else.

r: Are these covered there?

p: I guess something else would be how to um like the evolution of the app how to design around releasing new version of an app. Especially when it interacts with some backend api kind of thing. Like you have written some backend api and you have an ios and android app and you have to leap those in synch with each other and in synch with the API. Trying to manage all of that without breaking stuff.

r: ok

p: evolution.

r: cool

p: Like with the alzheimer's app I had preferences saved and I was considering how that would be affected if I added new stuff. would the thing die if it can't find something that used to be there in a previous version.

r: Um couple more questions. Is there anything else that you learned about mobile app development that you didn't know before this interview?

p: hmmmm

r: maybe reading through the domain tactics.

p: It is all stuff that I have been exposed to without being a mobile app developer. Like even in other domains you still need to cache stuff and not block your main thread and store credentials. I guess maybe one of the things I would not have immediately thought about would be determining user context thing. Yeah you get exposed to a lot of this stuff pretty early in the process even with all this layout stuff one of the first things you are hit with when you are trying to design stuff. Yeah

r: Would you use these domain tactics if you had to start building a mobile app?

p: Yep. Like I said, they are all things I have come across before. Like you mean the tactics, they try to encapsulate everything in there.

r: Do you think this model and the problems help you understand mobile app development better?

p: yeah.

r: How would you like this information to be presented to you or developers?

p: Would this be in the context of an eventual tool I would be using?

r: It could be, it could be a documentation whatever.
p: ok um, I guess like even from this state I get an idea of how everything fits together but if I was going to change diagram for instance I would have subtitles under each one. About what each one to jog my memory when I was looking at it. If it was going to be a tool um I guess yeah maybe if it was yeah it sounds like it would be something that was useful up front when you are starting development questionnaire that you could go through so you could identify what the important things are and the risks for the app you are going to be developing.

d: features or whatever?

p: yeah so like I don't know lets say for instance the tool you go through a wizard or something or just a questionnaire and it asks you "does your app have like need to be social and connect with social media" and then here are the things you need to consider.

d: Would this information be of interest to a novice developer?

p: yeah it hasn't been presented to me in this way like given that like I said before even though I haven't done a great deal of native mobile development. It is all stuff I have come across organically. It is hard to say if on my first day if I was presented with this whether it would be helpful or not. Like even some of the stuff you learn through making mistakes.

d: yeah by doing.

d: yeah like the first time you block main thread you are like I shouldn't block the main thread.

d: yeah that is a good point

d: so maybe if it was presented in a way like kinda shows or highlights in someway why not following this is a bad idea.

d: highlight the negatives maybe?

d: yeah,

d: it is interesting about what you said about learn, because that is one of the big things about software. you learn how not to do it.

d: most of the skills I have acquired as my time as a developer have been through trial and error. So getting to the end of the project and saying I would totally do it another way now that i know all of this stuff. You can't really be told the best way of doing stuff. Like sometimes even when you are told a certain way is a bad way you do it anyway.

d: thanks for that, it has been very helpful.

Participant 6

r: Ok with the concepts in this model is there anything that you think is missing? any major concepts that you have come across.

p: Ah I going to say no. Not that I've experienced at the moment rebuilding the Prompa app. In here it is is essential covered or I have at least thought about it um yeah.

d: If you can't think of any...

p: oh was it just gaps that are missing or suggestions?

d: either anything

p: I was going to say you brought up... there are two things that I wanted to say off the top of my head. Number... 8 problem 7 store credentials for authentication. Really just the way that is done is really important because I think I wouldn't have a problem with that. On Android it isn't too hard to get in there and see what is being stored in memory. Nah any login sensitive data it is more overall how that is going to be implemented it needs to be verified I guess well thought through from a security sense.

d: Ok thanks, I'll add a note here so I can look to it.

p: One more
r: yep

p: Import export, yeah. Um I haven't had much experience, I know on Android there is something called ContentProviders kind of like a public database share what you have stored and grab data. I guess how to use that, it is bit of a grey area for me: I haven't really played with that but um just need to make the developer who uses this overall aware of exposing data through import export but anyone can see it. That bit said when I went to the Android talk, not the one with you, the week before hand a guy from Outware came out. He did a chat that providing he said was providing data through a content provider in the form of an analytics service and it was a bit sketchy with the way it was done.

r: I remember that one. There were security issues with that one.

p: Yeah and it sounded like he mentioned the content provider part then I thought ** everyone can see stuff it was for debugging purposes it depends how integrated it is if you are going to put it up in production then people are going to look at it so it goes back to number 13 it needs to well thought through.

r: ok, I've added a note here I'll go back through.

p: yeah but apart from those.

r: Is there anything you would remove from the model?

p: For a basic person?

r: yeah.

p: Let me think... I don't really think social interaction. Out of the whole lot I would say Social interaction is the least important because at the end of the day you can just share data like you don't need to integrate facebook or social media, it isn't crucial it is an add on but you can always have a button that lets you share data that you think . I would say that would be the weakest link out of them all.

r: Ok with these problems here can you rank the top 5? The top 5 by importance.

p: ok, with importance?

r: yeah.

p: Ok top 5...hard to think. Top 5 most important. I just realised number 10 the app frequently happens

r: They are just problems.

p: Ok very important. Authentication can be optional. Data from the app needs to be shared, battery drains quickly. Ok I've got my 5 now just need to rank them. Going to go with frequently crashing is number 1, 10, number 2 is 4 yeah definitely. 3 ah platform devices. Want me to say them out loud?

r: It is ok, I've recorded them. It doesn't matter it is just to

p: They could be the same the most important

r: What did you rank as importance or how did you rank importance?

p: ah, in terms of. Number 1 was the most important being able to use the app in the first place.

r: cool

p: If the app freezes or crashes then that is most important because no one can use the app. That is self explanatory.

r: yeah that's cool.

p: Then it was just, yeah the most important thing to think about.

r: Have you come across these development problems before?

p: Yes.

r: Ok which ones haven't you come across?

p: Out of all these problems.

r: yes

p: I don't know how to classify, just out of the top 5
r: no out of these (all problems)

p: I haven't really come across customising the behaviour of the app. That is purely optional, I haven't come across...hang on hang on. When I say data doesn't need to be shared with other people I have never had to deal with number 7 a lost of device.

r: Are there any problems that you would add to that list?

p: Any problems I would add to that list...

r: Or ones that you would remove

p: Number 11 is optional, you don't need that especially in the early days later on like when you get a good following and the user expects

r: Is there that you would add or need to add?

p: Anything to add? Any other problems...ah From just non technical perspective going into code I honestly think the top 5 are the most important things you need to think about anything else is less important and could be viewed as optional.

r: Have you used any of these design patterns or tactics before?

p: Ah yes, I have.

r: cool

p: Want me to name them?

r: nah it is fine. Are there any other tactics from these that you, are aware of that you have come across?

p: um, I would say no, yeap

r: Do you use these domain tactics in current tactics.

p: yeah

r: Which ones?

p: Ah, degrade gracefully, not really haven't had to avoid legacy platforms yeh, ship data haven't had to do that, batch process ahhhhhh yes and no

r: ok

p: Say yes, limit sensor usage of course definitely, thin clients no, settings screen yes, push notifications yes, specialise asserts, resize images. Just starting to store data on the devices using cache, respect single UI thread model um and store credentials yes and determine context yes.

r: so it is safe to say a fair few of them?

p: yeah

r: cool, last two questions. How would this information be best presented do you think? To developers like online, teaching tool, wiki, tool or.

p: For me I would probably say it would come down, I go on youtube and visually seeing the screen what is happening and listening to what they say it is the most effective way for me to learn

r: ok cool

p: that being said even if you, I remember when I did the iOS subject a bunch of this stuff got mentioned but at a really high level not going into depth at all. When you get students coming to uni they need to know this stuff.

r: So you would say this is important to new developers?

p: Yeah definitely. You need to know the foundations at the start so you begin with good.. what's the word.. Good habits the way you think about building stuff to start working in the right direction and you will get found out especially when you try to work with other people they will pick up on stuff. Say this is not the right way of doing it but like if you want to make good code then you need to stick with the foundations.

r: So it would be useful to teach this to new students
p: definitely
r: cool thanks thats it.

Participant 7

r: I have some questions. Do you think there is anything missing from the list of major things here? (domain concerns)
p: ahh, I think pretty much covered most of the things I can think of.
r: ok
p: I guess they are pretty broad most things you think of you could put in one of those.
r: Oh I'm happy to hear that. Is there anything there that shouldn't be there that should perhaps be removed from that model.
p: Um, I guess there are a couple of them like, there are a couple that a fairly close together. Sort of interlinked influences of one will influence other one
r: yep
p: Other than that they are all pretty important. There are fairly closely related as well. Part of personalisation is that you fit into your social contacts sort of,
r: yeah the connections there.
p: yeah.
r: cool. So you wouldn't really remove them or remove one of them.
p: I'm not sure.
r: ok
p: It depends on what granularity that you want to go. You could potentially combine them but you wouldn't necessarily.
r: ok, that is a good answer. Alright, so out of these problems I'm going to get you to look at these and list your top 5. Based on importance.
p: Sure.
r: I'm if you could avoid writing on that.
p: Ok, sorry.
r: Nah someone else has already tried that.. So that 3 the top ones.
p: Yeah probably consider those 3 all parts of the same issues.
r: Can you separate them out into 1, 2 and 3rd?
p: Possibly. I guess that order. These 2 would be swapped assuming differences in screen.
r: Alright how did you measure importance.
p: Probably um, mostly based on the frequency of how they come up.
r: Frequency? Cool. Have you faced these development problems before in the top 10? Not the top 10 these here.
p: Yeap, pretty much all of them to some extent or another. So I would say yeah.
r: Any that you haven't come across?
p: No, I probably have come across them all that 2 I haven't come across as often, probably battery draining as it is highly dependent on what you are trying to do as to whether it is likely to train more of the battery and um, the users current context again very app specific if it needs to be able to tell the context to decide what to do. They probably come up less often but definitely come up before.
r: Ok, that is good to know. Alright what other development problems have you come up before that are not listed there. In the context of mobile app development.

p: I think, for um one of the main ones I have come across in a lot of instances is the need for data storage and the transmission from the backend be secured and the use. Alongside authentication encrypting data schemes um encrypting any data actually saved on the device is something else that needs to be done relatively often.

r: that is a good one, I'll think about adding that one.

p: yeah

r: Is there any problems that you would remove?

p: As I mentioned here 1, 3 and 6 are variations on the same issue, especially 3 and 6 has screens are a type of hardware capability to at least merge 3 and 6 together. 1 I guess is slightly different in that it has to do with different operating systems as well as hardware capabilities it is essentially, you are still trying to get around are devices are different.

r: Yeah makes sense. Cool. Alright have you used any of these tactics before?

p: Yes, Do you want me to go through all of them and say or/

r: Um which list is smaller, the list you have used or haven't used.

p: Probably haven't.

r: Ok then maybe just list what you have used.

p: Sure, so degrade gracefully is not something I have ever had to implement.

r: So what number is that?

p: 1. Would have yes, batch process not something I have done for sensor reading. But I have done it for other data gathered through user interaction.

r: ok.

p: sort of yes, sensor usage is something I have had todo. Aggressive caching is something that is relatively rare.

r: Which number is that?

p: 6. The liability of synchronising history is quite high. yes, which is almost unavoidable. Storing credentials, thin clients. Hybrid is something that I haven't had to do. We don't do much of that here so that is probably why.

r: Which number?

p: 11, um social media, import/expoert, yes I have. Um, settings screen pretty common, push notifications yeah.

r: All done?

p: Yeah.

r: Are there any other tactics to these development problems that you would add to the collection?

p: sure, um. There are a couple of them I do have.

r: I mean new ones that you have used that you would add.

p: Yeah, I'm just looking through the problems again.

r: No worries, this might be easier where you can read them all at one go.

p: Ahh um, Sooo I guess one of the strategies around multiple platforms and different data capabilities um I guess comes along the thin client kind of lines just to the effort of um building out a common layer to interface with the backend services through UIs that share the backend piece of code.

r: Is this on the client?

p: Yeah this would be on multiple platforms so you would have a shared code between the different platforms. Then you would have platform specific code?

r: What is the tech for that?
• p: Um so, mostly you would do something like a, J to objective C a library that would take java code and create objective c code. It has a couple of problems where it does simple things really well and other not so well.

• r: And it produces Objective C code.

• p: yeah, haha. I guess now there are probably options around using something like swift and run code on ios and android as that is something that is possible. So that idea of having one set of things that are going to be exactly the same and that is usually going to be how you communicate with the backend service that you have. Then all the stuff you have that is slightly different and different enough that you need to write twice you just write it in the native implementation that you have.

• r: cool

• p: So that is one strategy that you have.

• r: yeah that is good, I like that.

• p: Determine Context.

• r: yep.

• p: I think it is probably all you can say about finding the context is all there is. I guess the extra bit I would put on is the... Developing multiple algorithms, there are already algorithms from various people that use multiple sensors.

• r: ah ok, so just piddy back on them ok.

• p: Um, yeah. the need to authenticate, to have some some first time authenticate with a secure password then having a short pin is a pretty common pattern for that one. Pretty much all the banks use that kind of pattern. Frequently freezes and crashes.

• r: testing right?

• p: I guess so. Customise behaviour I think, just a settings screen. Another option is allowing users to setup an account within the app then you can support multiple accounts or switch between.

• r: oh yeah, I've seen that pattern.

• p: And then you can customise based on how they use it or whatever else you want to do.

• r: Great. Do you use these tactics in current apps, some of these?

• p: um yeah, so I think most of these I use in current apps.

• r: yeah you sort of already went through them right?

• p: yeah, um probably one of the main things would don't do is 3 ship data. I don't think we currently do in our apps, it used to be something done more sort of being done less and less now the connections are a bit more reliable.

• r: ok, So how would you want this information to be presented to you or to a new developer?

• p: haha

• r: a tool, wiki, online, questionnaire

• p: this is tough. yeahhhh, some sort of tool that you could look it up specifically. Possibly also something that would prompt you to look at one of them in detail in however long, week to just say here is something a concept you may not have seen.

• r: Cool, that is a good idea. Do you think this information is useful to new developers to mobile?

• p: yep, pretty much. Most of this is pretty important I would say.

• r: Ok thank you that is everything, I'll stop the recording.

---

**Participant 8**

• r: First question, is there anything missing from this model? The technical domain model.

• p: Not really, I think it is covered by all of them.
r: Ok, are there any concepts that you would remove from the model?

p: I'm not familiar with the social interactions when developing mobile apps because so far building customisable applications. They are general purpose not social sharing stuff.

r: So social interactions if any of them?

p: yeah

r: Ok, mobile development problems these ones here, I'll get you to list your top 5 with 1 being the most important and 5 being the least. Most important however you define that.

p: yep

r: Ok so what were the ones?

p: 1, 3, 4, 10, 2

r: cool thank you. Ok what did you use as your measure of importance?

p: Ah basically be like important be like how it affects the project application you are going to build. like multiple platforms is important if you understand how many platforms you need to support you will justify the functionality of how you will build the application as different platforms have different limits like if you have to support multiple platforms you might start thinking about limiting the functionality in the mobile app and put most of the stuff in the server side so the mobile app won't be doing anything.

r: So the impact the problem has?

p: yeah

r: On the the test did you mark yourself as an app developer?

p: yes, ah no non-app developer.

r: ok. What do you think of these development problems in relation to other development problems you have come across in other domains?

p: some of them are similar but yeah maybe some differences probably the limitations hardware stuff like the screen size. You know some mobile device limitations other than that it is pretty similar to other development problems

r: Ok, they are similar to other...

p: yeah

r: Are there any development problems that you can think about that would be handy to know about when getting started with mobile app development?

p: Um,

r: Is there other problems that you are familiar with other software domains that would be relevant to mobile apps?

p: what else...I think you have covered most of the common problems. I don't think I can think of any more

r: Ok, Is there anything that you have learned about mobile app development that you didn't know about before this session? From reading the domain tactics or the problems there?

p: Um. Probably best project yeah you can combine so the connection with what you are going to do with the application and then... hybrid approach. I think

r: Never done hybrid app development?

p: yeah, I don't even know what kind of problem there can be in the real world. Like how do you need to use the hybrid approach.

r: Do you mean what kind of problems would be suitable for a hybrid approach?

p: yeah, because I guess there are other options and just those two. Yeah just those.

r: Would you use these domain tactics if you had to start building Mobile apps?

p: yes, of course, pretty handy you don't have to waste your time when you just realise you could have done that.

r: yeah so if the info is presented up front in stead of discovered?
p: yeah

r: Does the model help you understand more about mobile app concerns better?

p: um, yes since I already know some of them but yeah for someone who wants to start learning about mobile apps can get the basic idea of how you are going to be building mobile apps by looking at them.

r: ok, So would this information be best presented to a new developer? How would be the best way?

p: the diagram?

r: The diagram, the problems and all the domain tactics.

p: hmm, for me probably attach different problem with the solution we have. Giving this problem that can be solved with this kind of solution being offered and how can we relate these to the diagram?

r: At the moment these concerns are listed inside the domain tactic

p: yeah

r: So the domain tactics can be linked from here and vice versa but we are not sure how, this is all covered in here (domain tactic) and the problem are covered by the domain tactic. But they are all interconnected so we are not sure how to show that

p: probably by showing that, this is all the problems we have and for each different problem we can say like is this kind of solution to different

r: So problem basis

p: yeah


p: For me I guess we wiki would be alright because just space format, getting up the drawings and diagrams. In the solution you get a lot of information.

r: You mean at the moment?

p: yeah at the moment. If we guess I not sure if you can cut out some of the information from that because we need some of that information to help that try spend some quotes or problems. so wiki.

r: ok wiki, um up to the last question Is this useful for a novice developer? for this information to be given to.

p: yes of course, um yes they will at least get the basic idea of how to approach the problem that will be facing. As a novice might not found the problem just yet, and if you already know the solution you spend a lot of time trying to do it and you find that this would fix it this approach

r: ok, that's it I'll stop the recording there are no more questions.

p: ok

---

Participant 9

r: I'll start the recording, so with the domain model with this one is there anything missing? Any concepts you think are missing?

p: Um, no don't think so.

r: not really?

p: nah

r: Any others that you have come across?

p: Nope

r: nope? Is there any that you would remove that are not that relevant to app development?

p: um, not really. Although sensing and network connectivity I was going to say they are potentially the same but not really.
r: ok
p: I mean yeah, they are sort of the same thing with the network connectivity.
r: ah I see what you mean, more of a data connection,
p: yeah alright.
r: What are the top 5 development problems for app developers?
p: going by..
r: Whatever, this is the
p: umm yeah, so probably the first 1, like what is the degree of needing internet access. Um
r: yep.
p: What to do when you can't get data, to get the data to show. um.. Yeah so with sensors like limiting when you are using or optimising.
r: yep just two more.
p: Ah, Yeah I would say UI customisation
r: yep last one
p: Um authentication is always important
r: Ok what did you use to come up with the importance? Just your rough measure, is it something like frequency or what your thought process are on it how you came up with it.
p: Just been my experience developing apps
r: cool, you said you were a non app developer didn't you? You are an app develop
p: not really a mobile app, I mean I'm not doing it now
r: it is ok I'll put you as an app developer. Have you been faced with these problems when developing apps? Those ones (list of problems)
p: Yeah
r: Is there any that you haven't come across or haven't use?
p: Which ones haven't I... um I think I have encountered all of them.
r: That is reassuring
p: yeah
r: um are there any other problems that you have come across that aren't listed there?
p: ummmmm I'm trying to think, I guess hmm I don't know if it is much of a problem. I remember one of the apps we were making we weren't getting the most accurate results with sensors. I don't know if that is worth recording...
r: no that is still worth putting that down
p: that is all I can think of right now
r: nah that is cool, ok are there any problems there that you would remove from the list?
p: As in not common?
r: Ah just you don't think are worth considering as problems?
p: Nah they are all pretty important.
r: Um do you think of these problems when building mobile apps? I guess we have already answered that.
p: yeah
r: ok, Have you used these domain tactics before that we went through? That you went through sorry.
p: yeah
r: Which ones have you used? Or maybe which ones you haven't as you said you have come across a lot of them.
• p: Which ones haven't I used?
• r: yeah
• p: Ummmm I don't think I have done aggressive caching on the device as well
• r: yep
• p: but I would know how to do it
• r: yeah it is just ones that you haven't come across not necessarily ones that you couldn't do or couldn't figure out
• p: ah ok. Alternative layouts, specialise assets
• r: you have probably used them without knowing
• p: yeah
• r: if you have used them on different devices you have used them
• p: ah yeah, resize images probably not. All the rest I think I've done.
• r: Cool, all done?
• p: yeah.
• r: Out of the problems we have are there any other solutions that you have come across? To address those problems?
• p: Ah which problems?
• r: The problems at the bottom of page. Are there other tactics that you have used to overcome that were not there?
• p: I don't think so,
• r: ah ok, if you have used it that's ok. Do you use these tactics on apps you have currently built?
• p: These tactics?
• r: yeah or some of them, I guess so sort of a duplicate question
• p: yeah
• r: Um do you think these are useful for novice developers?
• p: yeah definitely?
• r: Any other comments about the problems or tactics and perhaps how you would like this information presented? Like a wiki a tool or a .. or a way to improve this (documentation)
• p: hmmm a wiki would be good
• r: One thing we are thinking about is part of the documentation with code snippets. So with these problems there would be code snippets for it for each platform.
• p: yeah, like a wiki thing
• r: yeah basically
• p: on wiki, like a computer science thing with code snippets and the approach
• r: the thing with these is that they are quite high level
• p: yeah
• r: so caching is quite involved so we rely on someone who knows how to do engineering to say this is basically how we do it
• p: yeah
• r: cool thanks a lot, that is everything.
r: Alright, is there anything missing from the domain model. Um that is the model I showed you at the start
p: Ummm Not from the model
r: Ok, is there something missing somewhere else?

p: Only on the common situations, I would have the err interruptions that may occur from other applications.

r: Alright are there any concepts that you would remove from the domain model?

p: No.

r: Ok, so from the list of problems there what would you rank as the top 5 most important?

p: Ahh, Degrade gracefully as number 1.

r: So if we were to number them, this is from the problems. The problems are underneath the model

p: Oh those ones?

r: Yep.

p: I'll start with must support many devices with different hardware capabilities.

r: 3rd from the top?

p: yep that one. App depends heavily on data stored in the cloud

r: yep.

p: User needs to be made aware of fresh content.

r: Yes got that one, so just two more.

p: This is more depending on the domain, but app relies on current user context.

r: ok

p: Important,

r: Just one more

p: And need to be able to run on varying sizes as well, screen sizes

r: ok, thank you. How did you rank importance for those 5?

p: I guess, from experience. These are the things that you usually want to tackle first.

r: Ok. With the domain tactics have you been faced with these development problems at least once? When developing a mobile app.

p: Yep.

r: Which ones have/haven't you come across? Depending which is the smaller list.

p: Haha, well we have come across all of these problems except for... data needed to be shared with other apps or social media or people.

r: Which tactic is that? Which tactic did you mean?

p: So this the top list or still related to the model?

r: Ah this to the top list the 18 or so?

p: top list?

r: Yup.

p: Ah,

r: Let me know the numbers of the ones you haven't come across.

p: 11 hybrid approach, 12, integrate with social media, 13 import/export data from other apps.

r: So 11, 12, 13?
p: Yeaaah, I guess 18 actually re-sizing images on the device itself. That is what you are referring to aren’t you, resizing the images on the device itself?

r: It could be either, it is just that strategy. Whether you do it client or server is irrelevant.

p: Yep. Those 4 I haven’t come across but everything else we have come across before.

r: Ok, cool. So back to the list at the bottom of the model. What other development problems have you encountered that aren’t listed there?

p: Development problems?

r: Or problems that are common when you are developing mobile apps?

p: Ummmm, I guess the other problem we have come across is timeliness?

r: Timeliness? What do you mean by that?

p: So when we running on hardware that is not very performant to make sure your app is still running reasonable speed. So when you are chaining between screens, making sure that doesn’t take up too much time or any other heavier behaviour like calculations that need to be done. Making sure the still perform well on less performant devices.

r: So this is a performance related to hardware issue or a real time issue?

p: Well it is real time but running into performance issues on lower spec devices. Something we came across for a Malaysian Bus tracker app, it was all intended to run big devices and we noticed that some of the hardware isn’t very fast and needed to make sure things ran fast enough without the user being frustrated.

r: Ok, with these development problems is there any that you would remove from the list?

p: Ummmm, I wouldn’t specifically remove it but I would not be overly concerned about the last one.

r: Ok.

p: It is not always relevant, it may be depending on the application but I don’t think it is something that should be highly considered.

r: So is it from an importance or a frequency in which it comes up or how do you mean?

p: Ahhh, for me from what I have experienced it is the frequency of things coming up.

r: Ok.

p: So the importance of being able to customise, there are things you might want to customise but yeah most of the time users are happy with what is given to them and they get used to how the app is and they don’t require it to be customisable.

r: ok, so frequency.

p: yup.

r: Now relating to the tactics as a whole that you have come across. Have you come across the domain tactics before or a variation of them?

P: What do you mean by that? As I have been developing apps have I come across these issues before?

r: This is related to the tactics above so the 18 or so

p: yep

r: Have you use some of them before or a variation?

p: Yes

r: Or which ones haven’t you used?

p: Umm

r: I think this is a duplicate question

p: Yeah.
r: Ok, let me move to the next one. Are there tactics to the problems and the ones you have added, are there tactics that could be added to the list above?

p: Ummm, No I think the tactics are covering the issues. That you have outlined. I can't think of any other specific tactics that I would add.

r: What about, are there any types of solution that you have used multiple times to solve a problem?

p: Umm, these are already defined in the tactics that you have.

r: Ok, so they are variations of what you have already done?

p: yeah, yup.

r: Um, ok. So do you use these tactics at the moment in current apps?

p: yes, a variation of.

r: Ok, Which ones do you use at the moment?

p: Umm, tactics such as um supporting legacy platforms as an example.

r: Ok.

p: Looking at what the functionality is of your application and showing that you are not releasing any code for platforms that are not able to support it for example.

r: What do you mean?

p: Being aware of what you are building is capable of running on the devices that you are releasing it to.

r: Ok that makes sense.

r: If you have any other things about this let me know.

p: I guess something that we don't do enough is to aggressively cache.

r: That is interesting to find out.

p: A lot of the applications I've worked on possibly require network connectivity so when you come back into the information will have to get fetched again.

r: So the network connectivity is part of the domain model.

p: yes. but that is important. A tactic we have never used before is the hybrid approach. We have not used any hybrid based application where ... or some portion of it using a web UI.

r: No applications have used a webview or a web UI?

p: No

r: That is interesting. Another question I have about, any comments you have for improving the domain tactics or list of problems that are there or the conceptual model. What do you recommend to improve it? One way to look at is how would you like the information in the tactics to be presented to you? A tool, wiki, blog? Video?

p: I guess, the problems are well defined and the concerns are also mainly well defined as to what some of the benefits and liabilities are the solutions are a little bit limited I guess.

r: What do you mean by limited?

p: Well the solutions are a suggestion as to how you want to solve that problem um but it doesn't include information as to how you would approach the entire way of solving that problem I guess. I would have more of a wiki with a bit more information as to how you would solve that specific problem with more than just a one liner. If that makes sense.

r: So you are talking specifically about the solution section of the domain tactics?

p: yes correct?

r: Ok

p: So additional information to the solution. It doesn't require it but others you might want to include others as to how you would approach this solution.

r: Thanks for that. One last question are these tactics, this model and problems useful for new or novice developers?
p: Yes most of them definitely are. Ones that might not be relevant for novice developers are things like resizing images for example.

r: Ok.

p: Umm an important point to consider is but this is more of an overall approach to you business model I guess.

r: What do you mean by business model?

p: Well if you look at most of your tactics here all tactics are specific to mobile app development where the thin client component for multiple platforms need to be supported, this is something almost outside of mobile development. We aren't just talking about something that is limited to the mobile devices themselves also the architecture of things outside of the mobile app. So although it is good to consider for novice developers it is not something they should be intently worried about if that makes sense? r: Yes that makes sense. Ok cool. Thanks for that I'll stop recording now.
Appendix E

Ethics Application

Included in this appendix is the ethics documents for the SHR Project 2014/313 including the ethics application, ethics approval, participant consent forms, ethics addendum, ethics addendum approval, addendum consent forms, final ethics report and change of thesis title. This ethics application covered all of the user evaluations covered in this thesis. The Ethics Application is referenced in Chapter 7.
APPLICATION FOR ETHICS APPROVAL of a RESEARCH PROTOCOL

SECTION A: GENERAL INFORMATION

Nb This application form should not be used for research involving clinical trials or ionising radiation. See below:**

PROJECT FULL TITLE
Bootstrapping Mobile App Development

SHORT TITLE
(If applicable)

APPLICANT DETAILS
RESPONSIBLE SWINBURNE FIRST INVESTIGATOR / SUPERVISOR
(Where project is part of student research degrees or dissertations, Senior Swinburne Supervisor must still be listed as the first investigator)

Name & Title/Position: Dr. Raj Vasa Senior Lecturer Deputy Head, R&D Software Group
Tel No(s) +61 3 9214 5011
Email rvasa@swin.edu.au
Fax -
Faculty / School / Centre / Institute: FSET
Swinburne Status: □ Swinburne Staff Member □ Adjunct Staff Member
Address for correspondence: EN514b

Main Student Investigator(s): Scott Barnett
Email sbarnett@swin.edu.au
Tel No(s) -
Student ID Number 6160506
Fax -
Degree Being Undertaken: PhD in IT

List below the names of other Chief/Associate Investigators and Research Assistants (including those with access to identifiable data). (Add (copy/paste) cells as required for additional investigators/assistants. Append Student lists for class projects.)

Name & Title: John Grundy Dean, School of Software and Electrical Engineering
Institutional Address: EN501c
Tel No(s) +61 9214 8731

Institutional Address:

Proposed Period During Which Human Research Activity Requiring Ethics Approval is Needed:

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
</tr>
</thead>
<tbody>
<tr>
<td>05</td>
<td>01</td>
</tr>
<tr>
<td>dd</td>
<td>dd</td>
</tr>
<tr>
<td>08</td>
<td>08</td>
</tr>
<tr>
<td>mm</td>
<td>mm</td>
</tr>
<tr>
<td>2014</td>
<td>2016</td>
</tr>
<tr>
<td>yyyy</td>
<td>yyyy</td>
</tr>
</tbody>
</table>

[Double-click on YES/NO ‘check box’ to select box, then enter Default Value as Checked □ or leaving as Not Checked □ ]

TYPE OF ACTIVITY
(Select as many boxes as applicable)

□ Research by Staff Member
□ Supervised Postgraduate Research
□ Contract Research (Attach copy of contract)
□ Supervised Undergraduate Research
□ Supervised Class Projects:
No of students involved:

Subject Code & Short Title:

Broad Category of Research
Select one category box which best fits the application:

□ Social/Cultural/Humanities
□ Business/Management
□ Education/Training/Program Evaluation
□ Psychological/Brain/Neuro-sciences
□ Health/Safety
□ Engineering/Science/Technology
□ Other (please specify) ………………………………………………………

[** For research involving Clinical Trials or Ionising Radiation, please contact the Research Ethics Officer.]
Human Research Risk/Review Classification (Nb Checking to be consistent with published risk criteria.*)

To enable a determination as to whether prima facie your research activity is Minimal Risk and/or Low Impact, please clarify by selecting [X] any one or more boxes below as to whether your research activity involves:

[ ] Vulnerable participants, children or those dependent on care*  [ ] Indigenous Peoples* or Special Cultural/Ethnic groups
[ ] Externally funded research requiring HREC-level clearance*  [ ] Multi-centre/Other sites requiring HREC-level approval*
[ ] Research conducted overseas  [ ] Conflicts of interest or dual researcher-professional roles
[ ] Data access/use without an individual's prior consent*  [ ] Data access/use subject to statutory guidelines &/or reporting*
[ ] Identification of participant individuals/groups in research outcomes without full consent or there is unclear consent for this*
[ ] Sensitive information/issues vis-à-vis context/impact (legal*, regulatory compliance*, commercial, professional, cultural, etc)
[ ] Personally intrusive/confronting or quite inconvenient/embarrassing questioning or other activity
[ ] Physically confining/invasive techniques or significant physical contact/stimulation (TMS*, X-ray*, CT scan*, MRI*, clothing change, etc)
[ ] Handling hazardous substances (eg, asbestos*, radioactive material*, explosives*, etc) or equipment
[ ] Administration of medical/herbal substances*/treatments*  [ ] Administration of other (non-medical) substances/treatments
[ ] Health/medical diagnosis*/therapy*  [ ] Non-minimal impact therapeutic or other devices*/activity*
[ ] Screening for healthy participant inclusion/exclusion  [ ] Medical or psychiatric assessment/conditions*
[ ] Serious psychological profiling, investigation or exploration  [ ] Withdrawal of treatment/services or use of placebo
[ ] Withdrawal/substitution of educational/professional/commercial/recreational/other programs or services
[ ] Deception or covert observation  [ ] Limited or non-disclosure of research information/procedures
[ ] Participant recruitment/selection via third party  [ ] Human research activity commenced without clearance
[ ] Participation incentives, prizes or significant payments  [ ] Research placing researchers/assistants at risk

PLEASE NOTE: If you have selected any one or more of the above boxes, your project will ordinarily be put for SUHREC ethical review. Items above marked * must be put to SUHREC proper. But in other cases, you may wish to put a case for expedited review by a SUHREC Sub-Committee (SHESC) in the (expandable) box below in relation to the criteria for determining risk/impact. If you put forward a case, then in the first instance your application will be put to the relevant SHESC; however, the relevant SHESC may still consider the project needs full SUHREC appraisal or SUHREC may review or override the SHESC decision.

Risk/Impact Checked with a Research & Ethics Advisor (REA)?  Yes [ ] No [ ] REA Comment, Initials & Date:

---------------------------------------------------------------
**A1 WHY IS THE PROJECT TO BE UNDERTAKEN**

Summarise in sufficient detail why the project is being undertaken. If references are quoted, full citations should be given. Include the educational and/or scientific aims of the project. (boxes will expand for your text)

| In 2015 the mobile app industry is estimated to be worth $25 billion[1] and there are well over 1 million apps available for each of the major mobile app platforms[2], many new apps are released daily. These apps face plenty of competition and their developers need to make a favourable first impression. Unlike traditional software distribution methods, apps are distributed through an app store[3][4] and are readily available for download. More importantly, users rank and review these apps and readily highlight any issues with an app such as frequent crashes. Success of an app, lots of downloads, is often riding on favourable reviews. Recent research shows that generally these rankings do not change over time[5] and there is emerging evidence that mobile app users have short attention spans[6], together this creates pressure on developers to produce high quality apps early in the lifecycle -- in effect they may not have the luxury to fail early or slowly improve quality. It is well understood that quality attributes are satisfied through software architecture[7]. The architecture of a mobile app is influenced by the type of app that is built. There are three types of apps that can be developed: a web app using pure web technologies, a native app that uses the SDK (Software Development Kit) provided by a mobile platform or a hybrid of two approaches, typically a web container that allows some access to the underlying platform's SDK. The choice of the type of app to develop is determined by a range of constraints including technical, the cultural background of engineers and the business requirements. After initial analysis, a significant proportion of teams choose to build a native mobile application. This is typically motivated by UX (user experience) or performance objectives. In this scenario, to start development of an app, engineers can either: a) Use the project structure generated by an IDE that contains all the build files and one or two generated files with code, b) Use custom generation tools that provide a wider range of project types and libraries, i.e., Android bootstrap or a script developed in-house, c) Use code from a similar project as it will contain relevant custom/third-party libraries and architectural patterns, or d) Use a MDD (Model Driven Development) approach involving a Visual Modelling Language[8][9] or a DSL (Domain Specific Language)[10]. As many of the early design decisions are architectural in nature it is vital that the appropriate structures and components are in place at the start of a project. The architecture of an app is consists of the following components: Visual Layout, Functional Aspects, Background Services, Data Model, Sensors, Session Management, Security Model, API and Data Persistence. These components form the internal structure of an app and must satisfy the concerns of an end user. Code needs to be written for each of these components. In addition to this, developers need to take into consideration the concepts of Data Flow, Error Handling, Caching strategies, Concurrency, UI Guidelines and State Models while following Software Engineering best practices. In short, there are a lot of things for a developer to consider when building a mobile app -- especially when starting a new project. This PhD thesis looks at how a model based automation tool can assist the developer by generating scaffolding code at the start of a new app. The generated code will provide the structure for the architecture and acts as a guide for the rest of the implementation of the app. There are two stages to this study. First we will conduct a voluntary developer survey, to elicit the tools and processes used in the early stages of development. This will help steer our research direction by ensuring that we develop a solution that fits in with the current tools and approaches. It will also help us identify the major short comings of these tools from a developers perspective. The second stage of our study will involve an end user evaluation of a code generation tool that we created. Developers and students will both take part in this stage. Below is an outline of how the evaluation will take place.  

- First participants will be asked to complete a questionnaire about their experience developing mobile apps.  
- Then there will be a session to teach participants to use our tool by walking through sample code and completing small tasks.  
- After the learning tasks are complete participants will fill out a questionnaire to gauge their confidence with our tool.  
- Next participants will complete two timed development tasks one using the Android SDK (internet resources and 3rd party libraries are allowed) and one with using our tool (samples from the learning session will be available).  
- There will also be a task where you will be asked to modify the code that is generated by our tool.  
- Finally there will be the last questionnaire to fill out to provide your feedback and thoughts on using our tool.  

The experiment has been designed to take approximately 2 hours. The results from this study will highlight the differences between current tools and our new tool. These findings will also clearly define the pros and cons of using our tool. In addition to this we plan to show the usefulness of our tool and the types of apps that can benefit most from using our tool as seen from a developers perspective. We hope to demonstrate that using our solution developers can build apps that have a solid, reusable architecture in less time. |
**A2 WHAT - BRIEF DESCRIPTION OF PROJECT**

**In plain English**

The goal of this project is to better understand the practices of a professional mobile app developer and to evaluate a new Android app code generation tool. Existing literature contains many examples of using model driven development tools to build mobile apps. Little is understood about whether these tools adequately satisfy the needs of a developer in the early stages of building a new app. Additionally, it is unclear what necessary constructs are required for modelling sophisticated data-driven apps.

This study is composed of two parts: 1) a survey and 2) a target end user evaluation. The survey aims to uncover the current practices used by developers when they start a new app. From the results of this survey we hope to determine the current processes and tools used by developers to help us better understand what is needed in new code generation tools. The second phase of this project involves both students and professional Android app developers evaluating our tool that aims to improve developer productivity.

Participants will be asked to complete a number of small programming tasks and to fill out a questionnaire. This is so that we can gather both quantitative and qualitative data. Quantitative data will be collected through timing the programming tasks. Qualitative data will be gathered from analysis of the answers in the questionnaire. We hope to show that our approach is easy to learn and contains the necessary constructs for building sophisticated mobile apps. We also want to try and quantify any productivity increases that might result from using our tool.

**A3 HOW - PROCEDURES**

Please detail clearly and sufficiently the proposed research/statistical method(s), procedures and instruments to be used in the project, including all screening and research 'procedures' to which the participants will be subjected, and asterisk those which may have adverse consequences. Please include as appendices all screening instruments, questionnaires, interview protocols etc (at least in draft form if not finalised).

<table>
<thead>
<tr>
<th>Phase 1: Mobile Developer Practices Survey</th>
</tr>
</thead>
<tbody>
<tr>
<td>The first phase of this study will involve a qualitative online survey. We will distribute the survey, Mobile Developer Practices shown in Appendix H, via email and social media. Pilot studies have been conducted within the research group, with colleagues who have a software engineering background, to refine the survey questions.</td>
</tr>
<tr>
<td>Questions in the survey have been grouped into the following categories:</td>
</tr>
<tr>
<td>Background Information: These questions are designed to elicit a developer's experience. There is no personally identifiable information obtained from these questions.</td>
</tr>
<tr>
<td>Prototyping: These questions are designed to determine whether the participant uses prototypes and whether they think they are useful.</td>
</tr>
<tr>
<td>3rd Party Tools: Questions in this section have been selected to better understand the usage of libraries and tools created by third parties as opposed to the platform owners.</td>
</tr>
<tr>
<td>Getting Started: These questions elicit how the participants start development of a new mobile app.</td>
</tr>
<tr>
<td>Tooling: This category examines participants' views on the quality of the current tooling available.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Selection of participants</th>
</tr>
</thead>
<tbody>
<tr>
<td>Participants will be recruited by inviting people from online user groups and mailing lists to take part in the survey. Emails will also be sent out to contacts that are known by the student researcher, principal supervisor or associate supervisor. A sample invite email can be found in section C2.</td>
</tr>
<tr>
<td>Participation in the study is voluntary and participants are free to stop participating at any time they wish. Participants can also request that their collected data be deleted prior to publication. All participants will be given a consent letter and a consent form (See Appendix B) to sign before proceeding with answering the survey. All data from Phase 1 will be stored electronically in an encrypted format, on a password protected computer and server in Swinburne University of Technology, Australia, for 5 years of study. Paper-based collected data will be stored in a locked cabinet for 5 years in the FSET office and only accessible to the researchers. Electronic documents will be deleted from the server after 5 years of study. Paper-based records will also be securely destroyed at the end of this period.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Only qualitative data will be collected for this phase. Qualitative analyses will be descriptive and will examine the distribution of all answers instead of only calculating medians.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Phase 2: Tool Evaluation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Participants will be recruited from respondents who participated in the online survey and from contacts in industry. Notices will also be circulated amongst the local Android Meetup group, Melbourne Mobile and the</td>
</tr>
</tbody>
</table>
Mobile Monday Melbourne group. Student participants will be recruited on a voluntary basis with posters placed around the FSET faculty buildings. Students must have completed a basic level mobile app development class and be able to build simple Android apps. The tool evaluation involving students will be conducted at Swinburne University of Technology. Professional developers will be recruited from a set of companies, which the research team members contact. Please refer to Appendix F for Onsite protocols.

The evaluation phase will be conducted twice, once with student participants and separately, with professional developers. Participants will be split into two groups, each group completing their tasks in separate sessions. At the start of each session the researcher will greet the participants and inform them of what will take place during the session. For the sake of consistency, the introduction will follow a documented script. All participants will be told that they are free to stop participating in the experiment at any time and can leave if they so wish. Participants will be given the opportunity to ask questions and then will be asked to read and sign a consent form. The researcher will then proceed with the training session.

The training session will involve a demonstration of a concept using the tool. The participants will then be asked to complete a simple task using that new concept. There are six learning tasks for the participants to complete. On completion of the training session the participants will be asked if they have any questions about using the DSL and if there is anything they are unsure about. At this point participants will be asked to fill out a simple questionnaire (Appendix E) outlining how confident they feel with using the DSL for simple tasks.

The task set for the session will then be introduced to the participants. Each task set involves four tasks: two tasks using the tool and two tasks without. The tasks involving using our tool will be completed last as using our tool first would assist the developers in completing the other tasks. Each task set can be found in Appendix C & D. Participants will be asked if they have any questions about the assigned tasks before commencing development. The participants will then begin the tasks indicating when they have completed each task. The researcher will then record the time it took each participant to complete each task. The tasks are designed to take approximately 1.5 hours total.

Finally, the participants will be asked to fill out a questionnaire gauging their sentiment towards using our tool compared to not using our tool. This will allow us to capture qualitative feedback on our approach. The final questionnaire is divided into four sections: Overall Reactions - used to gauge how developers feel towards using the tool; Language Constructs - questions here are designed to expose any missing constructs in our language; Language Comparison - to uncover how developers view our language from the perspective of traditional development approaches; Subject Comments - to capture any additional feedback from developers that they may wish to give us on our research and approach.

**Student Evaluation**

Evaluations involving students will be conducted in a Swinburne classroom. Participants will be asked to provide their own laptops already setup for Android development. The Student Researcher will arrange to be at the classroom one hour before the experiment is to start to assist students in setting up their laptops. Students will be recruited on campus, via word of mouth and through advertising in the appropriate classes. Participants will first contact the Student Researcher to ensure that their background makes them eligible to take part in the study. All participants must have completed a mobile app development unit, have access to the internet, have their own Android device and a USB cable. The study will include approximately 15 students. Once 15 eligible students have agreed to participate, two days will be set aside to run the two sessions, each session on a different day. Students will be randomly assigned to a session.

**Developer Evaluation**

The evaluation involving developers will take place at the company office where they work. The Employer Consent form is shown in Appendix G. For participants who are unable or unwilling to use their workplace offices, the evaluation will take place in the usability lab at Swinburne University of Technology. Participant recruitment will follow the same process as the one outlined above in Phase 1. For the developer evaluation, we aim to have approximately 20 participants take part.

The results from this study will uncover how using our DSL affects productivity in standard Android development. Through the qualitative questions in the survey, we will also uncover developers' attitudes towards using a productivity tool and what is required in order for it to become an additional tool in their toolbox. The study will also reveal how well we have captured the concepts of a data driven mobile app and will expose any missing constructs.

All data from Phase two will be stored electronically in an encrypted format, on a password protected computer and server, in Swinburne University of Technology, Australia, for five years of study. Paper-based collected data will be stored in a locked cabinet for five years in the FSET office and will be only accessible...
to the researchers. Electronic documents will be deleted from the server after five years of study. Paper-based documents will be securely destroyed after the same time period.

Analysis

Qualitative analyses of free text answers will be conducted using thematic analysis.

Copies of the surveys for the developer to fill out, pre and post evaluation, can be found in Appendix E.

If you feel that it is necessary to include further material, please append.

### A4 Describe Any Risk That May Arise to the Participant / Donor?

**Risk to participants (and to researchers) can be real but does not need to be physical. Risk includes such as self esteem, regret, embarrassment, civil or criminal liability, disease, physical harm, loss of employment or professional standing, etc. Please consider such possibilities carefully.**

Some research activities may put the participant at risk through what is being done or simply through their participation. Please describe the risk you perceive and the protective measures to be taken.

We do not collect or store any identifying information and seek only voluntary participation so do not foresee any risk to the participants. The questionnaire is designed so that there is no place for the participant to include identifying information. See Appendix E and H for the surveys.

### A5 Describe Any Risk That May Arise to the Researcher / Administrator?

Some research activities may put the researcher at risk through what is being done or simply through their participation.

Please describe the risk you perceive and the protective measures to be taken.

**Minimal risk, no more than everyday life**

### A6 What Benefits Are Anticipated From the Project

Ethical principles would require that benefits flowed from the activities - but please avoid grandiose claims.

(a) **To the Participant (what and how so)**

There is no direct benefit to the participants other than being aware of and having access to this research underway at Swinburne University.

(b) **More generally (to society, profession, knowledge, understanding, etc, and how so.)**

From this research project, we hope to create a better understanding in the research community about the needs of mobile app developers. Hopefully this will lead to the creation of more tools and approaches that will improve mobile app developer productivity.

### A7 Potential Problems

From time to time in the course of a research project important information, such as an individual found to be at risk, or entirely unforeseen events may come to pass. What procedures are in place to handle unexpected or particularly significant personal or other information that may come to light through the project, eg, unknown medical/psychiatric condition, a particularly distressed participant, civil or criminal liability, etc.

Participants will be informed of their right to withdraw at any time.

### A8 Professional/Ethical Ability & Training (Researchers/Students/Assistants)

NS 1.15 Research must be conducted or supervised only by persons or teams with experience, qualifications and competence appropriate to the research … using (appropriate) facilities … (and with appropriate skills and resources for dealing with any contingencies…

(a) Sufficiently detail what investigators/assistants will do in this project and their expertise/competence to do so.

**Chief Investigator: Dr. Rajesh Vasa**

Dr. Rajesh Vasa has been involved in the IT industry for over 15 years. His major research interests lie in solution architecture, mobile technology and analysis of large data sets. He has worked in a range of roles including Software Engineer, Data Scientist, CTO, as well as running an engineering team as an IT Director at Thomson-Reuters. He is a trusted advisor for various technology companies in Australia, Japan and India. Rajesh is also involved in consultant activities and has received several awards and grants. He has extensive experience with both qualitative and quantitative software engineering research methods.

**Associate Investigator: Professor John Grundy**

John has approximately 20 years experience in the IT industry as a Programmer/Analyst and consultant. His research interests include software tools and techniques, software architecture, model-driven software engineering, visual languages, software security engineering, service-based and component-based systems and user interfaces. John's current position at Swinburne University is Dean of the School of Software and...
Electrical Engineering and the Director of the Swinburne University Centre for Computing and Engineering Software Systems (SUCCESS) research group. John has a lot of experience with ethics approvals with over 30 approvals in the past or in progress. Predominantly, they have been on surveys and have been approved by the University of Auckland Human Subject Ethics Committee and Swinburne University Human Research Ethics Committee. Recent examples include:

- 2013/010 - Evaluation of a model visualisation and transformation tool (CONVerT)
- 2013-003 – A preliminary expert evaluation of ABC4GSD
- 2012/243 - Personality traits of software testers

Student research: Scott Barnett

Dr. Rajesh Vasa and Prof. John Grundy will preside over the research work of Scott Barnett. Scott's qualifications include BSc (Hons) Computer Science and BEn Robotics and Mechatronics. The practical research component will be carried out by Scott. Scott has participated in an Ethics training course at Swinburne University of Technology.

(b) Sufficiently detail any further training/qualifications required for investigators/assistants to carry out the project.

No further investigator training is required.

A9 FUTURE USE OF DATA

Will any of these data be used by yourself, your students or others for any purpose other than for this project as described in the protocol? If so please describe.

The data obtained will only be used for this research project. The data will be retained for five years after the completion of the student researcher's PhD thesis. The thesis is expected to be completed by the end of 2015. The archive of the data will be in the care of the principal investigator, Dr. Rajesh Vasa, after the completion of the student researcher’s PhD thesis.

The electronic archive will be encrypted and stored on a secure FSET sever and then securely destroyed after five years of the completion of the project. Paper records will also be securely destroyed at the end of the five year time period.

A10 EXTERNAL INVOLVEMENT

Is a body external to Swinburne involved in initiation or support of the project?

☐ Yes Name of body/organisation.

If an external body is associated with the project you must provide the HREC with detail of the arrangements, including details of any funding or other resources being provided. A copy of relevant pages from the contractual arrangements should be attached.

☐ No

A11 EXTERNAL APPROVALS

Projects involving other organisations or entities may require approval from other institutions or their ethics committees, etc. for such things as access to prospective participants, contact lists, data, facilities, etc. A copy of such approvals may be required to be provided to the HREC at the time of application or be made available as soon as possible. In which case, the project may not commence, until such evidence is provided.

Please indicate, as appropriate, if formal clearance/permission has been obtained or sought:

Institutional Yes ☐ Documentation Attached ☑ or to follow ☑

Next of Kin (for special groups) Yes ☐ Documentation Attached ☑ or to follow ☑

(estimate when likely to be obtained)

We are in the process of obtaining permission to advertise for voluntary participation from a number of companies. As these approvals come in we will send them through to the Research Ethics Office.

☐ No (please explain)

A12 RESEARCHER / SPONSOR RELATIONSHIP

Is there any relationship or association between the sponsor and any of the researchers listed in Section A of this form, for example are any of the researchers directors, officers, employees, shareholders or promoters of the sponsor or do they receive any personal benefits from the sponsor under any other contracts or arrangements?

☐ Yes ☐ No

Yes (please explain the relationship(s), including how a vested or a conflict of interest situation does not arise.)

Not applicable
SECTION B: ETHICAL ISSUES OVERVIEW

B ETHICAL ISSUES

[Double-click on YES/NO 'check box' to select box, then enter Default Value as Checked ☑ or leaving as Not Checked ☐]

(a) Non-/Limited Disclosure or Deception: Is any detail in relation to research purposes, methods or questions being withheld from participants? Or will deception of any kind be involved? Or any covert/undeclared observation? (Refer National Statement Chap 2.3)

(b) Does the data collection process involve access to confidential personal data (including access to data provided for a purpose other than this particular research project) without the prior consent of subjects?

(c) Will participants have pictures taken of them, e.g., photographs, video recordings?
   If "YES", please explain how you intend to retain confidentiality and ultimately dispose of the material.

(d) If interviews are to be conducted, will they be recorded by electronic device?
   If "YES", please explain how you intend to retain confidentiality and ultimately dispose of the material.

(e) Will participants be asked to perform any acts or make statements which might compromise them, diminish self esteem or cause them embarrassment or regret (minimal, moderate or significant)?

(f) Might any aspect of your study reasonably be expected to place the participant at risk of criminal or civil liability (not just immediately or directly)?

(g) Might any aspect of your study reasonably be expected to place the participant at risk of damage to their professional/social/cultural/financial standing or employability?

(h) Will the research involve access to data banks subject to privacy legislation?*  
   (NOTE: Annual reporting to Government may be required on this item. For info: please contact the Research Ethics Officer.)

(i) Will participants come into contact with any equipment which uses an electrical supply in any form e.g., audiometer, biofeedback, electrical stimulation, magnetic stimulation, etc.? If "YES", please outline below what safety precautions will be followed.

(j) Will any treatment be used with potentially unpleasant or harmful side effects?

(k) Does the research involve any stimuli, tasks, investigations or procedures which may be experienced by participants as stressful, noxious, aversive or unpleasant during or after the research procedures?

(l) Will the research involve the use of placebo control conditions or the withholding/substitution of treatment, programs or services (health, educational, commercial, other)?

(m) Will any samples of body fluid or body tissue be required specifically for the research which would not be required in the case of ordinary treatment?

(n) Will participants be fingerprinted or DNA "fingerprinted"?

(o) Are there in your opinion any other ethical issues involved in the research?

NOTE: If the answer to any of the above questions is "yes", please explain and justify below in sufficient clear detail. (The box below will expand to fit your response.)

Not applicable

Attach further documents if appropriate
SECTION C: PARTICIPANT DETAILS

C1 PARTICIPANT DETAILS
The composition of the participant group may, in some circumstances, distort and invalidate an outcome, and risks may arise through the composition of the participant group.

How many individual participants will be involved? (Number/number ranges for which approval is sought)

Males: N/A  
Females: N/A  
Total participants: 100 (max for each phase)

Over what range of ages?
From (youngest): 18  
To (Oldest): No limit

If there is a gender or age imbalance in the number of participants please explain why.

C2 RECRUITMENT
How will participants be recruited/selected?
Please outline the process in sufficient detail how this is to occur.
Note: Where participants are obtained from or through schools, hospitals, prisons or other institutions, appropriate institutional or other authority will probably be needed. If soliciting for participants by advertisement or poster please attach proposed copies or text.
(See also Project Information Consent Statements and Signed Consent Forms info at the end of this application form.)

Student Recruitment:
Students will be recruited through the use of posters placed around Swinburne University and will only take part in the Tool Evaluation phase of the study. Students who volunteer to participate will be asked the following questions to establish if they are suitable for this study:

Have you completed a mobile app development unit?
Have you developed any Android apps before?

Students must answer yes to both questions to be eligible.

Developer Recruitment:
Developers will be recruited through presenting at user and interest groups in Melbourne with prior approval of the moderators of these interest groups: http://melbourne.mobilemonday.com.au/, http://www.meetup.com/android-australia-user-group-melbourne, http://www.meetup.com/MelbourneMobile/ and http://www.meetup.com/Mobile-Monday-Melbourne/. At these meetups we will also ask for companies who would be willing for us to advertise on their internal mailing lists for developers that would like to participate. In addition to this we will also post to online message boards with moderator approval. After the companies have agreed to take part in the survey we will ask them to sign the organisational approval form (Appendix G). Once participants have volunteered to participate they will read the Consent Information Statement (Appendix B) and sign the Consent Form (Appendix J).

C3 PRE-EXISTING CONDITIONS
In some situations an underlying medical or other significant condition of a participant may result in an otherwise relatively innocuous situation causing excessive stress and exacerbate the condition. Researchers must, therefore, be alert to such situations and be able to address the resulting issues.

Do participants have any medical or other significant condition of which you are aware, eg. diabetes, asthma, depression, epilepsy? What steps are in place to handle any resulting problems (you may need to correlate with A3, A4 and A7 of this form)?

Not applicable

C4 DISCLOSURE AND INFORMED CONSENT
How will participants be informed about the project in order to give valid consent:

☑ Consent Information Statement(s)/Letter(s) and Signed Consent Form(s) will be used.
   A copy must be attached to your application. A guide to consent instruments is given at the end of this form.
☐ Consent Information Statement(s)/Letter(s) and consent implied by return of anonymous questionnaire
☐ Verbal advice (Please explain how and why)
☐ Other (Please explain how and why)
Participants will be emailed an invitation. Refer to C2. The Consent Information Statement and Signed Consent Form is in Appendix B.

Copies of appropriate consent instruments must be attached to your application. Please consult the Guide to Human Research Informed Consent Instruments in carefully preparing informed consent instruments.

C5 COMPENSATION
Consent to participate must be freely given and not induced through the level of reward, perceived reward, or power relationships
Provide details of any financial or other reward or inducement being offered to subjects for participation. Indicate the source of the funds.

Not applicable

C6 RELATIONSHIP TO INVESTIGATOR(S)
Free consent may be difficult to ensure if the participant is dependent upon the investigator for employment, assessments etc.
Some relationships cause special ethical issues to arise
Are participants linked with the investigator through some particular relationship - eg. employees ultimately responsible to or superiors of the investigator, students of investigator, family members, friends etc.

The evaluation will run outside of the regular semester to ensure that the Supervisors are not lecturing any of the student participants.

Developer participants are peers to the investigators. The participants in no way depend upon any of the investigators for professional advancement. Likewise the participants are not responsible for the investigators.

C7 INVOLVEMENT OF SPECIAL GROUPS
Particular issues of consent may arise where special groups of participants are to be involved. There may be, for example, a need to obtain informed consent from persons other than the direct participant. Examples of such special groups include special cultural groups - eg. indigenous Australians; children and young persons (Guidelines section 4.2); groups with special circumstances - eg. persons with an intellectual or mental impairment (Guidelines s. 5)

Please identify and describe the nature of the groups and procedures used to obtain permission.

Note. Persons proposing research projects involving Indigenous Australians should consult with the relevant University manager of indigenous programs prior to finalising definition of the project.

Not applicable

C8 PRIVACY
The University is subject to the Victorian Information Privacy and Health Records Acts as well as the Commonwealth Privacy Act and, in particular, the Information/Health/National Privacy principles (IPPs/HPPs/NPPs) set out therein and is required to report annually on projects which relate to or utilise particular records.

Does the research involve access to data which was collected by an organisation for its own purposes (ie. not specifically collected for this project) such as student records, other data banks, human pathology or diagnostic specimens provided by an institution/s?

If yes, please indicate source/s.

Not applicable

C9 LOCATION OF STUDY
Please indicate where the research will be carried out. If the research will not be on University premises permission of owner / occupier may be required. If so, please indicate what authority or permission may be required and how will be obtained. NB: Where required, please attach to this application evidence of authority obtained or provide the Secretary, HREC as soon as practicable.

Phase 1: Current practices survey
As mentioned in C2, this questionnaire will be online and can be completed anywhere.

Phase 2: Tool Evaluation
The evaluation with the students will be held at Swinburne University in a spare classroom.
The evaluation with developers will need to be held in their respective offices. The Employer Consent form is shown in Appendix G. For participants unable or unwilling to use their offices, the evaluation will take place in the usability lab at Swinburne University of Technology.
SECTION D: DATA & PUBLICATION ARRANGEMENTS (Nb Section D Revised Aug 2007)

PLEASE CONSIDER CAREFULLY YOUR RESPONSES TO THIS SECTION. YOU NEED TO BE CLEAR AS TO WHAT IS OCCURRING WITH RESPECT TO DATA COLLECTION, RETENTION AND DISPOSAL.

(In your responses, you should demonstrate familiarity with National Statement requirements for confidentiality, relevant Privacy Principles and Swinburne’s Policy on the Conduct of Research, eg, Sect 4, see URL: http://www.swinburne.edu.au/corporate/registrar/pd/docs/PolicyontheConductofResearch.pdf).

D1 DATA COLLECTION/RECORDING (Nb Section D1 Revised Aug 2007)

Please note that, with any information or data collected/retained, if any individual can reasonably be identified, the information can be deemed “personal information” or “health information” under National/Health/Information Privacy Principles (NPPs/HPPs/IPPs).

(a) How or in what form will data be collected/recorded?

(eg, notes; verbatim, audio and/or video recordings; transcriptions of recordings; recorded or signed consents; etc)

Phase 1: Current practices survey
- Online survey – web artefacts

Phase 2: Tool Evaluation
- Code for evaluation tasks
- Time it takes the participant to complete each task
- Answers to the questionnaires to be asked before the learning task, after the learning task and on completion of the whole experiment.

(b) As regards any individual, in relation to any data collection or retention, you need to acknowledge either or both of the following:

☐ An Individual can be identified OR is Potentially Identifiable / Re-Identifiable

(An individual can be identified at some point or by the very nature of the data collected/retained: at time of an interview, by signed consent form, identified or labelled voice or image recording, pen-and-paper questionnaire, on-line survey instruments, etc.

Whilst data may not have (explicit) identifiers, an individual’s identity can still be reasonably worked out.

Or data may have (explicit) identifiers removed and replaced by codes that permit matching of an individual with the data collected/retained, in which case it is possible to identify or re-identify the person to whom the data relates.)

☐ An Individual is Non- or Un-identifiable

(Data collected/retained anonymously and with no reasonable possibility of being identified.)

Your acknowledgement may require further explanation or clarification; if so, please include in the following box.

The online surveys only ask participants about their working background and not for any personally identifying information.

If any participant includes personal-identifying information e.g., company name or their own, name, this will be redacted before data analysis.

D2 DATA SECURITY (Nb Section D2 Revised Aug 2007)

Please note that “data must be held for sufficient time to allow reference. For data that is published this may be for as long as interest and discussion persists following publication. It is recommended that the minimum period for retention is at least 5 years from the date of publication but for specific types of research, such as clinical research, 15 years (or more) may be more appropriate.” (Sect 4.3 of Swinburne’s Policy on the Conduct of Research)

Please indicate how data (all types of data, including, eg, signed consent forms) will be securely retained (eg, electronic form in password-protected disk drive, locked filing cabinet, etc) and where? With more than one type of data, will the types be separately stored?

In your explanation, you will need to make clear how due confidentiality and/or anonymity will be maintained.

(a) During the study

All types of data will be stored electronically in an encrypted format, on a password protected computer and server in Swinburne University of Technology, Australia, for five years of study. Paper-based collected data will be stored in a locked cabinet for five years in the FSET office and only accessible to the researchers.

After this time, paper based records will be securely destroyed.

(b) Following completion of study

Documents including answers to the questionnaires and source code for the evaluation tasks will be deleted from the server after five years of study. Paper based records will be securely destroyed after five years of study.
D3 PUBLICATION/OUTPUT (Nb Section D3 Revised Aug 2007)

Please explain in sufficient detail:

(a) What, if any, publication (conference, news media, academic journal, other journal, etc) is envisaged following on or in relation to this project, both in terms of data proper and/or analysis of data?

(b) Will participants be informed about any envisaged research publication/outcome? (This information is normally to be included in the information given prior to obtaining informed consent.)

(c) Would any participants be able to be identified through the publication of data proper or research findings? If so, explain why this is necessary.

(a) The results will be used in journal articles, conference papers and in a PhD thesis.

(b) Yes, see Appendix B.

(c) No.

D4 INDIGENOUS ISSUES

Storage arrangements for data relating to research into Indigenous matters must be determined in compliance with the Policy on the Conduct of Research after consultation with the communities involved.

What consultation has taken place and what arrangements have been made.

Not applicable

D5 OTHER ISSUES (Nb Section D5 Revised Aug 2007)

Are there any other issue relating to data collection, retention, use or disclosure which the ethics committee should be made aware of and, if so, please explain how you are to deal with this.

(Eg. Research outcomes unduly impacting on any individual or group not directly participating, etc.)

Not applicable
**SECTION E: SUBSTANCES & CLINICAL ISSUES**

No matters in this section are applicable to the study or

**E1 ADMINISTRATION OF SUBSTANCES/AGENTS**

<table>
<thead>
<tr>
<th>Name of substance(s)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Dosage per administration</td>
<td></td>
</tr>
<tr>
<td>Frequency of administration</td>
<td></td>
</tr>
<tr>
<td>Total amounts to be administered</td>
<td></td>
</tr>
</tbody>
</table>

Anticipated effects:

NOTE: If the research involves administration of foreign substances or invasive procedures, please attach a statement accepting responsibility for those procedures by a medical or paramedical practitioner with Indemnity insurance.

STATEMENT ATTACHED

**E2 BODY FLUIDS OR TISSUE**

What fluids or tissue? How will be samples be obtained?

Frequency and volume

How are samples to be stored?

How will samples be disposed of?

Who will take the samples?

What are their qualifications for doing so?

Do participants carry, as far as you know, the Hepatitis B or HIV virus? If so how will the risks be handled

Do participants carry, as far as you know, any other contagious diseases or viruses? If so how will the risks be handled

SECTION F  Declarations for Signature

1. With respect to this project, I / We, the undersigned investigator(s)/Assistant(s) agree:
   - To undertake human research activity or handle data confidentially in accordance with Swinburne requirements, including any standard or special ethics clearance conditions, under the proper direction of the responsible Swinburne manager and/or principal Swinburne (or other) researcher/ supervisor.

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIGNATURE</th>
<th>DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mr Scott Barnett</td>
<td><img src="signature" alt="" /></td>
<td>21 November 2014</td>
</tr>
<tr>
<td>Dr Rajesh Vasa</td>
<td><img src="signature" alt="" /></td>
<td>21 Nov 2014</td>
</tr>
<tr>
<td>Professor John Grundy</td>
<td><img src="signature" alt="" /></td>
<td>21st November 2014</td>
</tr>
</tbody>
</table>

All listed applicants must sign. The Chief Investigator/Supervisor is also responsible for personnel subsequently joining the project. Expand this table or duplicate this page as required. NB This information is subject to Swinburne's or external audit.

**Please note that: PROJECTS MUST NOT COMMENCE WITHOUT PRIOR WRITTEN APPROVAL from the Human Research Ethics Committee (SUHREC) or its appropriate Subcommittee (SHESC)**

2. Declaration of Compliance by Chief Investigator(s)/Student Supervisor(s).

I declare that the above project has been developed and will be conducted in accordance with relevant Swinburne standards, policies and codes of practice, including any standard or special conditions for on-going ethics clearance. I further declare that all listed and subsequently appointed researchers or assistants involved in this project will be made aware of the conditions of ethics approval as communicated to me, including approved documentation and procedures.

Signature & Date: ![](signature)
Name of Signatory & Position: Dr Raj Vasa, Senior Lecturer; Mr Scott Barnett, PhD

(Optional) Form checked by a Research & Ethics Advisor (REA)? Yes ☐ No ☐ REA Initials & Date: ...

3. Endorsement of Head of Academic Unit (or Delegate) or Above.

I declare that this project has been developed and will be conducted in accordance with relevant Swinburne standards, policies and codes of practice, and has research merit, secured resourcing and appropriate leadership/supervision.

Signature & Date: 24 NOV 2014
Name of Signatory & Position: Professor John Wilson, Executive Dean, FSET

(Please note: This endorsement must be given by an authorised official who is not also a chief or co-investigator of the project and who is not also the supervisor of a student investigator with an interest in the project.)

Human Research Ethics Committee
Appendix A. References


Appendix B. Consent Information Statement

Research Supervisor

Dr Rajesh Vasa
Deputy Head, R&D Software Group
Faculty of Science, Engineering and Technology (FSET)

Research Supervisor

Professor John Grundy
Dean, School of Software and Electrical Engineering

Researcher

Scott Barnett
PhD Candidate

Dear research participant,

You are invited to participate in this research study as part of the PhD research by Scott Barnett, supervised by Dr, Rajesh Vasa in the Faculty of Science, Engineering and Technology (FSET) of Swinburne University of Technology.

The Research Project

The purpose of this study is to ensure that our research into improving developer productivity is developer focused. Our tool is intended to speed up development by providing developers with a high level language that they can use to describe the features of an app. Once the description is complete the tool will generate an Android project with code that is ready to run on a device. The generated code is intended to be used as the scaffolding for a new project and will require a developer to complete it. This study will be broken up into two major parts: a questionnaire that will elicit current tools and practices used by professional app developers and a user evaluation with both students and developers.

Phase 1: Current practices survey: Procedure

This part of the study will involve filling out the Current Practices Survey. Should you choose to participate in this research, you will be asked to fill out a simple online survey. The survey will have a number of questions about the processes and tools you use when starting a new mobile app project. The survey should take no longer than 15 minutes to complete.

Phase 2: Tool Evaluation: Procedure

If you agree to participate in this research, the tool evaluation will take place with you at a time and location of your choice. The tool evaluation will take place in the following manner.

- First you will be asked to complete a questionnaire about your experience developing mobile apps.
- We will then have a session to teach you to use RAPPT by walking through sample code and completing small tasks.
- After the learning tasks are complete there is another short questionnaire to gauge your confidence with RAPPT.
- Next there will be two timed development tasks one using the Android SDK (internet resources and 3rd party libraries are allowed) and one with using RAPPT (samples from the learning session will be available).
• There will also be a task where you will be asked to modify the code that is generated by RAPPT.
• Finally there will be the last questionnaire to fill out to provide your feedback and thoughts on using RAPPT.
The experiment has been designed to take approximately 2 hours.

Anonymity
No personal information that can be used to identify the participant will be gathered.

Privacy & Confidentiality
The survey results will only be used for academic research purposes. Individual responses will be kept confidential and will not be disclosed. All types of data will be stored electronically in an encrypted format, on a password protected computer and server in Swinburne University of Technology, Australia for 5 years of study. Paper-based collected data will be stored in a locked cabinet for 5 years in the FSET office and only accessible to the researchers. Paper documents will be deleted from the server after 5 years of study. The findings will be published in a publicly available PhD thesis and are likely to be published in conference and/or journal papers at a later date. Data will be presented in collective form and no individuals will be able to be identified in these publications.

Voluntary Participation
Your participation in this research is voluntary. You may discontinue participation at any time during the research activity and we will delete data collected from you (until publication, at which point deletion is no longer possible).

Research Output
The findings from this study will be published in a PhD thesis that will be publically available. Results may also be published in peer-reviewed academic conference and journals as outlined above.

Research Ethics
The project has been approved by the Swinburne University Human Research Ethics Committee.

Further information about the project – who to contact:
If there are any questions you have about the research project you are welcome to contact the researcher, Scott Barnett, or the principal investigator, Dr. Rajesh Vasa,

Dr Rajesh Vasa
Email: rvasa@swin.edu.au

Scott Barnett
Email: sbarnett@swin.edu.au

Concerns/complaints about the project – who to contact:
This project has been approved by or on behalf of Swinburne’s Human Research Ethics Committee (SUHREC) in line with the National Statement on Ethical Conduct in Human Research. If you have any concerns or complaints about the conduct of this project, you can contact:

Research Ethics Officer, Swinburne Research (H68),
Swinburne University of Technology, P O Box 218, HAWTHORN VIC 3122.
Tel (03) 9214 5218 or +61 3 9214 5218 or resethics@swin.edu.au
Appendix C. Evaluation tasks

Task Set 1

Task 1. Mobile Developer Experience Questionnaire
Please fill out the following questionnaire: http://opinio.online.swin.edu.au/s?s=16078

Task 2. Develop a YAWA (Yet Another Weather App) using the Android SDK

For this task you are required to develop a simple single screen weather app, YAWA. Screenshots for YAWA are shown above. YAWA accepts user input, the name of a city, makes a request to a weather API and displays the temperature for that city. For the purpose of this experiment you do not need to validate the data but you must display an alert dialog to the user if there is no Internet connection. The weather API and the response format are shown on the next page. Stack Overflow and any internet resources are allowed. Please do not copy code from previous projects or use previous projects for inspiration.

Before you start development please answer the following questions: How long (minutes) do you estimate it will take you to complete this task? Would you use any third party libraries for this task?

Information:

API: http://api.openweathermap.org/data/2.5
Endpoint: /weather?q={city}

Note: The temperature returned is in Kelvin and does not need to be converted to Celsius.

Sample response:
Task 3. Build YAWA using RAPPT

How long do you think it will take to build the same app using the DSL? Using RAPPT complete the task described in Task 1.

Task 4. Extend the generated YAWA

Modify the generated app to convert Kelvin into Fahrenheit and display the result. Formula for conversion is: 
Fahrenheit = (9/5) * (Kelvin – 273) + 32

Task 5. Evaluation Questionnaire

Please answer the final questionnaire here: http://opinio.online.swin.edu.au/s?s=16079 , Thank you for participating!

Task Set 2:

Task 1. Mobile Developer Experience Questionnaire

Please fill out the following questionnaire: http://opinio.online.swin.edu.au/s?s=16078

Task 2. Develop a YACEA (Yet Another Currency Exchange App) using the Android SDK
For this task you will be asked to develop a single screen currency conversion app, YACEA. YACEA accepts user input, from and to currency, makes a request to a currency conversion API and displays the conversion rate. User input should be the three letter currency code and for this experiment no validation is required. The currency conversion API and the response format are shown on the next page. Stack Overflow and any internet resources are allowed. Please don’t copy code from previous projects or use previous projects for inspiration.

Before starting this task please answer the following questions: How long do you estimate it will take for you to complete this task? Would you use any third party libraries for this task?

Information:

API: http://currency-api.appspot.com/api

Endpoint: /{source}/{target}.json?key=cf6ccd6659ce43126c49f77eb5f4cec3285ef3a1

Sample response:

```
{
  "target":"GBP",
  "success":true,
  "rate":0.5678,
  "source":"AUD",
  "amount":0.57,
  "message":"
}
```

Task 3. Build YACEA using RAPPT

How long do you think it will take to build the same app using the DSL? Using RAPPT complete the task described in Task 1.

Task 4. Extend the generated YACEA

Modify the generated app to prompt the user for the amount and display the result.

The new endpoint is: /{source}/{target}.json?key=cf6ccd6659ce43126c49f77eb5f4cec3285ef3a1&amount={amount}
Appendix D. Learning tasks

Introduction to the Domain Specific Language (DSL)

Welcome everyone. Thank you for participating in this study on improving mobile app developer productivity. The purpose of this study is to ensure that our research into improving developer productivity is developer focused. Our tool is intended to speed up development by providing developers with a high level language that they can use to describe the features of an app. Once the description is complete the tool will generate an Android project with code that is ready to run on a device. The generated code is intended to be used as the scaffolding for a new project and will require a developer to complete it.

1. Hello Scott

Snippet:

```java
app {
    landing-page LandingScreen
    android-sdk "/path/to/android/sdk"
}

screen LandingScreen "Welcome" {
    layout mainLayout {
        label labelId "Hello Scott!!!"
    }
}
```

This example shows the smallest possible code to create an app. It is composed of two major blocks, an app block and a screen block. The screen block contains a layout block that holds the elements displayed to the user. Run this code in the tool and browse the source code. Highlight the directory structure and android annotations.

Learning Task: Create an app that resembles the following screenshot. You will need to use an additional element, 'text-input'.
2. Navigation

Snippet:

```
app {
    landing-page LandingScreen
    android-sdk "/path/to/android/sdk"
}

screen LandingScreen "Landing" {
    layout mainLayout {
        button buttonId "To About Screen" {
            to AboutScreen
        }
    }
}

screen AboutScreen "About" {
    layout mainLayout {
        label labelId "Your name here"
    }
}
```

Buttons are also supported in the DSL and the syntax is shown above. The button block contains instructions that will be executed. In this example, the “to” instruction navigates to the “AboutScreen”.

Learning task:

Add an additional screen to the DSL and navigate to it from an “action” shown on the AboutScreen. An action follows the same pattern as a button except with the button keyword.

3. Navigation Patterns

Snippet:
Major navigation patterns are added to the app block. In this case, a tabbar is added to the app using the tabbar keyword. A tabbar block has a single parameter, its ID, and only contains tab values. A navigation drawer is also a supported navigation pattern and follows the same structure as a tabbar except with the “drawer” keyword.

**Learning Task:** Modify the DSL snippet so that a navigation drawer is shown instead.

### 4. Fetching content from an API

This example shows how to make HTTP requests from an API and display the part of the response.

**Snippet:**

```plaintext
app {
    landing-page AboutScreen
    android-sdk "/path/to/android/sdk"
    tabbar tabbarId {
        /*
        tab <tab ID> "<tab string>" to <Screen ID>
        */
        tab AboutScreenTab "About" to AboutScreen
    }
    screen AboutScreen "About" {
        layout mainLayout {
            label labelId "Your name here"
        }
    }
}
```

Learning Task: Modify the example above so that it fetches and displays a random postcode, API provided below. Remove the fields for displaying Australia and add fields for displaying ‘postcode’, ‘country’ and ‘region’. Hint: To access the postcode in the response: result.postcode:string
API: [https://api.postcodes.io/](https://api.postcodes.io/)

Endpoint: GET /random/postcodes

Sample response:

```json
{
  "status":200,
  "result":{
    "postcode":"CF64 4DA",
    "quality":1,
    "eastings":315609,
    "northings":171535,
    "country":"Wales",
    "nhs_ha":null,
    "admin_county":null,
    "admin_district":"Vale of Glamorgan",
    "admin_ward":"Dinas Powys",
    "longitude":-3.21547406997654,
    "latitude":51.4365084093045,
    "parliamentary_constituency":"Vale of Glamorgan",
    "european_electoral_region":"Wales",
    "primary_care_trust":null,
    "region":null,
    "parish":"Dinas Powys",
    "locality":null,
    "soa":null,
    "msoa":null,
    "nuts":null,
    "incode":"4DA",
    "outcode":"CF64",
    "ccg":null
  }
}
```

5. Displaying a list

This example shows how to display content from an API in a list.

Snippet:

```javascript
app {
  landing-page Main
  android-sdk /Users/scottbarnett/android-sdk
}

api RestCountries "http://restcountries.eu/rest/v1" {
  GET all "/" {
    /* Indicate response is a list */
    list
  }
}

screen Main "List Example" {
  layout mainList {
    label msgId "List of countries"
    list listId {
      /* Note: on-load goes inside list rather than in the layout
      on-load in both a layout and a nested list is not supported*/
      on-load {
        call RestCountries.all
      }
      row rowId {
        label name:string
        label capital:string
      }
    }
  }
}
```

Learning Task

Based on the example above create an app that loads the latest changes made to the OpenLibrary and displays them in a list.

API: [http://openlibrary.org](http://openlibrary.org)

Endpoint: /recentchanges.json
6. **Passing data between screens**

The recommendation is to use all of the prior concepts in the final learning task.

This example shows how to pass data between two screens.

Snippet:
Learning Task:

For this task, you will build two screens: one with a list and one drilldown screen that you will navigate to by clicking on a list item. The required API snippet for the DSL has been included for you. It contains an api-key parameter. The api-key parameter is used to specify a URL parameter used for all API requests, in this case the accessKey for the Cambridge Dictionary API.

```javascript
api cambridgeDictionary "https://dictionary.cambridge.org/api/v1" {
  api-key accessKey "27EKK8kJ5vJDFeqzjcrCq6cd3uXEkgdpXJgkeJNCkdcJGTOrpCx01GPVdxfID4R"
  GET all "/dictionaries" {list}
  GET dictionary "/dictionaries/{dictCode}"
}
```

For the list screen display the dictionary name and, in the drilldown screen, display all three fields.

Sample response for /dictionaries:
Thank you for completing the learning tasks. Before we start the actual tool evaluation please answer this short questionnaire: http://opinio.online.swin.edu.au/s?s=16512.

**Appendix E. Survey Questions**

See accompanying file Surveys.pdf

**Appendix F. Onsite Safety Protocol**

**Communicating with potential participants**

1. An introductory email will be sent out to the potential participant to brief them about the research, along with a Consent Information Sheet and Informed Consent Form.

**Scheduling onsite evaluation**

1. As the evaluation is designed to take approximately 2 hours, this requirement will be made clear to the participant prior to commencement.
2. The research supervisors, Dr. Rajesh Vasa and Prof. John Grundy, will be informed of the time and place of each of the evaluations.
3. Where possible, evaluations will be conducted during office hours. There may be situations where participants prefer to complete the tasks outside of office hours. In these cases, the evaluation will take place in the usability laboratory at Swinburne University of Technology. The research supervisors will be informed if this situation arises.

**Evaluation**

The procedure for the evaluation will involve running through the tasks outlined in Appendix C.
Appendix G. Organizational Approval

Swinburne University of Technology

Project Title: Bootstrapping Mobile App Development

Principal Investigator(s): Dr. Rajesh Vasa

1. On behalf of: ………………………(Name of Organisation)………………………………

I hereby authorise the following employee(s) to participate in the project in a representative capacity, the project’s particulars having been satisfactorily explained to me:

<table>
<thead>
<tr>
<th>Name of representative(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
</tr>
<tr>
<td>2.</td>
</tr>
<tr>
<td>3.</td>
</tr>
<tr>
<td>4.</td>
</tr>
<tr>
<td>5.</td>
</tr>
</tbody>
</table>

2. In relation to this project, please circle your response to the following:

- I agree that s/he can be interviewed by the researcher
- I agree that s/he can complete questionnaires about Mobile app development practices
- I would like to check any transcription / citation in respect of my organisation’s involvement for accuracy

3. Please circle your response to the following:

- I give my permission for the organisation to be named in any publication arising from the research.
- I give permission for the experiment to be run at the organization during office hours and run for 2 hours.

4. I acknowledge that the data collected for the Swinburne project will be used for research purposes and not for direct profit; research purposes may include publishable / peer reviewed outcomes.

Name of Person of Authority and Position: ……………………………………………………………

Signature & Date: …………………………………………………
Appendix H. Example Questions
Questions shown in Thesis Appendix G. User Evaluation.

Appendix I. Data Recording Form

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Appendix J. Consent Form
Please see attached Consent Form document.
Participant Consent
1. I consent to participate in the project named above. I have been provided with a copy of the project consent information statement to which this consent form relates and any questions I have asked have been answered to my satisfaction.

2. In relation to this project, please circle your response to the following:

- I agree to make myself available for further information if required **Yes No**
- I agree to complete questionnaires asking me about the tools I use and the practices I follow when developing mobile apps **Yes No**
- I agree to participate in completing a number of small tasks that will involve building small Android apps **Yes No**

3. I acknowledge that:

(a) my participation is voluntary and that I am free to withdraw from the project at any time without explanation;

(b) the Swinburne project is for the purpose of research and not for profit;

(c) any identifiable information about me that is gathered in the course of and as the result of my participating in this project will be (i) collected and retained for the purpose of this project in the manner described above and (ii) accessed and analysed by the researcher(s) for the purpose of conducting this project;

(d) my anonymity is preserved and I will not be identified in publications or otherwise without my express written consent.
By signing this document I agree to participate in this project.

Name of Participant:
……………………………………………………………………………………………………

Signature & Date: …………………………………………………………………………………
Dear Dr. Vasa,

**SHR Project 2014/313 Bootstrapping Mobile App Development**

Approved duration from: 19/02/2015 to 18/02/2017 [adjusted]

I refer to the ethical review of the above project protocol by a Subcommittee (SHESC1) of Swinburne's Human Research Ethics Committee (SUHREC). Your responses to the review, as per the emails sent on 04 and 13 February 2015, were put to the Subcommittee delegate for consideration.

I am pleased to advise that, as submitted to date, the project may proceed in line with standard on-going ethics clearance conditions here outlined:

- All human research activity undertaken under Swinburne auspices must conform to Swinburne and external regulatory standards, including the current *National Statement on Ethical Conduct in Human Research* and with respect to secure data use, retention and disposal.

- The named Swinburne Chief Investigator/Supervisor remains responsible for any personnel appointed to or associated with the project being made aware of ethics clearance conditions, including research and consent procedures or instruments approved. Any change in chief investigator/supervisor requires timely notification and SUHREC endorsement.

- The above project has been approved as submitted for ethical review by or on behalf of SUHREC. Amendments to approved procedures or instruments ordinarily require prior ethical appraisal/clearance. SUHREC must be notified immediately or as soon as possible thereafter of (a) any serious or unexpected adverse effects on participants any redress measures; (b) proposed changes in protocols; and (c) unforeseen events which might affect continued ethical acceptability of the project.

- At a minimum, an annual report on the progress of the project is required as well as at the conclusion (or abandonment) of the project. Information on project monitoring, self-audits and progress reports can be found at: [http://www.research.swinburne.edu.au/ethics/human/monitoring/ReportingChanges/](http://www.research.swinburne.edu.au/ethics/human/monitoring/ReportingChanges/)

- A duly authorised external or internal audit of the project may be undertaken at any time.

Please contact the Research Ethics Office if you have any queries about on-going ethics clearance. The SHR project number should be quoted in communication. Researchers should retain a copy of this email as part of project recordkeeping.

Best wishes for the project.

Yours sincerely,

Astrid Nordmann

SHESC1 Secretary

------------------------------------------------------------------------

Dr. Astrid Nordmann

**Research Ethics Officer**

Swinburne Research (H68)

Swinburne University of Technology

PO Box 218, Hawthorn, VIC 3122
SECTION A: PROJECT DETAILS

SUHREC Project No: 2014/313
Project Title: Bootstrapping Mobile App Development

Chief Investigator/Supervisor: Dr. Rajesh Vasa
Faculty/Div: FSET

Current Approved Duration: 19/02/2015 to 18/02/2017

Student Investigators: Scott Barnett
Co-Investigators: John Grundy

SECTION B: SUMMARY OF PROJECT PROGRESS

Please provide a brief outline of project progress to date, particularly as relevant to this application for ethics approval.

Based on our current work, we have created a Conceptual Domain model that captures the core elements required to be understood by mobile app developers when build a mobile app. Using this model to analyse mobile apps, we have identified a number of reoccurring design patterns specific to mobile app development. We refer to these domain specific design patterns as Domain Tactics as they influence the quality attributes of an app. This proposed addendum seeks to evaluate the Conceptual Domain model and our set of Domain Tactics with professional developers and students.

SECTION C: PROJECT MODIFICATIONS/ADDITIONS

C1 Research project title(s), aims and methods

Are there any changes or additions to any of the following:

☐ Project Title(s), particularly on consent documents? ☑ Yes ☐ No ☐ n/a
☐ Project aims as given on current ethics clearance documentation? ☑ Yes ☐ No ☐ n/a
☐ Project methods/procedures/instruments? ☑ Yes ☐ No ☐ n/a
☐ Project participant cohort(s) or number(s)? ☑ Yes ☐ No ☐ n/a

If YES, please sufficiently explain and justify any modification in relation to what has currently been approved, if need be correlating with information given elsewhere on this form or attached. Please also note information to be given elsewhere on this form.

Additions to Project aims:
Evaluate mined Domain Tactics (Appendix 1) with professional developers and students to validate the current research.

We will hold semi-structured interviews asking open and closed questions (Appendix 2). The following research tasks and interview process is based on [1] and [2].
Research Tasks:
1. Schedule interviews with 5-10 app developers and 5-10 non-app developers i.e. web developers. Interviews will be conducted at the workplace of the participant or at a mutual location such as a coffee shop. The length of the interview will be 1 hour.
2. Prepare interview guides for both app and non-app developers.
3. Conduct interviews with participants with recording (if permission is granted by the interviewee).
4. Summarise content of interview.
5. Transcribe and analyse.

Steps for the interview:
1. Introduction: Clarify use of data (i.e. interviewee is unidentifiable and data will be used in research publications) and ask interviewee for permission to record the interview.
2. Present research: Show a short slide deck (Appendix 3) to the participant highlighting why we are conducting this research, the relevance it has to them and the role they will play.
3. Introduce objective: To validate a set of Domain Tactics to ensure that the information they contain accurately models the domain of mobile app development and to determine whether they are of use to novice mobile app developers.
4. Ask background questions i.e. How many years have you been developing mobile apps?
5. Ask main interview questions about the concepts in the Conceptual Domain model and for the Domain Tactics.
6. Summarise and clarify the major findings with each of the interviewees.
7. Thank interviewees for their time.

Justification for changes:
Adding semi-structured interviews will enable us to assess if our work is relevant to mobile app developers. By interviewing app developers we can gain an insight into the relevance of the information we are capturing and identify any omissions. Conducting interviews with non-app developers will help us validate the structure of the domain specific information and the usefulness to novice developers.

We will use personal contacts to source interviewees.

Contacts at the following companies will be asked if they wish to participate in the research:
JTribe
Kiandra IT
B2cloud
Outware
Itty Bitty Apps
Prompa
Pondr

Each organisation will be asked to sign the Organisation Approval form as described in the original ethics application.

The key decision makers at the above listed companies have verbally agreed to assist in the research having met them at conferences or meetups.

We will email the CEO or MD of the companies listed using the message as outlined in Appendix 4.

Interview Safety protocol is shown in Appendix 5.

C2 Project personnel changes

Are there any changes/additions to or departures re:

<table>
<thead>
<tr>
<th>Item</th>
<th>Yes</th>
<th>No</th>
<th>n/a</th>
</tr>
</thead>
<tbody>
<tr>
<td>Responsible chief investigator/supervisor of the project?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Co-investigators other than students?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Student investigators (ie students researching for their course)?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Other project personnel accessing identifiable data?</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

If YES, please sufficiently outline per named individual using the following form:
- Name, title, gender and institutional affiliation(s)
- Brief summary of project role and qualifications and experience to perform the role including supervision
- What (if any) academic, research or other benefits may accrue to the individual, including whether or to what extent there can be any perceived, potential or actual conflicts of interest and how this can be addressed (information may need to be correlated with C3 and C4)
- With additional researchers, consideration should be given to amending the consent/other instruments to disclose the added interests, more so with regard to identifiable information being collected, used and/or disclosed for research purposes. Student status and outcomes constitute an interest which should ordinarily be disclosed for consent purposes.
C3 Changes to project resourcing and interests

Have there any been any changes/additions or developments re
- Financial or other resourcing for the project? □ Yes  ☒ No  □ n/a
- Researcher interests**? □ Yes  ☒ No  □ n/a

(** eg, staff researcher now using research for a qualification; researcher now has association with key project funder; etc)

If YES, please provide a brief explanation and rationale (where appropriate) including with respect to any reporting to Swinburne or other organisation/authority:

C4 Changes to recruitment and consent arrangements

Are there any changes/additions to? [consider with regard to detail given at C1, C2 and C3 above]
- Participant recruitment / selection? □ Yes  ☒ No  □ n/a
- Consent arrangements / instruments? □ Yes  ☒ No  □ n/a

If YES, please sufficiently outline relative to type of participant or participant group:

C5 Variation/extension to project duration

Is an extension to ethics clearance needed?
- ☒ No  □ YES. If YES, please give the new date for end of data collection/access:
  [NB Maximum duration of ethics clearance is for up to five years, after which a fresh ethics clearance application may be needed using current proforma.]

Extension/Further extension to     dd / mm / yyyy

If YES, please also provide a brief explanation for the request:

C6 Ethical and review issues

With respect to any modification or extension of the project (as per information given above), are there any ethical, governance or other significant issue(s) which now require attention?
- ☒ No  □ YES

If YES, please provide a brief explanation including with respect to any reporting of project progress or outcomes to Swinburne or other authority:

Nb Please note that modification(s) requested might require a higher level of ethical review and/or additional expertise to complete the ethical review.
SECTION D: CHECKLIST FOR THIS FORM

Are any of the following attached (as applicable):

- Additional/revised research instruments (questionnaires, interview schedule, etc) ☒ Yes ☐ No ☐ n/a
- Additional/revised consent instruments (info statement, consent form, etc) ☐ Yes ☒ No ☐ n/a
- Additional/revised recruitment or publicity instruments (flyer, advert, etc) ☐ Yes ☒ No ☐ n/a
- Additional/revised debriefing instrument ☐ Yes ☒ No ☐ n/a
- Additional/revised external permissions/authority ☒ Yes ☐ No ☐ n/a

NB Revised procedures/instruments should preferably be submitted using tracked changes. As well, a 'clean' version using version control should be submitted. Cross reference info supplied on this form with clearly delineated/paginated/headed attachments.

SECTION E: DECLARATION BY CHIEF INVESTIGATOR(s)/STUDENT SUPERVISOR(s)

DECLARATION BY CHIEF INVESTIGATOR(s)/STUDENT SUPERVISOR(s)

I declare that the above report accurately reflects the outcome or progress of the project to date

I acknowledge that an internal Swinburne or external audit may be conducted on the conduct of the project and as regards secure data retention/disposal.

Signature & Date: ___________________________ 16/11/2015

Name of Signatory & Position: RAJESH VASA / ASSOCIATE PROFESSOR

For official use only:

One or more type/level of ethical review needed ☐ SUHREC ☐ SHESC ☐ Delegate ☐ Other

Research Ethics Office Notes:
References


Appendix 1

Problem 1. Must support many devices with different hardware capabilities.

1. Name: Degrade Gracefully

Domain Concerns:

- Hardware Constraints: - Hardware features are not built into all devices
- Sensing - Hardware features often involve new sensors
- Ecosystem - New devices frequently come on the market

Benefits:

- Reliability - functionality is available for all users to use

Liabilities:

- Portability - different degrading strategies may need to be implemented for different platforms
- Maintainability - Changes to the functionality may need to be implemented in multiple ways

Solution:

- Check devices hardware capabilities on startup and setup appropriate functionality
- Implement an alternative means to achieve the same functionality for devices without the necessary hardware.

2. Name: Avoid Legacy Platforms

Domain Concerns:

- Hardware Constraints - Hardware features are not built into all devices
- Sensing - Hardware features often involve new sensors
- Ecosystem - New devices frequently come on the market

Benefits:

- Portability & Maintainability - Ecosystem moves along quickly so reduce the amount of legacy code

Liabilities:

- Reliability - App won’t be available for all users of target platform Solution:

  - Check availability of hardware features with manufacturers and platform vendor.
  - Set a minimum supported platform version covering devices with desired features.

Problem 2. App depends heavily on data stored in the cloud.

3. Name: Ship Data

Domain Concerns:

- Network Connectivity - App may be opened when there is limited or no network connection
Benefits:

- Reliability - App is available even without a network connection

Liabilities:

- Suitability - Content showed in the app may be stale

Solution:

- Provide data with the app to load and show to the user while new data is being fetched.

6. Name: Store data on the device (aggressively cache)

Domain Concerns:

- Network Connectivity - Data for the app depends heavily on a network connection.
- Threading Model - Update data using a background service.

Benefits:

- Reliability - App content is available even when no network connection.
- Usability - The app has partial functionality available when there is no connection.

Liabilities:

- Maintainability - Increased complexity as code needs to be written to synchronize data between client and server.

Solution:

- Aggressively cache all API calls to the network.
- Show old content to the user when calls to the network are being made.
- Enable the user to interact with old content when there is no network connection but provide a visual cue to indicate that there is no connection.
- Update content in the background using background services.
- Only make request to the network where absolutely necessary.

Problem 4. The battery drains quickly.

4. Name: Batch process

Domain Concerns:

- Sensing - App's algorithms operate extensively on values read from sensors.

Benefits:

- Portability - App is easier to port to other mobile platforms as processing done on a remote server.

Liabilities:

- Reliability - Processing of data is done once ??

Solution:

- Send a bunch of sensor readings to a server together for processing.

5. Name: Limit Sensor usage
Domain Concerns:

- Sensing - App relies heavily on the device's sensors.
- Hardware Constraints - Battery capacity is drained through excessive use of sensors.
- Lifecycle - Sensors do not always need to be running in every state of an app.

Benefits:

- Performance Efficiency - Prolonged battery life.

Solution:

- Where possible only access sensors when the user directly benefits from the result.
- Limit the time the sensors are enabled by responsibly enabling and disabling sensor usage in the appropriate states of the app's lifecycle.

---

Problem 6. App frequently freezes or crashes.

7. Name: Respect single UI threaded model.

Domain Concerns:

- Threading Model - Mobile devices have a single thread for the UI thread.
- Lifecycle – Performing the operations in the wrong state of an app can limit performance.

Benefits:

- Usability - Prevents the mobile operating system from shutting down the app.

Solution:

- Avoid running long running operations on the UI thread and run them in a background service when possible.
- Conserve resources when the user is not using the app.
- Check operations are running in the appropriate lifecycle state.

---

Problem 7. Users need to authenticate with the App.

8. Name: Store credentials

Domain Concerns:

- Personal - Mobile devices have a single primary user.
- Lifecycle - Credentials saved and restored in the appropriate states in the Lifecycle.

Benefits:

- Usability - Convenience offered to the user by removing the need to enter text into the screen

Liability:

- Security - Once the phone's lock screen has been passed there is free access to the app.

Solution:

- Store user credentials to avoid unnecessary form filling and skip the login screen when the user next uses app.
Problem 8. App relies on the current user's context i.e. location, time etc.

9. Name: Determine Context

Domain Concerns:
- Sensing - Context of a user depends upon a number of different sensors.

Benefits:
- Functional Suitability - Custom functionality can be provided given the user's context

Liabilities:
- Performance Efficiency - Reading more sensor values increases power consumption.

Solution:
- Take readings from multiple sensors when developing algorithms for determining a user's context.
- Track the history of a user's context to inform the current context.

Problem 9. Multiple platforms need to be supported.

10. Name: Thin clients

Domain Concerns:
- Network Connectivity - Creating thin clients increases the dependency on the server layer.
- Ecosystem - A wider range of users can be reached by supporting multiple platforms.

Benefits:
- Portability - easier to port the application to additional platforms.

Liabilities:
- Availability - A greater dependency on a network connection.

Solution:
- Develop business logic in the server layer and keep the clients as thin as possible.
- Clients should focus on user experience and usability.

11. Name: Hybrid approach

Domain Concerns:
- Sensing - Hybrid approaches may have variable levels of support for different sensors on the different platforms.
- Ecosystem - Integration into each ecosystem the app will run on needs to be considered

Benefits:
- Portability - App will run on each of the supported platforms.

Liabilities:
- Portability - App may have variable levels of support on different platforms.
Maintainability - Adding new functionality can be difficult as the layer between web and native code may need to be extended.

Solutions:

- Select a hybrid based framework and design the app as a web application limiting the use of native features.

Problem 10. Data from the app needs to be shared with other people or integrated with other apps.

12. Name: Integrate with social media

Domain Concerns:

- Social - Content in the app needs to be shared with other people.
- Personal - People are now sharing personal content with others.

Benefits:

- Sociability - More people can engage with content created with your app.

Liabilities:

- Privacy - Content created in the app may appear to a wider audience.

Solution:

- Integrate with social media platforms providing ways to share, comment and like content.
- Ensure that what gets shared to social media platforms is made clear and provide options to disconnect.

13. Name: Import/Export

Domain Concerns:

- Personal - Personal content can be made available to other apps.

Benefits:

- Sociability - Content can be shared with other devices and vendor lock in is avoided.
- Usability - Apps that import content from another source are easier to get up and running.

Liabilities:

- Security (Privacy)- Content once confined to an app is now available on the device
- Maintainability - Complexity increases for handling different import formats.

Solution:

- Provide an option and suitable formats for exporting the content within an app.
- Import content provided by other apps or sources.

Problem 11. User needs to customise the behaviour of the app.

14. Name: Implement a Settings Screen

Domain Concerns:
- Personal - App needs to provide user preferences to customise the app.
- Ecosystem - Mobile device platforms provide customise-able functionality.

Benefits:
- Usability - Users feel more engaged with the app as they can express their preferences

Liabilities:
- Maintainability - user preferences need to be permeated throughout the app.

Solution:
- Add a settings screen to the app to allow users to set their preferences.

Problem 12. Users need to be made aware of fresh content (or new event).

15. Name: Push Notifications

Domain Concerns:
- Ecosystem - Mobile platforms provide a mechanism to push content to mobile devices.
- Network Connectivity - Push notifications depend on a network connection.

Benefits:
- Usability - Personal information can be made available precisely when the user needs it.
- Availability - Content is made available to the user.

Liabilities:
- Usability - Notification spamming can get annoying.

Solution:
- Integrate with a notification push system and ensure there is a screen to display recent notifications in the app.

Problem 13. App needs to run on devices with varying screen sizes.

16. Name: Specialise Asserts

Domain Concerns:
- Ecosystem - There are many different screen sizes and pixel densities in the mobile ecosystem.
- Hardware Constraints - Screen sizes is strictly a hardware constraint.

Benefits:
- Usability - the app is optimally designed for every supported screen of the ecosystem.

Solution:
- Find out from the target platforms the recommended supported screen sizes and pixel densities.
- Ensure that each asset in the app conforms to each platforms recommendations.

17. Name: Alternative layouts
Domain Concerns:

- Ecosystem - There are many different screen sizes and pixel densities in the mobile ecosystem.
- Hardware Constraints - Screen sizes is strictly a hardware constraint.

Benefits:

- Usability - the app is optimally designed for every supported screen of the ecosystem.

Solution:

- Design different layouts for different screens to take advantage of the available space.
- Different layouts can also be applied to a mobile device in different orientations.

18. Name: Resize Images

Domain Concerns:

- Ecosystem – There are many different devices and screen sizes to support

Benefits:

- Maintainability – Reduces the number of images that have to be made and stored on the server.

 Liability:

- Runs the risk of images not looking optimal for every platform as images need to be resized. This is most likely to occur for screen sizes that are at the extremes of standard devices i.e. very small of very large.

Solution:

- Provide a few image sizes for all devices and resize each image in the app for that device.
Common Situations/Problems of mobile app development

- Multiple platforms need to be supported.
- App depends heavily on data stored in the cloud.
- Must support many devices with different hardware capabilities.
- The battery drains quickly.
- Users need to be made aware of fresh content (or new event).
- App needs to run on devices with varying screen sizes.
- Data from the app needs to be shared with other people or integrated with other apps.
- App relies on the current user's context i.e. location, time etc.
- Users need to authenticate with the App.
- App frequently freezes or crashes.
- User needs to customise the behaviour of the app.
Appendix 2 - Questions
See Thesis Appendix User Interviews

Appendix 3 - Slides
Problem:

Simple development tasks take too long due to primitive tooling!!

Solution:

Better tools == Faster development time
Goal: Generate 80% of this app!


Roadmap

We are here

Generate 80% of an App

Build a Developer First Code Generation Tool

Build a Reference Architecture

Identify Repeating Patterns of App Development

Build and Engineer Mobile Apps (i.e. latest libraries, frameworks etc.)
Hello <participant name>,

I’m emailing to follow up on our discussion about our research on mobile app architecture. For an overview of our research and our goals please see the short slide deck attached.

Our main goal for this research is to automate the process of developing mobile apps and to do that we need to identify reusable design patterns.

So far we have identified 18 patterns by studying existing apps and seek to evaluate these with professionals in industry. Hopefully by interviewing your developers we can share knowledge to our mutual benefit. The design patterns we are interested in are generally applicable to the development of mobile apps and not app/business specific features. Interviews will run for 1 hour for each developer.

If you are interested in participating in this research please let me know so we can schedule a time for the interviews.

Thank you, Scott.
Appendix 5: Face-to-Face Interview Safety Protocol

Face-to-Face Interview Safety Protocol

The face-to-face interviews will be conducted by Scott Barnett and will be based on the following safety protocols.

Setting up audiotaping equipment
1. A portable audiotaping device will be used to record the interview session.

Pre-testing the interview protocol
1. The student researcher will discuss interview protocols and safety with her supervisors.
2. Prior to conducting research, at least two pilot interviews will be conducted with people outside the sample.

Communicating with potential participants
1. An introductory email will be sent out to the potential participants to brief them about the research project. They will also receive the Consent Information Statement. As the researcher will audiotape the interview, participants must be informed of the data recording process and processes in place to protect their confidentiality.
2. A follow-up email or telephone call will be used to set up the face-to-face interview appointment after the Signed Consent Form has been returned by each interview participant.

Scheduling each interview
1. The face-to-face interviews will be conducted at a location chosen by each participant.
2. The student researcher will ascertain that the place is suitable for interview such that there will not be any disturbance and there is no possibility of being overheard.
3. The research supervisors, Dr Rajesh Vasa and Prof John Grundy, will be notified about the time and location of the interviews. Where possible, the interview will be conducted during office hours.
4. The student researcher will keep a log file on which will be include participant contact information and the interview notes. This will be securely stored at all times in a locked filing cabinet in FSET office and never with the interview recordings.

During the interview
1. Prior to beginning the recording, the student researcher will ascertain the interviewee’s name and demographics.
2. The student researcher will ask the interviewee’s consent to begin recording.
3. The student researcher will summarize the project purpose, interview protocol and ask the interviewee to confirm they have read the Consent Information Statement Sheet. This will be confirmed by signing the Consent Form. The student will restate that the interviewee can terminate the interview at any time and can request that their recorded data be destroyed at any stage up to publication.
Appendix 5: Face-to-Face Interview Safety Protocol

4. The interview questions will be asked.
5. The interview will end and the student researcher will thank the interviewee and ask for the participant’s agreement for use of their answers in the research study.
6. Should the interviewees feel uncomfortable or the interview is terminated half way of the interview process for whatever reasons, the interview and consent are considered to be cancelled. The participant’s partial recording will be destroyed and the result will be noted in the log file.
## Appendix 6: Interview Protocol

### Interview Protocol

<table>
<thead>
<tr>
<th>Phase</th>
<th>Activities</th>
</tr>
</thead>
</table>
| Pre-interview | Contact participants by emailing them with a general introductory letter about the study. The following is an example of an email to be sent:  
Hello<participant name>,  
I’m emailing to follow up on our discussion about our research on mobile app architecture. For an overview of our research and our goals please see the short slide deck attached. Our main goal for this research is to automate the process of developing mobile apps and to do that we need to identify reusable design patterns.  
So far we have identified 18 patterns by studying existing apps and seek to evaluate these with professionals in industry. Hopefully by interviewing your developers we can share knowledge to our mutual benefit. The design patterns we are interested in are generally applicable to the development of mobile apps and not app/business specific features. Interviews will run for 1 hour for each developer.  
If you are interested in participating in this research please let me know so we can schedule a time for the interviews.  
If you have any questions regarding this study or any queries regarding the Project Information Consent Statements, you are welcome to contact the researcher. You can email Scott Barnett at sbarnett@swin.edu.au  
Thank you, Scott. |
|             | Prepare audiotape equipment – mobile phone device.                                                                                                                                                           |
|             | Prepare interviewees’ specific protocol sheets that contain a list of questions and responses. Each interviewee will be assigned a unique code. This code will be recorded onto the audiotape before we begin taping the particular interviewee’s responses. This will provide a method of matching the written record with the audio taped record for each interview session. |

| During interview | Script during the face-to-face interview:  
**1. Introduction**  
Hi [participant name]. How are you? Thank you for your time to |

1/6
Appendix 6: Interview Protocol

| participation in this interview. My name is [name]. I am a PhD student under Dr. Rajesh Vasa’s supervision at Swinburne University of Technology. |
| Show introductory slides Appendix 3. |
| I would like to stress that this participation is voluntary. You may stop the interview at any time. |
| For your information, this interview will be recorded using a taping machine. This taped record will be used to ensure accurate transcription of data. This interview session is expected to take approximately one hour. |
| Do you have any questions before we begin? |

2. Interview

At this stage, we will start the tape recording.

Start asking a list of interview questions. Appendix 2.

[When complete] Thank you. That completes this interview session. Turn off tape recorder. I will contact you if I need any clarification or further information. Please let me know how you would prefer to be contacted for follow up.

| Post interview |
| At the end of each interview, re-read the interview notes and transcribe it to electronic form. Transfer all the digital recording to a secure encrypted folder on a laptop. The original copy of the audio record will then be deleted from the taping device. |
Appendix 7. Consent Information Statement

Research Supervisor

Dr Rajesh Vasa  
Deputy Head, R&D Software Group  
Faculty of Science, Engineering and Technology (FSET)

Research Supervisor

Professor John Grundy  
Dean, School of Software and Electrical Engineering

Researcher

Scott Barnett  
PhD Candidate

Dear research participant,

You are invited to participate in this research study as part of the PhD research by Scott Barnett, supervised by Dr, Rajesh Vasa in the Faculty of Science, Engineering and Technology (FSET) of Swinburne University of Technology.

The Research Project

The purpose of this study is to ensure that our research into improving developer productivity is developer focused. Our tool is intended to speed up development by providing developers with a high level language that they can use to describe the features of an app. Once the description is complete the tool will generate an Android project with code that is ready to run on a device. The generated code is intended to be used as the scaffolding for a new project and will require a developer to complete it. This study will be broken up into two major parts: a questionnaire that will elicit current tools and practices used by professional app developers and a user evaluation with both students and developers.

Based on our current work, we have created a Conceptual Domain model that captures the core elements required to be understood by mobile app developers when build a mobile app. Using this model to analyse mobile apps, we have identified a number of reoccurring design patterns specific to mobile app development. We refer to these domain specific design patterns as Domain Tactics as they influence the quality attributes of an app. This study seeks to evaluate the Conceptual Domain model and our set of Domain Tactics with professional developers.

Phase 1: Current practices survey: Procedure

This part of the study will involve filling out the Current Practices Survey. Should you choose to participate in this research, you will be asked to fill out a simple online survey. The survey will have a number of questions about the processes and tools you use when starting a new mobile app project. The survey should take no longer than 15 minutes to complete.

Phase 2: Tool Evaluation: Procedure

If you agree to participate in this research, the tool evaluation will take place with you at a time and location of your choice. The tool evaluation will take place in the following manner:

• First you will be asked to complete a questionnaire about your experience developing mobile apps.
• We will then have a session to teach you to use RAPPT by walking through sample code and completing small tasks.
• After the learning tasks are complete there is another short questionnaire to gauge your confidence with RAPPT.
• Next there will be two timed development tasks one using the Android SDK (internet resources and 3rd party libraries are allowed) and one with using RAPPT (samples from the learning session will be available).
• There will also be a task where you will be asked to modify the code that is generated by RAPPT.
• Finally there will be the last questionnaire to fill out to provide your feedback and thoughts on using RAPPT.

The experiment has been designed to take approximately 2 hours.

Interview : Procedure

If you agree to participate in this research the interview will go for about 1 hour and involve questions on our findings to date. The procedure for the interview will be as follows:

1. Present research: You will be shown a short slide deck introducing this new phase of our research.
2. Introduce objective: To validate a set of Domain Tactics to ensure that the information they contain accurately models the domain of mobile app development or to determine whether they are of use to novice mobile app developers.
3. Ask background questions i.e. How many years have you been developing mobile apps?
4. Ask main interview questions about the concepts in the Conceptual Domain model and for the Domain Tactics. You will be shown 18 reoccurring patterns in mobile app development that we have identified and asked a number of questions about them i.e. Have you used this pattern before?
5. You will be then be asked to clarify the major findings at the conclusion of the interview.

Anonymity

No personal information that can be used to identify the participant will be gathered.

Privacy & Confidentiality

The interview survey results will only be used for academic research purposes. Individual responses will be kept confidential and will not be disclosed. All types of data will be stored electronically in an encrypted format, on a password protected computer and server in Swinburne University of Technology, Australia for 5 years of study. Paper-based collected data will be stored in a locked cabinet for 5 years in the FSET office and only accessible to the researchers. Paper documents will be deleted from the server after 5 years of study. The findings will be published in a publicly available PhD thesis and are likely to be published in conference and/or journal papers at a later date. Data will be presented in collective form and no individuals will be able to be identified in these publications.

Voluntary Participation

Your participation in this research is voluntary. You may discontinue participation at any time during the research activity and we will delete data collected from you (until publication, at which point deletion is no longer possible).
Research Output

The findings from this study will be published in a PhD thesis that will be publically available. Results may also be published in peer-reviewed academic conference and journals as outlined above.

Research Ethics

The project has been approved by the Swinburne University Human Research Ethics Committee.

Further information about the project – who to contact:

If there are any questions you have about the research project you are welcome to contact the researcher, Scott Barnett, or the principal investigator, Dr. Rajesh Vasa,

Dr Rajesh Vasa
Email: rvasa@swin.edu.au

Scott Barnett
Email: sbarnett@swin.edu.au

Concerns/complaints about the project – who to contact:

This project has been approved by or on behalf of Swinburne’s Human Research Ethics Committee (SUHREC) in line with the National Statement on Ethical Conduct in Human Research. If you have any concerns or complaints about the conduct of this project, you can contact:

Research Ethics Officer, Swinburne Research (H68),
Swinburne University of Technology, P O Box 218, HAWTHORN VIC 3122.
Tel (03) 9214 5218 or +61 3 9214 5218 or resethics@swin.edu.au
Participant Consent
1. I consent to participate in the project named above. I have been provided with a copy of the project consent information statement to which this consent form relates and any questions I have asked have been answered to my satisfaction.

2. In relation to this project, please circle your response to the following:
   - I agree to make myself available for further information if required Yes No
   - I agree to complete questionnaires asking me about the tools I use and the practices I follow when developing mobile apps Yes No
   - I agree to participate in completing a number of small tasks that will involve building small Android apps Yes No
   - I agree to participate in the interview on Domain Tactics Yes No

3. I acknowledge that:
   (a) my participation is voluntary and that I am free to withdraw from the project at any time without explanation;
   (b) the Swinburne project is for the purpose of research and not for profit;
   (c) any identifiable information about me that is gathered in the course of and as the result of my participating in this project will be (i) collected and retained for the purpose of this project in the manner described above and (ii) accessed and analysed by the researcher(s) for the purpose of conducting this project;
   (d) my anonymity is preserved and I will not be identified in publications or otherwise without my express written consent.
By signing this document I agree to participate in this project.

**Name of Participant:**

........................................................................................................

**Signature & Date:** .................................................................
Dear Dr Vasa,

**SHR Project 2014/313 Bootstrapping Mobile App Development**

Dr Rajesh Vasa, Mr Scott Barnett (Student) - FSET

Approved duration from: 19/02/2015 to 18/02/2017 [adjusted]

Modified: November 2015.

I refer to your e-mail of 17 November 2015 in which you requested a modification to the project by interviewing professional developers and students. The documentation was reviewed by a SHESC1 delegate.

I am pleased to advise that, as modified to date, the project/protocol may continue in line with standard ethics clearance conditions previously communicated and reprinted below.

Please contact me if you have any queries about on-going ethics clearance, citing the SUHREC project number. Copies of clearance emails should be retained as part of project record-keeping.

As before, best wishes for the project.

Kind regards,

Astrid Nordmann

---

Dr Astrid Nordmann

**Research Ethics Officer**

Swinburne Research (H68)

Swinburne University of Technology

PO Box 218, Hawthorn, VIC 3122

Tel: +613 9214 3845

Fax: +613 9214 5267

Email: g nordmann@swin.edu.au
Dear Dr Vasa,

SHR Project 2014/313 Bootstrapping Mobile App Development

Approved duration from: 19/02/2015 to 18/02/2017 [adjusted]

I refer to the ethical review of the above project protocol by a Subcommittee (SHESC1) of Swinburne’s Human Research Ethics Committee (SUHREC). Your responses to the review, as per the emails sent on 04 and 13 February 2015, were put to the Subcommittee delegate for consideration.

I am pleased to advise that, as submitted to date, the project may proceed in line with standard on-going ethics clearance conditions here outlined.

- All human research activity undertaken under Swinburne auspices must conform to Swinburne and external regulatory standards, including the current National Statement on Ethical Conduct in Human Research and with respect to secure data use, retention and disposal.

- The named Swinburne Chief Investigator/Supervisor remains responsible for any personnel appointed to or associated with the project being made aware of ethics clearance conditions, including research and consent procedures or instruments approved. Any change in chief investigator/supervisor requires timely notification and SUHREC endorsement.

- The above project has been approved as submitted for ethical review by or on behalf of SUHREC. Amendments to approved procedures or instruments ordinarily require prior ethical appraisal/clearance. SUHREC must be notified immediately or as soon as possible thereafter of (a) any serious or unexpected adverse effects on participants any redress measures; (b) proposed changes in protocols; and (c) unforeseen events which might affect continued ethical acceptability of the project.

- At a minimum, an annual report on the progress of the project is required as well as at the conclusion (or abandonment) of the project. Information on project monitoring, self-audits and progress reports can be found at: http://www.research.swinburne.edu.au/ethics/human/monitoring/ReportingChanges/

- A duly authorised external or internal audit of the project may be undertaken at any time.

Please contact the Research Ethics Office if you have any queries about on-going ethics clearance. The SHR project number should be quoted in communication. Researchers should retain a copy of this email as part of project recordkeeping.

Best wishes for the project.

Yours sincerely,

Astrid Nordmann

SHESC1 Secretary

----------------------------------------------
Dr Astrid Nordmann
Research Ethics Officer
Swinburne Research (H68)
Swinburne University of Technology
PO Box 218, Hawthorn, VIC 3122
Tel: +613 9214 3845
Fax: +613 9214 5267
Email: anordmann@swin.edu.au
----------------------------------------------
**SECTION A: PROJECT DETAILS**

<table>
<thead>
<tr>
<th>SUHREC Project No:</th>
<th>2014/313</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Title:</td>
<td>Extracting Technical Domain Knowledge to Improve Software Architecture (Formerly &quot;Bootstrapping Mobile App Development&quot;)</td>
</tr>
<tr>
<td>Chief Investigator/Supervisor:</td>
<td>A/Prof Jean-Guy Schneider</td>
</tr>
<tr>
<td>Faculty/Div:</td>
<td></td>
</tr>
<tr>
<td>Current Approved Duration:</td>
<td>19-02-2015 to 18-02-2017</td>
</tr>
<tr>
<td>Student Investigators:</td>
<td>Scott Barnett</td>
</tr>
<tr>
<td>Co-Investigators:</td>
<td>Prof John Grundy/A/Prof Rajesh Vasa</td>
</tr>
</tbody>
</table>

If there are any corrections or further details needed to the above, please clarify below.

NB If there are personnel changes not previously notified, please check C2 as applicable

### SECTION B: PROGRESS SUMMARY

**B1 Status of the Project (as at the date of this report)**

Please check [double-click] one or more of the following:

**PROGRESS REPORT**

- [ ] Data collection/access yet to commence  
  [Explain below then go to B3 then Sect D]
- [ ] Data collection/access commenced/continuing  
  [Explain below then go to B2 →]

**FINAL REPORT re HIGHER DEGREE RESEARCH STUDENT COMPONENT ONLY**

(*To be completed when a project is still continuing but the HDR student has completed their component of the project. Explain/complete rest of form as appropriate(B2 to D) to demonstrate ethical acceptability of the project to date or continuing.*)

- [ ] For thesis submission and examination purposes only
  
  Name of relevant student: ____________________________  
  ID number of student: ________

**FINAL REPORT**

- [ ] Data collection/access completed  
  [Explain below then go to Sect C →]
- [ ] Project Abandoned before data collection/access  
  [Explain below then go to Sect D]
- [ ] Project abandoned during or after data collection/access  
  [Explain below then go to Sect C →]
- [ ] Other
Please provide a brief explanation as to the project status indicated, including any delays.

**B2 Participants**

How many participants have been recruited to date?

**B3 Given any delays or other factor (as per B1 above), do you need an extension of ethics clearance without any other modification to the project?**

- [ ] No
- [ ] YES. If YES, please give the new date for end of data collection/access: 
  
  dd / mm / yyyy

NB information given below re changes may mean a simple extension of ethics clearance cannot be given and a separate ethics clearance application process for modifications needs to be followed.

**SECTION C: CONDUCT OF PROJECT**

**C1 Compliance with the approved protocol**

Has the project been conducted in line with the approved protocol, including standard and any special conditions of ethics clearance?

- [ ] Yes
- [ ] NO

If NO, please provide a brief explanation.

**C2 Project / Protocol Modifications / Additions**

Please check [double-click] one or more of the following if there are any:

- Changes/additions to project investigators (including students) and personnel accessing identifiable info
- Changes/additions re project personnel deriving a personal benefit from the research
- Changes/additions to the research protocol (title, aims, procedures, measures, sampling, etc)
- Changes/additions to the consent instruments/arrangements (including re personnel)
- Changes/additions to the recruitment material/methods
- Changes/additions to participant sampling or numbers
- Changes/additions to project resourcing (financing or otherwise)
- Other changes/additions

Have any of the above changes been put for ethical review?

- [ ] NO
- [ ] Yes

If No, please briefly explain the situation, including any separate submission(s) for ethics clearance for the modification(s) indicated. NB This form cannot be used for modification requests. Information on applying for ethics clearance for any modification(s) can be found here:
**C3 Project incidents**

Have there been any incidents that affected the conduct of the project or which have impacted adversely on participants and/or the researchers?

[ ] NO  [ ] YES

If YES, please provide a brief explanation including with respect to any reporting to Swinburne or other authority:

---

**Level of impact of incident:**

Would any of the incidents related above be considered serious?

[Serious adverse events (SAEs) include, eg, harm or distress to individuals or groups, loss of significant or sensitive data, breach of confidentiality.]

[ ] NO  [ ] YES

If YES, please provide a brief explanation including with respect to any reporting to Swinburne or other authority, where appropriate attaching a copy of the report(s):

---

**C4 Issues or experiences of ethical significance?**

Have there been any issues or experiences which have been or remain of ethical significance, especially as regards the ethical conduct of the project and/or project outcomes, including any actual or potential conflicts of interest not identified previously or formal complaints received/processed?

[ ] NO  [ ] YES

If yes, please briefly explain.

---

**C5 Project Outcomes (as at the date of this report)**

Please check all of the following:

- Compensatory payments made or prizes awarded and records kept
  [ ] Yes  [ ] No  [ ] n/a
- Student thesis/theses submitted for examination
  [ ] Yes  [ ] No  [ ] n/a
- Results have been published or presented
  [ ] Yes  [ ] No  [ ] n/a
- Results are to be published or presented
  [ ] Yes  [ ] No  [ ] n/a
- A lay summary of the project outcomes is given below^  
  [ ] Yes  [ ] No  [ ] n/a
- Project outcomes have been made available to participants
  [ ] Yes  [ ] No  [ ] n/a
- Project outcomes are to be made available to participants
  [ ] Yes  [ ] No  [ ] n/a
- Other
  [ ] Yes  [ ] No  [ ] n/a

If Yes or No, please provide a brief explanation as to the items checked as appropriate:

---

^Brief lay summary of project outcomes (not more than ¼ page):

...
C6 Study Materials/Documents

Please check one or more of the following:
- Project documents/material securely stored for the minimum period
- Project material to be made available for future research/other researchers. If so, in what form?

Briefly explain what storage or archiving has occurred, including the location(s) and length of secure storage as well as intended secure data disposal arrangements:

Are research material retention and disposal arrangements in line with what was outlined in the approved project protocol?  
- NO  
- YES

If NO, please explain why.

C7 Project Audits

Please check one or more of the following:
- Project self-audit(s) have been conducted during or at conclusion of project
  [Click here for a self-audit tool]
- Swinburne audit(s) have been conducted during or following completion of the project
- External audit(s) have been conducted during or following completion of the project

Please provide a brief explanation as to any audits conducted:

SECTION D: DECLARATION BY CHIEF INVESTIGATOR/SUPERVISOR

DECLARATION BY CHIEF INVESTIGATOR(s)/STUDENT SUPERVISOR(s)

I declare that the above report accurately reflects the outcome or progress of the project to date
I acknowledge that an internal Swinburne or external audit may be conducted on the conduct of the project and as regards secure data retention/disposal.

Signature & Date: .................................................................
Name of Signatory & Position: ....................................................

Student Investigator(s) (where possible)

I agree with the above declaration signed by the Chief Investigator/Supervisor

Signature & Date: .................................................................
Name of Student: .........................................................................

For Swinburne Research use only:

[ ] Progress Report  [ ] Final Report  [ ] HDR report
☐ 'Received' status on SEBAT, CI notified
☐ 'Satisfactory' status on SEBAT, CI notified
☐ Presented at SUHREC meeting _ _ _ _ _ _ _

Research Ethics Office Notes:

Date and initial: ______________
### Human Research Project Monitoring – Self Audit for Research Personnel

**All research personnel** should reflect on their research conduct and comply with guidelines for responsible research conduct. The project principal or chief investigator/supervisor or manager can or should use this form with each member of the research team. A record of each Self Audit undertaken should be retained in the project files.

<table>
<thead>
<tr>
<th>Question</th>
<th>True</th>
<th>False</th>
<th>N/A</th>
</tr>
</thead>
<tbody>
<tr>
<td>If I left suddenly, my project could be completed or replicated as the documentation for my projects is up to date, accessible, clearly ordered and comprehensible. The Principal Researcher/Coordinating Supervisor knows where to find all relevant documentation and has been provided with the passwords to the databases.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I am conducting the study in accordance with the protocol (procedures, documents, etc) approved by/on behalf of the Ethics Committee. Any modifications have been reported to and approved by the Committee and the relevant documents updated.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I have obtained signed consent forms from all participants (where applicable) and stored these securely. They are available for audit.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I have reported all serious and unexpected adverse incidents to Swinburne Research Ethics Office.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I have provided all study participants with a copy of the consent info statement (and, where appropriate, the consent form) approved by the ethics committee.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I have provided a translator and/or a translated copy of the consent info statement/consent form in his/her own language to all non-English speaking participants.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I have received ethics committee approval for all publicity or advertising text or material that seeks volunteers to participate in the study.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Approaches to potential participants have been made only by individuals aware of the study protocol and of the risks and inconveniences associated with participation (&amp; approved by the ethics committee).</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>All paper-based questionnaires have readily identifying information removed immediately after processing and are then matchable only with a code. The ‘code-key’ is stored separately and securely at all times.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Computer files containing study data are stored on a secure Swinburne network drive where they are regularly backed up.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>All computer files/diskUSB sticks/portable devices containing study data are protected by passwords or encryption.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Participants know who to contact if they have a question, concern/complaint or (where applicable) an emergency.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>There is a regular meeting of the study team including the Principal Researcher/s to discuss the progress of the study and a record of these meetings is maintained.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The team has submitted timely Progress/Final Reports.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**Further advice available from Swinburne Research Ethics Office, on 9214 5218, 9214 8145, 9214 8356.**

Form completed by:………………………………………………………………..  Date:………………

Project CI/Supervisor/manager:…………………………………………………………………..  Date:………………

---

1 Swinburne Research gratefully acknowledges permission by The Alfred Research and Ethics Unit and Department of Epidemiology and Preventive Medicine, Monash University to use/adapt the Self Audit Tool.
11 May 2016

Scott Barnett
27 Blamey Dr
MELTON, VIC 3337 Australia

Student ID: 6160506

Dear Scott

Subject: Change of thesis title

I am writing to advise you that your application to change the title of your thesis was approved in accordance with the provision of the Research Training Statement of Practice.

The approved title is, as requested:
Extracting Technical Domain Knowledge to Improve Software Architecture

Yours sincerely

Graduate Studies
Swinburne Research
Swinburne University of Technology
Tel +61 3 9214 3886
Email HDR@swin.edu.au

cc. Venkata Rajesh Vasa, Principal Coordinating Supervisor
Dear Jean-Guy,

Re: Final Report for the project 2014/313

'Extracting Technical Domain Knowledge to Improve Software Architecture (Formerly 'Bootstrapping Mobile App Development')' (Report Date: 04-05-2017)

The Final report for the above project has been processed and satisfies the reporting requirements set under the terms of ethics clearance.

Thank you for your attention to this matter.

Regards Research Ethics Team

Swinburne Research (H68), Swinburne University of Technology, PO Box 218, HAWTHORN VIC 3122, Tel: 03 9214 3845, Fax: 03 9214 5267, Email: resethics@swin.edu.au
Appendix F

Meta-model Process Examples

In this appendix we provide an example for each phase described in Chapter 5 on how to build a meta-model.

**Phase 1: Mining Screen Categories Example**

To illustrate how Phase 1 process worked we will walk through an example that involves a single iteration. We will use a Complex Screen shown in Figure F.1 from the Footy Live app1.

1. **Collect Data-intensive Mobile Apps.** Footy Live is an app that focuses on displaying information about Australian Football to the user. This information includes team statistics, live scores, current table standings and team tweets. As this app focuses on fetching information from a webservice and displaying it to the user it classifies as a data-intensive mobile app.

2. **Analyse Information Density for a Screen.** By the time we analyse this screen we had a framework in place shown in Figure F.2. When we analyse the screen shown in Figure F.1 we can see that not all of the information in the list is the same. Collections are screens that contain information that is of the same type but different instance such

---

as a list of contacts or a list of tv shows. The information in this list shows information for a news article using an image and a heading as well as a row about making sports bets. This is two different pieces of information shown in a single collection and cannot be classified as a Collection nor does it consist of multiple collections so cannot be considered a Multicollection we need a new category.

3a. Add New Category. We decided to create a new category for collections that contain different information shown in the same list. This is a pattern that also appears in news feeds on social media sites. The new category Complex now needs to be incorporated into the framework. As it is not a multicollection it needs to be inserted after that point and we also need a way to separate a Complex screen from a Collection. We added the question “Are the collection items predominantly in the same format?” to classify Complex screens. The new framework with the added category, Complex can be seen in Figure F.3.

3b. Classify Screen. Now that the framework has been extended and the new screen can now be classified we can continue on with the analysis process.
Figure F.2: Subset of classification framework for mobile app screens without support for Complex screen.

**Phase 2: Analysing App Composition**

High level abstractions are insufficient for addressing the needs of the technical domain. For example, dealing with Network Connectivity requires concepts for network requests, caching, data models and concurrency. To demonstrate the need for adding lower level abstractions to the meta-model to address technical domain concerns we will use MovieDB\(^2\) as an example app. MovieDB has a menu that is accessible from each screen from the top right hand corner shown open in Figure F.4. Each of these menu items navigates to the user to another screen in the app. In Android development these items are known as actions. The tabbar at the top of the screen also shows a magnifying glass that represents the search action highlighted in a red box.

1. **Meta-model Construction.** In our analysis of the information density platform idioms such as actions were not considered as they were not related to the information density. After analysing many apps along with the screen categories we realised we needed to have the concept of a tabbar, which would contain the actions and the concept of an action. We also found that actions displayed in a menu are commonly added to other elements such as list rows. Both of these concepts were included in the meta-model as a specific type of View. A View needed to contain other elements such as a label for each action and an event handler to represent what would happen after the user selected an item. The actions also had a distinct style, made accessible from clicking on a menu item with 3 dots and styled in a menu which is represented in the meta-model by the concept of Style. A subset of the relevant concepts from the meta-model for modelling actions is shown in Figure F.5.
2. **Building DSL.** When we discovered the concept of actions we first added the concept as a first class citizen to the meta-model. As we developed other apps and implemented additional features we realised...
that the characteristics of an action are shared with other concepts such as a screen. While a screen functions differently from an action and has an entirely different implementation the characteristics needed to describe a screen are similar to an action. For example, both have a unique style. In the case of the screen the style is consistent across the app but may also be specific to that screen. Whereas an action has a certain style across all implementations. The screen and the action both have events for a screen the events are associated with the app lifecycle and user interaction whereas an action has only user events. One of the key differences is that a screen may contain actions which is captured in our meta-model by allowing a View to consist of other Views as shown in Figure F.5.

3. Generate Existing Apps. The focus of this generation stage was to produce an app that closely resembled the analysed app. Not all of the features were implemented so the focus was on the navigation, UI elements and retrieval of data from web services. These 3 elements are common to many different apps which meant we could easily identify features that were not supported by our DSL. These generated apps were then passed to an evaluation stage where the key differences were compared.

For example, to build a Twitter client, the mobile app needs to authenticate with the Twitter API using OAuth2 authentication. This is in contrast with the MovieDB app that used an API key for authentication. The AML code required to configure OAuth2 for Twitter authentication and the generated login screen are shown in Figure F.6. From implementing the Twitter and MovieDB apps with their different authentication methods we realised the complexity of interfacing with existing APIs and the many different approaches to authentication. Our meta-model not only needed to support all of these but we needed to implement the main types of authentication and then find apps that used that method for further evaluation.


The generated apps were evaluated by comparing the app composition
with the original app. Figure F.7 shows a generated screen for the Twitter app compared with the original app. In the original app we see that there is a hamburger menu (1) that is not present in the generated screen. The hamburger menu slides out a pane from the left hand side showing navigation options to the user. In our meta-model the hamburger menu can be replicated by combining the concepts of view and events as such it does not represent a new concept – adding hamburger menu support is an implementation concern. We wanted to identify missing features that were unique to the meta-model and not just an implementation concern of the tool or DSL.

In the generated app we have 3 dots (2) which when selected displays a menu showing various actions a user can take. Whereas the original app has a single contextual action shown by the magnifying glass. This shows that our model supports the concept of an action and menu. In the original app the tabbar (3) shows customised icons for each of the tabs. Highly customised features that are specific to an app are not modelled by our system as our focus is on identifying features that

![AML code for OAuth authentication with the Twitter API and generated twitter login screen](image)

**Figure F.6:** AML code for OAuth authentication with the Twitter API (A) and generated twitter login screen (B)
are generally used in apps. This ensures that our tool supports a wide range of systems. Finally at (4) the original app shows advertising where as the generated app does not. From a UI perspective the content for the advertising content is captured by our system where as the integration into an ad network is currently unsupported.

Throughout this phase the evaluation criteria was heavily influenced by the constraints that we placed on the tool we were building. The key thing to remember here is that if we were building a tool that had a different set of design principles the meta-model would have been different. For example, consider a tool that needs a meta-model of data-intensive apps for use in a test generation tool. Test generation requires the concept of test data for generation which is required as a first class citizen for our tool that focuses on generating a scaffolding for an app.
**Phase 3: Domain Expert Feedback**

To demonstrate Phase 3 of building a meta-model, we will use one of the first iterations with professional developers that used a primitive version of our DSL as an example. A snapshot of the language that became the App Modelling Language (AML) can be seen in Figure F.8.

**Developer.** In one of the iterations for this phase we asked developers to build an example MovieDB app. This app required developers to describe the existing MovieDB API, define the UI and connect the data to the UI elements.

```aml
screen <detailT> (details, "Event Details") {
    resource(eventbrite, GET, objectT, "/discover/{id}/details") {
        view(viewId) {
            show-fields: [picture<pictureT>, title<stringT>,
                startDate<stringT>, location<stringT>,
                address<stringT>, distance<stringT>,
                shortDescription<stringT>, cost<stringT>]
            link(mapT, "Map", [latitude<stringT>, longitude<stringT>])
            link(getTickets, "Get Tickets")
            link(description, "Description")
        }
    }
    back: true
}
```

**Figure F.8:** AML code from an early version describing a screen that showed event details to the user.

**Feedback.** An initial version of our DSL connected the data source directly with the UI elements. This can be seen in Figure F.8 by the "show-fields" keyword which is used to indicate which fields to display to the user. Here is a case where the developers could use a declarative approach to specifying their app. They specified the data they wanted to display. How that data was structured, fetched and rendered was left up to the program. Developers found this confusing as they could not clearly see what the structure of the data was. Even though this required more work on their behalf they preferred to have the control over the structure and manually describe the data.
Evaluation.

As our DSL is intended to be used by practitioners it is paramount that we included the feedback from developers into the design of the language. One of the early findings of this evaluation phase was the need to model the architectural pattern Model View Controller (MVC). The MVC architectural pattern is commonly used in mobile apps and developers wanted to be able to directly express these concepts. Based on these recommendations we added separate blocks in the language to model each component of the MVC architectural pattern models, views and controller respectively.

Syntax was another aspect of the language that developers mentioned as needing improvement. Developers preferred to use a language that mapped to languages that they were already familiar with. This meant that the syntax of our language borrowed heavily from the C family of languages i.e. used curly braces {} to denote begin and end blocks.

Another major finding of testing our language with developers is that they preferred writing explicit verbose code rather than short clever code. One of the major influences upon the design of our language was to include a minimal number of concepts. However, we found that practically speaking the meta-model was desirable to keep to a minimal number of concepts additional abstractions were needed at the syntax level to ensure developers could use these concepts. A small meta-model ensured a few core concepts needed to be learnt where as the verbose syntax helped people manipulate the concepts in the meta-model.
Appendix G

User Evaluation

In this appendix are all of the files for the User Evaluation including the demographics questionnaire, post learning questionnaire and post evaluation questionnaire. The tasks that the participants had to complete to learn how to use RAPPT and to complete the evaluation are included as part of Appendix E. The User Evaluation is discussed in Chapter 7.
Comment report

Lists all the questions in the survey and displays all the comments made to these questions, if applicable.

Table of contents

Report info............................................................................................................................................................................................1

Question 1: How many years experience do you have in mobile app development?.................................................................2

Question 2: Gender?...........................................................................................................................................................................3

Question 3: How many apps have you built?......................................................................................................................................4

Question 4: How familiar are you with the Android Annotations library?.............................................................................................5

Question 5: What is your background?...............................................................................................................................................6

Question 6: What is the highest qualification you have?.....................................................................................................................7

Question 7: How would you rate your skills to be able to build a simple data driven mobile app?.....................................................8
<table>
<thead>
<tr>
<th>RespondentId</th>
<th>StartDate</th>
<th>CompletedDate</th>
<th>LanguageCode</th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
<th>Q5</th>
<th>Q5OtherOther</th>
<th>Q6</th>
<th>Q7</th>
</tr>
</thead>
<tbody>
<tr>
<td>1205361</td>
<td>2015.04.16</td>
<td>2015.04.16</td>
<td>en</td>
<td>&lt; 1</td>
<td>Male</td>
<td>0</td>
<td>Not Familiar</td>
<td>Researcher (IT related)</td>
<td>PhD</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>1205798</td>
<td>2015.04.17</td>
<td>2015.04.17</td>
<td>en</td>
<td>3-May</td>
<td>Male</td>
<td>6-Oct</td>
<td>Have heard of it</td>
<td>Developer</td>
<td>Degree</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>1205804</td>
<td>2015.04.17</td>
<td>2015.04.17</td>
<td>en</td>
<td>&lt; 1</td>
<td>Male</td>
<td>1-May</td>
<td>Familiar</td>
<td>Developer</td>
<td>Degree</td>
<td>5(Excellent)</td>
<td></td>
</tr>
<tr>
<td>1205811</td>
<td>2015.04.17</td>
<td>2015.04.17</td>
<td>en</td>
<td>1-Mar</td>
<td>Male</td>
<td>1-May</td>
<td>Have heard of it</td>
<td>Developer</td>
<td>Degree</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>1205816</td>
<td>2015.04.17</td>
<td>2015.04.17</td>
<td>en</td>
<td>1-Mar</td>
<td>Male</td>
<td>1-May</td>
<td>Have heard of it</td>
<td>Developer</td>
<td>Completed high school</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>1205817</td>
<td>2015.04.17</td>
<td>2015.04.17</td>
<td>en</td>
<td>1-Mar</td>
<td>Male</td>
<td>1-May</td>
<td>Familiar</td>
<td>Developer</td>
<td>Degree</td>
<td>5(Excellent)</td>
<td></td>
</tr>
<tr>
<td>1206638</td>
<td>2015.04.20</td>
<td>2015.04.20</td>
<td>en</td>
<td>1-Mar</td>
<td>Male</td>
<td>&gt;10</td>
<td>Familiar</td>
<td>IT/Business</td>
<td>Degree</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>1206678</td>
<td>2015.04.20</td>
<td>2015.04.20</td>
<td>en</td>
<td>&lt; 1</td>
<td>Male</td>
<td>1-May</td>
<td>Not Familiar</td>
<td>Developer</td>
<td>Degree</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>1206711</td>
<td>2015.04.20</td>
<td>2015.04.20</td>
<td>en</td>
<td>&lt; 1</td>
<td>Female</td>
<td>1-May</td>
<td>Familiar</td>
<td>Developer</td>
<td>Diploma</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>1206807</td>
<td>2015.04.20</td>
<td>2015.04.20</td>
<td>en</td>
<td>3-May</td>
<td>Male</td>
<td>1-May</td>
<td>Familiar</td>
<td>Developer</td>
<td>Completed high school</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>1207192</td>
<td>2015.04.21</td>
<td>2015.04.21</td>
<td>en</td>
<td>1-Mar</td>
<td>Male</td>
<td>1-May</td>
<td>Familiar</td>
<td>Developer</td>
<td>Degree</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>1207227</td>
<td>2015.04.21</td>
<td>2015.04.21</td>
<td>en</td>
<td>&lt; 1</td>
<td>Male</td>
<td>6-Oct</td>
<td>Familiar</td>
<td>Developer</td>
<td>Degree</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>1207254</td>
<td>2015.04.21</td>
<td>2015.04.21</td>
<td>en</td>
<td>1-Mar</td>
<td>Male</td>
<td>1-May</td>
<td>Familiar</td>
<td>Developer</td>
<td>Masters</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>1207409</td>
<td>2015.04.22</td>
<td>2015.04.22</td>
<td>en</td>
<td>&lt; 1</td>
<td>Male</td>
<td>0</td>
<td>Not Familiar</td>
<td>Developer</td>
<td>Degree</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>1207543</td>
<td>2015.04.22</td>
<td>2015.04.22</td>
<td>en</td>
<td>&lt; 1</td>
<td>Male</td>
<td>1-May</td>
<td>Not Familiar</td>
<td>Researcher (IT related)</td>
<td>PhD</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>1207592</td>
<td>2015.04.22</td>
<td>2015.04.22</td>
<td>en</td>
<td>1-Mar</td>
<td>Male</td>
<td>1-May</td>
<td>Not Familiar</td>
<td>Researcher (IT related)</td>
<td>Masters</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>1207649</td>
<td>2015.04.22</td>
<td>2015.04.22</td>
<td>en</td>
<td>&lt; 1</td>
<td>Female</td>
<td>0</td>
<td>Not Familiar</td>
<td>IT/Business</td>
<td>Degree</td>
<td>1(Poor)</td>
<td></td>
</tr>
<tr>
<td>1207652</td>
<td>2015.04.22</td>
<td>2015.04.22</td>
<td>en</td>
<td>&lt; 1</td>
<td>Male</td>
<td>1-May</td>
<td>Not Familiar</td>
<td>Developer</td>
<td>Degree</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>1208266</td>
<td>2015.04.24</td>
<td>2015.04.24</td>
<td>en</td>
<td>&lt; 1</td>
<td>Female</td>
<td>0</td>
<td>Not Familiar</td>
<td>Researcher (IT related)</td>
<td>PhD</td>
<td></td>
<td>2</td>
</tr>
</tbody>
</table>
Comment report

Lists all the questions in the survey and displays all the comments made to these questions, if applicable.

Table of contents

Report info............................................................................................................................................................................................1

Question 1: How confident are you to build a small data-driven app with RAPPT?.................................................................2

Question 2: How well do you understand the constructs in the RAPPT?....................................................................................3

Question 3: How would you rank your abilities as an Android app developer?..........................................................................4
<table>
<thead>
<tr>
<th>RespondentId</th>
<th>StartDate</th>
<th>CompletedDate</th>
<th>LanguageCode</th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1205381</td>
<td>2015.04.16 11:59</td>
<td>2015.04.16 11:59</td>
<td>en</td>
<td>3</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>1205806</td>
<td>2015.04.17 15:10</td>
<td>2015.04.17 15:10</td>
<td>en</td>
<td>4</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>1205819</td>
<td>2015.04.17 15:57</td>
<td>2015.04.17 15:57</td>
<td>en</td>
<td>2</td>
<td>2</td>
<td>1(Poor)</td>
</tr>
<tr>
<td>1206643</td>
<td>2015.04.20 09:51</td>
<td>2015.04.20 09:51</td>
<td>en</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>1206724</td>
<td>2015.04.20 12:14</td>
<td>2015.04.20 12:14</td>
<td>en</td>
<td>3</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>1206845</td>
<td>2015.04.20 16:14</td>
<td>2015.04.20 16:14</td>
<td>en</td>
<td>3</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>1207223</td>
<td>2015.04.21 16:31</td>
<td>2015.04.21 16:31</td>
<td>en</td>
<td>4</td>
<td>5(Very well)</td>
<td>4</td>
</tr>
<tr>
<td>1207253</td>
<td>2015.04.21 16:31</td>
<td>2015.04.21 16:31</td>
<td>en</td>
<td>3</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>1207257</td>
<td>2015.04.21 16:32</td>
<td>2015.04.21 16:32</td>
<td>en</td>
<td>3</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>1207413</td>
<td>2015.04.22 10:40</td>
<td>2015.04.22 10:40</td>
<td>en</td>
<td>4</td>
<td>4</td>
<td>1(Poor)</td>
</tr>
<tr>
<td>1207650</td>
<td>2015.04.22 14:28</td>
<td>2015.04.22 14:28</td>
<td>en</td>
<td>4</td>
<td>5(Very well)</td>
<td>4</td>
</tr>
<tr>
<td>1207659</td>
<td>2015.04.22 14:41</td>
<td>2015.04.22 14:41</td>
<td>en</td>
<td>2</td>
<td>4</td>
<td>1(Poor)</td>
</tr>
</tbody>
</table>
Code report

Lists all the questions in the survey and displays all the comments made to these questions, if applicable.

Table of contents

Report info............................................................................................................................................................................................1

Question 1: It was easy to use RAPPT...............................................................................................................................................2
Question 2: It is easy to understand what each icon represents. .......................................................................................................3
Question 3: It was easy to learn how to use RAPPT...........................................................................................................................4
Question 4: It was easy to create a screen with a list. .......................................................................................................................5
Question 5: It was easy to load data from an API and render it to the screen. ..................................................................................6
Question 6: It was easy to add a tabbar to the app. ...........................................................................................................................7
Question 7: It is easy to remember how to use RAPPT......................................................................................................................8
Question 8: It is easy to avoid making errors or mistakes. .................................................................................................................9
Question 9: If you had to start to develop an app again what would you prefer to use?.................................................................10
Question 10: RAPPT makes you more productive when starting a new project than using templates from an.................................11
Question 11: RAPPT helps you avoid common errors when starting a new project........................................................................12
Question 12: The concepts in RAPPT are sufficient for modelling a mobile app..............................................................................13
Question 13: What concepts would you like to see included in the visual model?............................................................................14
Question 14: What concepts would you like to see included in AML?..............................................................................................15
Question 15: What would you like to see in the visual model that is already in AML?.................................................................16
Question 16: What would you like to see in AML that is already in the visual model?.......................................................................17
Question 17: Using RAPPT is more efficient than starting with a raw Android project.................................................................18
Question 18: RAPPT is useful for mobile app development.............................................................................................................19
Question 19: You are satisfied with using RAPPT. ..........................................................................................................................20
Question 20: You would recommend RAPPT to a friend. ................................................................................................................21
Question 21: It was fun to use. .........................................................................................................................................................22
Question 22: In your opinion, which types of apps are not suited to be developed with RAPPT?....................................................23
Question 23: What did you like the most about RAPPT?...................................................................................................................24
Question 24: What did you dislike the most about RAPPT?.............................................................................................................25
Question 25: What needs to be added or fixed for you to use RAPPT for all your data intensive Android.......................................26
<table>
<thead>
<tr>
<th>RespondentId</th>
<th>StartDate</th>
<th>CompletedDate</th>
<th>LanguageCode</th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
<th>Q5</th>
<th>Q6</th>
<th>Q7</th>
<th>Q8</th>
<th>Q9</th>
</tr>
</thead>
<tbody>
<tr>
<td>1205416</td>
<td>2015.04.16</td>
<td>2015.04.16</td>
<td>en</td>
<td>4</td>
<td>5 (Strongly Agree)</td>
<td>5 (Strongly Agree)</td>
<td>4</td>
<td>4 (Strongly Agree)</td>
<td>5 (Strongly Agree)</td>
<td>5 (Strongly Agree)</td>
<td>5 (Strongly Agree)</td>
<td>RAPPT</td>
</tr>
<tr>
<td>1205797</td>
<td>2015.04.17</td>
<td>2015.04.17</td>
<td>en</td>
<td>5 (Strongly Agree)</td>
<td>5 (Strongly Agree)</td>
<td>4</td>
<td>5 (Strongly Agree)</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>RAPPT</td>
</tr>
<tr>
<td>1205813</td>
<td>2015.04.17</td>
<td>2015.04.17</td>
<td>en</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3 (Strongly Agree)</td>
<td>3</td>
<td>2</td>
<td>RAPPT</td>
<td></td>
</tr>
<tr>
<td>1205824</td>
<td>2015.04.17</td>
<td>2015.04.17</td>
<td>en</td>
<td>4</td>
<td>5 (Strongly Agree)</td>
<td>4</td>
<td>5 (Strongly Agree)</td>
<td>4</td>
<td>5 (Strongly Agree)</td>
<td>4</td>
<td>4</td>
<td>Programming</td>
</tr>
<tr>
<td>1205832</td>
<td>2015.04.17</td>
<td>2015.04.17</td>
<td>en</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>5 (Strongly Agree)</td>
<td>5 (Strongly Agree)</td>
<td>4</td>
<td>4</td>
<td>RAPPT</td>
</tr>
<tr>
<td>1205837</td>
<td>2015.04.17</td>
<td>2015.04.17</td>
<td>en</td>
<td>5 (Strongly Agree)</td>
<td>5 (Strongly Agree)</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>5 (Strongly Agree)</td>
<td>4</td>
<td>3</td>
<td>RAPPT</td>
</tr>
<tr>
<td>1206672</td>
<td>2015.04.20</td>
<td>2015.04.20</td>
<td>en</td>
<td>5 (Strongly Agree)</td>
<td>5 (Strongly Agree)</td>
<td>4</td>
<td>4</td>
<td>5 (Strongly Agree)</td>
<td>5 (Strongly Agree)</td>
<td>4</td>
<td>3</td>
<td>RAPPT</td>
</tr>
</tbody>
</table>
1206733 2015.04.20 12:47 2015.04.20 12:48 en 4 4 4 4 4 5(Strongly Agree) 3 2 Programing

1206772 2015.04.20 15:23 2015.04.20 15:26 en 4 5(Strongly Agree) 4 5(Strongly Agree) 4 5(Strongly Agree) 4 3 RAPPT

1206803 2015.04.20 14:50 2015.04.20 14:50 en 3 4 4 3 4 3 3 3 Programing

1207103 2015.04.21 11:53 2015.04.21 11:58 en 3 5(Strongly Agree) 4 5(Strongly Agree) 4 5(Strongly Agree) 4 3 RAPPT

1207275 2015.04.21 17:33 2015.04.21 17:33 en 4 5(Strongly Agree) 4 4 5(Strongly Agree) 5(Strongly Agree) 3 4 RAPPT

1207411 2015.04.22 10:53 2015.04.22 10:53 en 5(Strongly Agree) 5(Strongly Agree) 5(Strongly Agree) 5(Strongly Agree) 5(Strongly Agree) 5(Strongly Agree) 5(Strongly Agree) 4 4 RAPPT
<table>
<thead>
<tr>
<th>Timestamp</th>
<th>Rating</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>2015.04.22 11:11</td>
<td>4</td>
<td>5(S) 2(S) 4(4)</td>
</tr>
<tr>
<td>2015.04.22 12:5</td>
<td>5</td>
<td>4(S) 4(4) 2(2)</td>
</tr>
<tr>
<td>2015.04.22 14:33</td>
<td>5</td>
<td>4(S) 4(4) 3(3)</td>
</tr>
<tr>
<td>2015.04.22 14:57</td>
<td>2</td>
<td>4(4) 3(3) 4(4)</td>
</tr>
<tr>
<td>2015.04.22 15:8</td>
<td>4</td>
<td>3(3) 2(2) 3(3)</td>
</tr>
<tr>
<td>2015.04.22 16:23</td>
<td>3</td>
<td>4(4) 3(3) 4(4)</td>
</tr>
<tr>
<td>2015.04.24 16:34</td>
<td>4</td>
<td>5(S) 4(4) 5(S)</td>
</tr>
<tr>
<td>Question</td>
<td>Rating</td>
<td>Comment</td>
</tr>
<tr>
<td>----------</td>
<td>--------</td>
<td>---------</td>
</tr>
<tr>
<td>Q10</td>
<td>5(Strongly Agree)</td>
<td>more fine grained toolbox and page development UIs, for example would like to have the labels in the drag and drop page.</td>
</tr>
<tr>
<td>Q11</td>
<td>4</td>
<td>Perhaps I can specify APIs, data sources more abstractly now that the code gets generated automatically.</td>
</tr>
<tr>
<td>Q12</td>
<td>3</td>
<td>Label specification and other UI elements. Perhaps bindings as elements to drag and drop?</td>
</tr>
<tr>
<td>Q13</td>
<td>Free Text</td>
<td>I would prefer the visual elements over AML. Move everything to visual elements please!</td>
</tr>
<tr>
<td>Q15</td>
<td>Free Text</td>
<td>Live preview of how it will be looked when app is installed.</td>
</tr>
<tr>
<td>Q16</td>
<td>Free Text</td>
<td>Syntax auto complete, list declaration and initiation (row content), so then in AML we can just connect data model with it.</td>
</tr>
<tr>
<td>Q17</td>
<td>5(Strongly Agree)</td>
<td>Module composition and plugins.</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>Theming.</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>More objects (like buttons and what not), that you can drag into the screens.</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>Maybe realtime checking for simple errors (like for method brackets not closed), which underlines the text.</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>Something to signify activities are linked via a mutual tabbar, so separate areas of the app with different tabbars can be differentiated (with colours for e.g.).</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>I expected curly brace pairs to create lexical scopes. Data sources in separate screens should be able to have the same name. Optional naming for things I don't care about (e.g. view groups, list rows, tab bars, &amp; labels with no data binding).</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>For each screen, a representation of the views it contains. Some sort of representation of the API, and how each screen interacts with its endpoints.</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>Drawer Toolbar Buttons Browser</td>
</tr>
<tr>
<td>Rating</td>
<td>Comment</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>---------</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>A way to setup the layout of the individual screen.</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>A way to setup which variables are being passed between screens.</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>Designer for screens to change the look and feel of the apps.</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>Some further usage of its drag-n-drop to perhaps have visual data models, or even some rudimentary element positioning within the screens.</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>More input types, although there could very well be support for this already and I am just unaware. The ability to have one screen and switchable data sources.</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>Some more ability to create connections. Perhaps even a visual data model that relations can be created between.</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>More visual components</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>Syntax for adding more components</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>Editing text in the visual model for individual screens</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>I would like to see code suggestions</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>I would like to have the ability to make connections between UI elements and other screens. For example, connection between tab bar and other screen.</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>Nothing specific because AML is more advanced than visual model</td>
<td>1</td>
</tr>
</tbody>
</table>

**Ratings:**

- Strongly Agree (5)
- Agree (4)
- Neutral (3)
- Disagree (2)
- Strongly Disagree (1)
<table>
<thead>
<tr>
<th>Score</th>
<th>Feedback</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>A list placeholder just for design purposes</td>
</tr>
<tr>
<td>4</td>
<td>Maybe intents? e.g., grab a photo or something might be useful for apps with a 'social' component.</td>
</tr>
<tr>
<td>5</td>
<td>Lists, as noted above. Also is this a trick question?</td>
</tr>
<tr>
<td>4</td>
<td>Video and Audio concepts... drag and drop audio/video files</td>
</tr>
<tr>
<td>5</td>
<td>Android Emulator so as to be able to visually see and interact with the output</td>
</tr>
<tr>
<td>2</td>
<td>More difficult to add UI components such as scrolling full page text, rich text content, side swipe menus etc.</td>
</tr>
<tr>
<td>3</td>
<td>Live syntax highlighting / errors. Get the code only for a screen or other component when you double click. Binding of events.</td>
</tr>
<tr>
<td>4</td>
<td>Visual indicators for the tab bar navigation that can be turned on and off</td>
</tr>
<tr>
<td>5</td>
<td>So far so good</td>
</tr>
<tr>
<td>5</td>
<td>So far so good</td>
</tr>
<tr>
<td>5</td>
<td>So far so good</td>
</tr>
<tr>
<td>5</td>
<td>Visual indicators for the tab bar navigation that can be turned on and off</td>
</tr>
<tr>
<td>5</td>
<td>So far so good</td>
</tr>
<tr>
<td>5</td>
<td>So far so good</td>
</tr>
<tr>
<td>5</td>
<td>So far so good</td>
</tr>
<tr>
<td>5</td>
<td>So far so good</td>
</tr>
<tr>
<td>5</td>
<td>So far so good</td>
</tr>
</tbody>
</table>

Note: The table reflects the scores given to various features or concepts.
<table>
<thead>
<tr>
<th>Q18</th>
<th>Q19</th>
<th>Q20</th>
<th>Q21</th>
<th>Q22</th>
<th>Q23</th>
<th>Q24</th>
<th>Q25</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>-</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>cannot think of any!</td>
<td>The drag and drop of pages to start development, the lack of previous knowledge of API and app development</td>
<td>The error re-posting could be improved. Perhaps the code generation and installing could be included as part of the tool as well, not a separate Android studio app</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>It's able to produce ready to use app instantly</td>
<td>The high-level abstraction and terseness of the code.</td>
<td>More fine-grained elements in the tool box to be used in page element development, less need for writing code (I am biased here!)</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>Stand alone app (no need to connect to any server)</td>
<td>It's able to produce ready to use app instantly</td>
<td>App browser can be resize (cannot see items when it is too wide)</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>Highly visually customised apps.</td>
<td>Highly visually customised apps.</td>
<td>Modularity and the idea of evolving a codebase, as opposed to just using it as a once-off bootstrapper.</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>Apps with a single or two screens that require lots of information and data manipulation</td>
<td>I was able to create, compile and run a project with no trouble, with a headache</td>
<td>More visual objects for the designer screen!</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>Games with complex interaction</td>
<td>The website is pretty and the DSL straight-forward.</td>
<td>The samples need descriptions. It was difficult to see (at a glance) how they differed. It is unclear as to how arguments passed to screens are bound. For example, <code>alpha2Code</code> (in the samples) is not explained. More IDE features, auto-completion being the primary one.</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>Those with complex UIs, many background services, and that interact with non-RESTful APIs.</td>
<td>Games with complex interaction</td>
<td>- Tab spacing is inconsistent (generated is 4 spaces, tab is 2 spaces) - Response field name conflicts with keywords (results.title -&gt; screen.title) - Conflict in endpoint naming and screen naming (even with _ placed in front of screen name) - Naming conflict with datasource in different screens. You shouldn't need to have datasource0, datasource1, etc for different screens</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>Games Browser Chromecast based Media based</td>
<td>Speed of getting an app set up</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>4 (Strongly Agree)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>------------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Larger apps with more complicated functionality requirements.</td>
<td>The simplicity of setting up the core elements of an app.</td>
<td>The random auto formatting. My OCD does not like code being out of line and all over the place.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Games, and apps which require customised UI elements or are very complex.</td>
<td>Quick and has examples</td>
<td>Improvements to the titles of the example AML code. They were not entirely clear without clicking in each of them what they were for.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Saving / Retrieving (doesn't have to be cloud based)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Predictive code completion / hints would be nice too.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>3</th>
<th>4</th>
<th>4 (Strongly Agree)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Custom apps, that don't follow standard Android design. As far as I know, I don't know if I could include, for example, maps or custom elements using RAAPT.</td>
<td>AML over using the the IDE. It was much faster to create new screens and tabs using AML.</td>
<td>I kept trying to save my code... which doesn't work in browser.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Custom element design via RAAPT.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>4</th>
<th>4</th>
<th>4 (Strongly Agree)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anything that is not data driven probably wouldn't be ideal, for example a game isn't feasible. Also if there are any cases where you want to use libraries (for example Fabric.io) to display data there is obviously no support.</td>
<td>I liked the combination between the visual model and the AML to create the app.</td>
<td>I felt that the visual model was really only around to quickly create the AML for the screens. There was no way to remove the tabbar from a screen, it had to be changed in the AML. AML felt it missing reference of possible structs.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>I don't feel like anything needs to be fixed, however I do believe that if some of the features/concerns stated in previous questions were addressed it might make the experience a little crisper.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>4</th>
<th>5 (Strongly Agree)</th>
<th>5 (Strongly Agree)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not sure</td>
<td>the ease and simplicity saves a visual model sometimes LOT of time (seriously!) appears buggy</td>
<td>More components to the visual model would be a good addition. Not able to suggest any fixes at the moment as I need more time to play around with Rappt to determine any fixes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>5 (Strongly Agree)</th>
<th>5 (Strongly Agree)</th>
<th>5 (Strongly Agree)</th>
</tr>
</thead>
<tbody>
<tr>
<td>I think games are not suited to be developed with RAAPT.</td>
<td>The interface is easy to use and the AML language is easy to learn.</td>
<td>The ability to use the local database.</td>
</tr>
<tr>
<td>Rating</td>
<td>Comment</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>---------</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td><strong>Strongly Agree</strong></td>
<td>Non-data driven apps e.g. games, or apps that integrate more strongly with the Android OS (camera, intents). The model block was a really nice way of specifying API bindings, with the arrows clearly showing the flow of data. There isn't really a way to see if the app actually works without having to tangle with Android Studio; I'm not sure if that's a sticking point for actual Android devs but if a Rappt -&gt; AS -&gt; Rappt cycle develops during dev that might be annoying.</td>
</tr>
<tr>
<td>5</td>
<td><strong>Strongly Agree</strong></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td><strong>Strongly Agree</strong></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td><strong>Strongly Agree</strong></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>Ease of creating project and integrating APIs</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>Gaming and complex</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td><strong>Strongly Agree</strong></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td><strong>Strongly Agree</strong></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td><strong>Strongly Agree</strong></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>Fully scoped projects.</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>Complex apps.</td>
</tr>
<tr>
<td>5</td>
<td><strong>Strongly Agree</strong></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td><strong>Strongly Agree</strong></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td><strong>Strongly Agree</strong></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>Applications that require lots of user interactions, such as phone game.</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td><strong>Strongly Agree</strong></td>
<td></td>
</tr>
</tbody>
</table>
Appendix H

User Evaluation Criteria

Participants who completed the User Evaluation had to hand in their app to ensure they had completed the task correctly. The User Evaluation Criteria is referenced from Chapter 7 Each completed app was subject to the following criteria:

- The app had been built using RAPPT.
- There was no code in the unmodified tasks that had been added to the project.
- The app compiles and runs on the device without crashing.
- Yet Another Weather App displays the weather for a given city.
- Another Currency Exchange App fetches the appropriate currency.
- The modified app compiles and runs on the device.
- No additional functionality was added to the app.
Appendix I

Mobile App Tools

Below are the mobile app tools (and URLs) that were analysed as part of the literature review discussed in Chapter 2. Concepts from these tools helped inform the design decisions made when building RAPPT, covered in Chapter 7.

POP - http://popapp.in/#overview
Codiqa - http://www.codiqa.com/
Proto - http://proto.io/
Fluid UI - https://www.fluidui.com/
App in seconds - http://appinseconds.com/
Jimu - http://jimulabs.com/
Tapcanvas- http://www.tapcanvas.com/
Divshot - http://www.divshot.com/
Moqups - https://moqups.com/#/
Balsamiq - http://www.balsamiq.com/products/mockups
Justinmind - http://www.justinmind.com/
Axure -http://www.axure.com/
Protoshare - http://www.protoshare.com/
HotGloo - http://www.hotgloo.com/
Pixate - http://www.pixate.com/
Mockingbot - https://mockingbot.com/
Appcooker - http://www.appcooker.com/
Blueprint - http://www.groosoft.com/blueprint/
FieldTestapp - http://fieldtestapp.com/
FileSquare http://www.filesq.com/#
FlairBuilder - http://flairbuilder.com/
EndLoop - http://www.endloop.ca/imockups/
Invisionapp - http://www.invisionapp.com/
Keynotopia - http://keynotopia.com/
Mockability - http://www.mockability.com/
Prototypesapp - http://prototypesapp.com/
Realizerapp - http://realizerapp.com/
Refine - http://refine.io/
AppBreeder - http://www.appbreeder.com/
BuildAnApp - http://www.buildanapp.com/home
Fivespark - http://www.fivespark.com/
Mippin - http://mippin.com/web/
MobBase - http://www.mobbase.com/
mobileappwizard - http://www.mobileappwizard.com/
mobileroadie - http://www.mobileroadie.com/
Mobincube - http://www.mobincube.com/
MyAppBuilder - http://myappbuilder.com/
Shoutem - http://www.shoutem.com/
Swebapps - http://www.swebapps.com/
TapLynx - http://www.taplynx.com/
The - App Builder http://www.theappbuilder.com/
Tiggzi - http://appery.io/
Droidux - http://www.droidux.com/
FlinTo - https://www.flinTo.com/
Marvel - https://marvelapp.com/
Skala - http://bjango.com/mac/skalapreview/
AppNotch - http://www.appnotch.com/pro/
Canappi - http://www.canappi.com/
Applause - https://github.com/applause/applause
MetaCase - http://www.metacase.com
TAP - http://ipro.to/tap/ipad/build/
Catrobat - http://developer.catrobat.org/
Napkee - http://www.napkee.com/
Firebase - https://www.firebase.com/
Appscend - http://appscend.com/
Buzz touch - http://www.buzztouch.com/
AppBuilder - http://www.telerik.com/appbuilder
Appmkr - http://appmakr.com/
Basic4Android - http://www.basic4ppc.com/
Appendix J

Analysed Apps

An essential part of the analysing the domain to build a technical domain model was the study of mobile apps as discussed in Chapter 4. In this appendix we list all of the apps that were used in the analysis stage and their App Id's mentioned on the Play Store. At the time of analysis the apps were available for download from https://play.google.com/store/apps/details?id={App Id}. 
Table J.1: Table showing data-intensive mobile apps used in analysis.

<table>
<thead>
<tr>
<th>App Name</th>
<th>App Id</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prompa</td>
<td>net.prompa.production.release</td>
</tr>
<tr>
<td>TED</td>
<td>com.ted.android</td>
</tr>
<tr>
<td>Gmail</td>
<td>com.google.android.gm</td>
</tr>
<tr>
<td>UBank</td>
<td>com.ubank.internetbanking</td>
</tr>
<tr>
<td>Eventbrite</td>
<td>com.eventbrite.attendee</td>
</tr>
<tr>
<td>ABC</td>
<td>android.AbcApplication</td>
</tr>
<tr>
<td>Shopping List</td>
<td>com.shoppinglist</td>
</tr>
<tr>
<td>Australia Post</td>
<td>au.com.auspost.m</td>
</tr>
<tr>
<td>Westpac Mobile Banking</td>
<td>org.westpac.bank</td>
</tr>
<tr>
<td>Barcode Scanner</td>
<td>com.google.zxing.client.android</td>
</tr>
<tr>
<td>Twitter</td>
<td>com.twitter.android</td>
</tr>
<tr>
<td>Flipboard</td>
<td>flipboard.app</td>
</tr>
<tr>
<td>Google I/O</td>
<td>com.google.samples.apps.iosched</td>
</tr>
<tr>
<td>HN</td>
<td>com.manuelmaly.hn</td>
</tr>
<tr>
<td>Gumtree AU</td>
<td>com.ebay.gumtree.au</td>
</tr>
<tr>
<td>Evernote</td>
<td>com.evernote</td>
</tr>
<tr>
<td>IM Db</td>
<td>com.imdb.mobile</td>
</tr>
<tr>
<td>MovieBase</td>
<td>de.linuxwhatelse.android.moviebase</td>
</tr>
<tr>
<td>Cricket Australia</td>
<td>au.com.cricket</td>
</tr>
<tr>
<td>Smart Measure</td>
<td>kr.sira.measure</td>
</tr>
<tr>
<td>GeoQuiz</td>
<td>org.urban.android.quiz.geographyquiz</td>
</tr>
<tr>
<td>V/Line</td>
<td>com.vortilla.myline</td>
</tr>
<tr>
<td>Melbourne Train</td>
<td>com.tincan.traintimes</td>
</tr>
<tr>
<td>Facebook</td>
<td>com.facebook.katana</td>
</tr>
<tr>
<td>WhatsApp Message</td>
<td>com.whatsapp</td>
</tr>
<tr>
<td>YOW! 2012</td>
<td>com.confer</td>
</tr>
<tr>
<td>Coles App</td>
<td>com.coles.android.shipmate</td>
</tr>
<tr>
<td>eBay</td>
<td>com.ebay.mobile</td>
</tr>
<tr>
<td>Woolworths</td>
<td>com.woolworths</td>
</tr>
<tr>
<td>NYTimes</td>
<td>com.nytimes.android</td>
</tr>
</tbody>
</table>
Appendix K

Prompa Case Study

The Prompa app was used as part of an industry case study as discussed in Chapter 7. In this appendix we use the technical domain model to analyse the way the app was built.

Prompa\(^1\) is a commercial app that provides a service for rostering staff for a shift. Prompa can be used by supervisors to allocate staff for a shift and view who has arrived on time. Staff can accept shifts they have been assigned too and submit timesheets. There is also a ranking mechanism for staff to rank work and for shift supervisors to rank employees.

Example 4: Hardware Constraints, Threading Model and Lifecycle.

Description: The Prompa app requires users to login using their email and password to use the app. Users also have to fill out a simple form to submit reviews of the work they have completed.

Issues: Filling out forms on mobile devices is a tedious process due to their size (Hardware Constraints). Entering text is particularly tedious as touch based mobile devices popup a virtual keyboard that fills a

\(^1\)https://play.google.com/store/apps/details?id=org.swin.prompa&hl=en
significant portion of the screen. Developers are well advised\(^2,3\) to limit the number of forms in an app to create a better user experience.

**Tradeoffs:** In the situation where a form is unavoidable there are a number of strategies that can make the experience smoother: Reduce the number of text fields, keep the form to a size that fits on a single screen and pre-fill as much of the form as possible. Pre-filling data requires caching of previously entered data which needs to be factored into the architecture of the whole app. Login screens are a common form that appears in a lot of apps. When caching authentication tokens this raises security concerns which also need to be considered.

**Solution:** Once the user has logged in, Prompa remembers the authentication details which allows users to by-pass the login screen on subsequent visits to the app. Prompa provides an option within the app for users to log out if they wish. When a user logs out Prompa remembers just the email address and prefills the appropriate field in the login screen next time the user visits the app. To do this appropriately the app needs to cache the appropriate authentication details of the app and restore the value in the correct state of the app’s Lifecycle. Web developers do not face this issue as caching of data is left to the concern of the browser. Likewise desktop developers traditionally do not have login screens for their applications.

*Example 5: Sensing, Threading Model and Hardware Constraints.*

**Description:** One of the core features of Prompa is its ability to detect if a worker has arrived at a shift or if they are still at the site of the shift. This monitoring is done through the use of a GPS sensor and a calculation to determine if they are considered onsite, *Sensing.*

**Issues:** For this functionality to work correctly there needs to be continuous monitoring of the worker’s location. GPS sensing works best when outdoors with a direct link to the overhead satellites, *Hardware Constraints.* Not all shifts will take place outdoors and should be fact-

\(^2\)http://developer.android.com/design/index.html
stored into the design of the app. The other main concern when using
the GPS, is power consumption.

**Tradeoffs:** There are other techniques that can be implemented to de-
determine if a worker is onsite. One approach is Wifi position triangu-
lation. This approach has the benefit of working indoors although it
requires more calculations and multiple WiFi points.

**Solution:** One of the design decisions facing the architects of Prompa
was on the frequency of checking the users location. Rather than con-
front this issue head on the developers opted to provide a user setting
to specify the period of the updates. This allows users to determine
the optimal setting for their device and permits greater accuracy when
it is required. Checking if an employee is at the shift location is an
important feature that needs to be running all of the time – it cannot
afford to be suspended as staff may be marked as onsite when they have
been working for a number of hours. The appropriate decision for this
problem cannot be made purely by knowing the requirements for this
functionality. More information is needed about the mobile operating
system’s *Threading Model* to make the appropriate decision.

Creating a background service also raises the concern about how to
communicate with the main thread. Developers have the option of mak-
ing an API call from within the background service or delegating this
responsibility to the data access layer of an app. The developers of
Promopa opted to make an API call from within the background ser-
vice. This option avoided the need to pass information between the
background service and another thread to make the call simplifying
the complexity of the code improving its maintainability. The tradeoff
of this option is that now the call backs for a successful and failed API
call need to be handled in the background service as well as in the data
access layer which also impacts upon the maintainability of the system.

*Example 6: Ecosystem and Frameworks, Lifecycle and Threading Model*

**Description:** Prompa is an app that does not integrate with any analyt-
ics frameworks. In this example we will use the concern of *Ecosystem*
and Frameworks from our model as a means to analyse the feasibility and implications of integrating an analytics framework into Prompa.

Issues: Analytics frameworks gather data from an app as it is being used then send this information to a server where it can be processed and analysed. For an app like Prompa that is part of a larger framework decisions need to be made on what to monitor and how frequently. There are also architectural changes that may need to be made to integrate an analytics framework depending on how much content needs to be tracked.

Tradeoffs: Capturing analytics data in a only a few places decreases the magnitude of the changes to the software architecture and to performance. The downside of this approach is that data captured may not be sufficient for actionable outcomes or may be the wrong information required. On the other hand capturing too much data can decrease performance.

Solution: Before integrating an analytics framework into Prompa we recommend consider the purpose for analysing the output. Determine the reason for adding analytics will inform the decision for how wide spread the integration will be and therefore, how significant the changes to the architecture are likely to be. Knowing where to track information requires understanding the implications of tracking in the appropriate application states, Lifecycle. Understanding the performance implications of a framework will also help make informed decisions on whether or not tracking can be run in a background service (Threading Model).

Example 7: Ecosystem and Frameworks (Platform Trends)

Description: At the time of development of Prompa there were three major UI patterns for grouping screens together for navigation. They were grouping links to a screen together in a drop down, displaying a tab for each screen or placing the links in a UI pane that slides in from the left hand side of a device. There were also two other ways to navigate to a screen, by clicking on a button or by selecting an action displayed at the top of the screen. The choice of navigation for a screen impacts
on how that screen is implemented which influences the architecture of the app.

**Issues:** Choosing the type of navigation for a screen impacts upon the usability of an app. Deep navigation hierarchies between screens decreases usability as access to functionality that users need is not readily available. Displaying a means to navigate to every screen on each screen of the app is not desirable as this takes up real estate which is at a premium on a mobile device. Another consideration on the choice of navigation system is the current UI patterns and trends.

**Tradeoffs:** A balance is required between usability and availability of the navigation of an app. Decisions need to be made around what the user needs access to most often or what is most important to the user. Following trends blindly is not a sure way of building good user experience into an app either.

**Solution:** The decision by the developers of Prompa was to implement the global navigation using tabs with secondary navigation in an action bar. The choice of using tabs was sufficient as there were only three major screens that needed to be displayed to the user as they contained the major functionality of the app. Should more screens need to be added to a tabbar at a later stage the choice of UI pattern may need to be reassessed – as more tabs are added to a tabbar they require the user to scroll horizontally to see each tab negatively impacting upon usability. An interesting side point of this choice would be how the architecture of the app would have changed if it had been designed now with today’s trends.

**Key Findings**

- End users are automatically logged in the second time they open the Prompa app which deals with Hardware Constraints, Threading Model and Lifecycle.

- Performance issues involving constantly checking the devices location were overcome by providing user settings to configure the
frequency of updates which addresses the technical concerns of Sensing, Threading Model and Hardware Constraints.

• The technical domain model can be used to evaluate the benefit of new features into an existing system. In this case the need for integrating analytics framework was analysed and the impact on the system depends on what the requirements are for tracking.

• The Ecosystem and Frameworks influence the UI design as there are UI patterns that are fashionable. In Prompa, at the start of the project tabbars were popular and then the navigation draw became popular. Keeping up with these trends is a core concern of mobile app maintainers.
Appendix L

Generalisation

Included in this Appendix are two technical domain models one for smart homes (Figure L.1) and one for building systems that use artificial intelligence or a prediction engine (Figure L.2).

![Diagram](image)

**Figure L.1**: The start of a technical domain model for a “smart home”.
Figure L.2: The start of a technical domain model for developing "AI Solutions".
References


References


[114] J. Hutchinson, M. Rouncefield, and J. Whittle, “Model-driven engineering practices in industry,” in *Proceedings of the 33rd Inter-


References


References


