Edge Hill University

An Ontology-Driven Approach To Personalised mHealth Application Development

Daniel George Campbell

Department of Computer Science

March 2018

A THESIS SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS OF EDGE HILL UNIVERSITY FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

Declaration

This thesis is submitted to Edge Hill University in support of my application for the degree of Doctor of Philosophy. It has been composed by myself, all copyrighted material appearing in this thesis and all such use, has been clearly acknowledged. This thesis has not been submitted in any previous application for any degree.

Daniel George Campbell

March 2018

In Loving Memory

Blue

2003 - 2016

Acknowledgements

It goes without question that my PhD journey has been an extremely challenging but also rewarding experience. It has not only enabled me to develop as an academic but also grow as a person. My journey would not have been the same without the support from numerous colleagues, friends and family. I therefore would like to take this opportunity to extend my gratitude to everyone involved.

I would like to begin by thanking my supervisors, Professor Ella Pereira, Dr Gary McDowell and Dr Chitra Balakrishna for all their support and guidance they have provided throughout my PhD journey. I would like to especially thank Professor Pereira for her constant positivity, motivation and her desire for me to succeed. You have been truly inspirational throughout my time as a student at the university and I can only wish to one day do the same for others.

I would like to thank my examiners, Professor Nik Bessis and Dr Muhammad Younas for investing their time in their busy schedules to read my work.

To colleagues (past and present) within the Department of Computer Science. It has been an honour to work alongside so many of you. I would like to especially thank: Dr Mark Hall and David Walsh for the inspirational 'just get it done' talks. Dr Chris Beaumont, for providing me with the opportunity to teach & inspire others. Collette Gavan who played a vital role in my development as a tutor, providing advice, guidance and most importantly sweets.

To all my friends, thanks for all the adventures, miss-adventures and the support you have all gave me. I apologise if I was not there when you needed me most, or let you down, or I was the cause of your demise or even kicked you. But I can assure you it is extremely difficult to play games and work on a PhD at the same time. All I can say is 'GG', 'WP' and 'my ultimate is ready'.

I consider myself extremely lucky to have also made some new friends along my journey. To the departments 'wizard' and good friend Dan Kay, you never failed to provide a hardware or software solution that I needed, and I have honestly lost count of how many pints I owe you. To my fellow PhD colleagues Alex Akinbi, Darryl Owens and Peter Matthew who also embarked on their journey at the same time; the office we all shared at the beginning may have been small but the influence you guys had on my journey was huge. Our discussions, ideas and rants will be forever remembered.

To my Mum, Dad and family; for providing me with the foundations to pursue my dreams. I would not be the man that I am today without the sacrifices you have made and for that, I am eternally grateful.

The final thanks and arguably the most important one. To my fiancée Chloé, I know the PhD journey has placed a strain on our relationship, but throughout you have shown nothing but love, support and compassion. From the beginning, you said to me that you believed in me, at the time I did not realise how important those words were. When times were tough, those words echoed in my mind and it was you that provided the light that guided me through when I got lost. Without you, this thesis would not have been possible and for that, I cannot thank you enough. But as this chapter in our life begins to come to an end, I would like to say I love you and look forward to the adventures that lie ahead.

Abstract

Mobile devices when provisioned with intuitive mobile healthcare (mHealth) applications provide a powerful platform that has been recognised to have made a significant impact on healthcare delivery. The popularity of mHealth applications is rapidly expanding amongst consumers and there is a continuous demand to improve the effectiveness of mHealth applications. Personalisation has already been acknowledged by the healthcare industry as a mechanism to improve healthcare delivery, recognising that each consumer is unique. Yet, a typical mHealth application is designed to cater for the needs of large target demographics and are frequently developed without the necessary knowledge and expertise of healthcare providers. As a result, they often fail to meet the consumer's specific healthcare requirements. Since healthcare professionals understand the specific healthcare requirements of a consumer, they are best suited for developing personalised mobile healthcare applications. However, they do not possess the familiarity, skills and knowledge to address the challenges associated with mobile application development.

Therefore, this research addresses the need for a new approach to personalised mHealth application development in the form of an extensible ontology-driven framework that enables healthcare professionals to create personalised mHealth applications for healthcare consumers. This research explored personalisation & the challenges of personalised mobile application development, existing approaches and related works. Followed by a detailed investigation into the various health-related functions available in mHealth applications designed for healthcare consumers, that led to the creation of the mHealth Application

Function Taxonomy. The next phase presents the theoretical design and development considerations of the Personalised Mobile Application Development (PMAD) ontology. The PMAD ontology encapsulates key knowledge associated with the development of personalised mHealth applications, that can be operationalised to compensate for the missing domain expertise during the personalised mHealth application development process. The final and contribution of this research describes and defines the approach and components of the Personalised Mobile Application Development ontology-driven framework that addresses the limitations of existing end-user programming solutions and enables healthcare professionals to create personalised mHealth applications for healthcare consumers.

Keywords: personalisation, mobile, healthcare, mHealth, taxonomy, ontology, framework

List of Publications

1. D. Campbell, E. G. Pereira, and G. McDowell, “Ontology Driven Framework for Personal mHealth Application Development,” *2014 Eighth Int. Conf. Next Gener. Mob. Apps, Serv. Technol.*, pp. 320–325, 2014.

2. D. Campbell and E. Pereira, "A novel ontology-based approach to personalised mHealth application development," 2016 SAI Computing Conference (SAI), London, 2016, pp. 985-989.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The first chapter in this thesis aims to clearly introduce: the motivation behind this research, the aim & objectives, scope, the intended contributions. It concludes by outlining the topics and themes of the remaining chapters.

1.1 Motivation

Mobile devices when provisioned with intuitive mobile healthcare (mHealth) applications provide a powerful platform that has been recognised to have made a significant impact on healthcare delivery [1, 2]. From a consumer's perspective mHealth applications provide a variety of functionality that provides an effective mechanism for promoting of long-term well-being and independence [3, 4]. As the popularity of mHealth applications is rapidly expanding amongst consumers, there is a continuous demand to improve the effectiveness of mHealth applications [5].

Personalisation has already been acknowledged by the healthcare industry as a mechanism to improve healthcare delivery, recognising that each consumer is unique [6]. Healthcare

providers possess a set of domain expertise to understand the unique healthcare requirements of the consumer and can optimise treatment, drugs and resources to deliver targeted personalised healthcare [7, 8]. Studies [9, 10] indicate that consumers respond more favourable when treatment is tailored to their personal needs and providing high-quality target care requires the expertise of a healthcare provider [11, 12].

Yet, a typical mHealth application is designed to cater for the needs of large target demographics and are often developed without the necessary knowledge and expertise of healthcare providers [13, 14]. As a result, they often fail to meet the consumer's specific healthcare requirements [13, 15]. Lee *et al.* highlights the need for medical providers to develop personalised mHealth applications, as they understand the consumer's specific healthcare requirements [13].

However, healthcare providers do not possess the familiarity, skills and knowledge to address the challenges associated with mobile application development [16]. Existing mobile application development services, platforms and frameworks [17–19, 13] that are designed to enable a layperson to develop mobile applications, would not be feasible for personalised mHealth application development; as they are restricted to basic functionality and lack the necessary capabilities to support a diverse range of healthcare scenarios. Moreover, other approaches to personalised mHeatlh applications [20–22] utilise ontologies to model consumer characteristics that drive personalised functionality within a mHealth application. Although they demonstrate the capabilities and advantages of personalised mHealth applications, they are restricted to a single healthcare scenario and still require mobile application development domain expertise.

Many of the challenges and issues highlighted throughout the literature have also been experienced first-hand by the author in his previous involvement in projects that are related to the development of mobile applications and software within the healthcare industry. Therefore, the demand for this research is driven by challenges identified in the literature

and previous experience to improve the effectiveness of mHealth applications for healthcare consumers. This goal is to develop a new approach that enables healthcare providers to create personalised mHealth applications, that is extensible and addresses the current challenges and limitations with existing approaches to mobile application development and personalisation.

1.2 Research Aim and Objectives

The aim of this research is to develop an extensible ontology-driven framework that enables healthcare professionals to produce personalised mHealth applications for a healthcare consumer. Fulfilling the aim requires the completion of the following four objectives.

- **Objective (a) - Understand the Technological Challenges and Issues of personalised mHealth Application Development**

  Mobile application development is a complex process and faces a vast array of challenges brought about by a range of factors including the mobile device, mobile technologies and users. The goal of this objective is to identify limitations, restrictions of existing approaches to related specifically personalisation and mobile application development to identify a series of requirements for the framework.

- **Objective (b) - Analyse mHealth Application Functions**

  The focus of this objective is to understand the various types of health-related functions that are available within mHealth applications that are intended to be used by healthcare consumers. The goal is to produce a taxonomy that categorises these functions into distinct categories and investigate their feasibility within the framework.

- **Objective (c) - Establish A Suitable Ontology Model**

  Knowledge acquired from previous objectives will form a solid understanding of a range of areas that include mobile computing, mHealth applications, personalisation and application development. The ontology model must be capable of being extended and encapsulating necessary and sufficient knowledge, so it can be utilised within the framework.

- **Objective (d) - Framework Design and Evaluation**

  Design a suitable framework that enables healthcare professional to build personalised mHealth applications on demand for health care consumers, without the intervention from mobile applications developers. The framework must also address the requirements identified from the completion of objective (a). A prototype that simulates critical components of the framework will be developed and used to test and evaluate the technical feasibility of the framework.

1.3 Research Scope

The focus of this research is surrounding the personalisation of mobile healthcare applications designed to be used by healthcare consumers. It is not the intention of the aim of this research to create a fully operational system, but rather a theoretical, modular, and high-level framework which represents and defines the main architectural components and services of such a system. For this reason, consultation with healthcare professionals would not suitable throughout during its development. However, to demonstrate the feasibility of the framework core components were implemented that integrated the ontology to simulate the key functionality of the framework.

1.4 Original Contributions to Knowledge

This thesis intends to deliver an original contribution to knowledge in three forms. Each contribution is summarised below:

- **A Taxonomy** - *A method for classifying healthcare related functions found within mHealth applications based upon the services they provide to the healthcare consumer.*
- **An Ontology** - *An extendible ontology model that encapsulates key knowledge associated with the personalised mobile healthcare application development process so it can be made operational via the framework, compensating for the missing mobile application development domain expertise.*
- **A Framework** - *A theoretical, multi-layered architecture for an on-demand service that enables healthcare professionals to create personalised mobile healthcare applications for healthcare consumers.*

1.5 Thesis Structure

Presented below are the structure and a description of the remaining seven chapters within this thesis.

- **Chapter 2: Background and Literature Review**

  Introduces relevant concepts, challenges and ideas that are considered to be influential and significant to the development of this research. It has been divided into two sections. It begins by introducing the area of mobile healthcare, it's benefits to the industry, healthcare delivery and the objective of the next generation of mobile healthcare. Followed by an in-depth discussion surrounding personalisation from

the perspective of healthcare and mobile application development and the challenges and issues of related work.

- **Chapter 3: Research & Development Methodologies**

  Discusses and justifies both the research and development methodologies that were adopted throughout this research. It will describe in detail the techniques used to explore, analyse, test and evaluate the products of this research.

- **Chapter 4: mHealth Application Function Taxonomy**

  The objective of this chapter is to gain an understanding of the health-related functions available within mHealth applications that are designed for healthcare consumer. It discusses in detail the considerations associated with design, development and evaluation of the mHealth Application Function Taxonomy (mHAFT).

- **Chapter 5: PMAD Ontology**

  The objective of this chapter is to present the theoretical, design and development considerations for the Personalised Mobile Application Development (PMAD) Ontology. The chapter is composed of two parts. The first part presents the theoretical considerations exploring key areas important to the development of an ontology. The second part of this chapter provides a detailed insight into the design and development process of the PMAD Ontology.

- **Chapter 6: PMAD Framework**

  The motivation behind this chapter is to present and discuss the design considerations of an ontology-driven framework. It is constructed from three parts. The first defines the purpose, scope and design criteria of the PMAD Framework. The second aspect discusses the creation of a conceptual model and explores various factors that influenced the design of the PMAD Framework, and the final part presents an overview of the PMAD Framework and discusses each component in detail.

- **Chapter 7: Evaluation**

  This chapter presents a three-stage evaluation process of both the ontology and framework. The first stage utilises the OntOlogy Pitfall Scanner to identify potential commonly found pitfalls within ontologies. The second stage evaluates the 'consistency' of the knowledge encoded within the ontology. The final phase evaluates the competence of both the PMAD Ontologyand PMAD Frameworkin fulfilling the overall aim of this research.

- **Chapter 8: Conclusion**

  The final chapter provides a summary of the research and how the work completed fulfils the aim and objectives discussed earlier, discussing the achievements, limitations and recommendations for future work.

Chapter 2

Background and Literature Review

The focus of this chapter is to introduce relevant concepts, challenges and ideas that are considered to be influential and significant to the development of this research. It has been divided into three sections. It begins by introducing the area of mobile healthcare it's benefits to the industry, healthcare delivery and the objective of the next generation of mobile healthcare. The following section introduces personalisation from the perspective of healthcare discussing the advantages, process and characteristics of personalised healthcare. This is followed by an in-depth discussion surrounding the challenges of developing mobile application, motivation for personalisation, existing approaches and related works. Finally, this chapter concludes by discussing the strengths and limitations of approaches to personalisation and the process of developing personalised mHealth applications.

2.1 Mobile Healthcare

Mobile health care or *mHealth*, is defined as the provision of health related services or activities via the utilisation of: information & communication systems, computing & internet and wearable & sensor technologies [3, 23]. mHealth applications play a vital role

within the healthcare industry and are designed to be utilised by both healthcare providers and consumers.

mHealth applications provide a unique opportunity to significantly improve the quality of consumer-oriented care, where patients are directly involved in the care process and are already deployed in a diverse range of healthcare scenarios such as; assisting patients with chronic diseases, provide warnings and vital information about an illness or for diagnosing a patient [24]. From a consumers preservative, mHealth applications are designed to transfer the expertise of the care provider to the consumer, without the constraints of physical boundaries [4, 5].

At the consumer's fingertips, a mobile application enables unique access to a diverse range of tools, resources and utilities. In addition to the on-board capabilities of the mobile device, a mobile application can make efficient use of other technologies such as wearable computing devices and sensors to gather data, improve accuracy and extend its capabilities. Provisioned correctly they provide for an efficient mechanism for promoting of long-term well-being and independence [25, 26]. Making it an ideal flexible platform for pervasive and ubiquitous healthcare delivery [27].

As the popularity of mHealth applications is rapidly expanding amongst consumers, there is a continuous demand to improve the effectiveness of mHealth applications [28]. Mobile technologies have been recognised to enable healthcare services to scale while reducing the strain and extending the current capacity and accessibility [4]. However, developments in technologies such as cloud computing, smartphones and 4th and 5th generation mobile networks are laying the foundations for the next generation of mHealth applications to provide targeted, personalised healthcare services [25].

2.2 Personalisation

The Oxford English dictionary defines the term personalise as 'Design or produce (something) to meet someone's individual requirements' [29]. However, how personalisation is achieved in different domains differs. This section explores how personalisation is achieved within the healthcare and mobile application domains and is divided into three parts. Section 2.2.1 explores the notion of personalised healthcare, its components and process. Section 2.2.2 focuses on personalisation of mobile applications analyses, the motivation for personalisation, approaches to achieve personalisation and presents existing and related work. Section 2.3 examines the strengths and limitations of existing approaches to personalisation. Followed by a review of the process of personalised development and discussion surrounding the requirements a framework. The final section provided a summary of the key issues raised within this chapter.

2.2.1 Healthcare

The notion of healthcare is subjective to the individual characteristics that make a person unique, what is right for one person may not be right for another [30]. The primary stakeholder in the healthcare delivery process is the consumer [31]. Recognising that each consumer is unique, the healthcare providers utilises personalisation as a mechanism to improve health care delivery. Personalised healthcare, is the product of long-term research and observation of how patients with the same diagnosis react to the treatment they receive. Patterns and trends discovered in evidence-based research and patient data have allowed healthcare providers to optimise; treatment, drugs and resources to the unique personal requirements of the consumer, enabling consumers to take more of a proactive approach to the care they receive [6–8].

Throughout the literature, there have been multiple studies surrounding the effectiveness of personalised healthcare. Studies such as [6, 9, 10, 32–34] have indicated that personalised healthcare provides the consumer with more efficient targeted care, can further improve the quality of life and overall consumers react more favourably. For example, Nicole *et al.* studied the effectiveness of personalised interventions against universal interventions in tobacco cessation. Those who received a personalised intervention were assessed, and their care was tailored based upon the following characteristics: gender, age, sense of humour, the level of tobacco use, readiness to quit and illness. Overall the study concluded that the cessation rates were significantly higher for individuals who received personalised care compared to those who received universal care [32].

2.2.1.1 Process and Components of Personalised Healthcare

Providing high quality, effective healthcare requires the expertise of a healthcare provider [12]. Typically, in a healthcare scenario there are two key stakeholders a healthcare consumer and a healthcare professional. According to Shieh *et al.*, the practice of healthcare can be modelled as a three stage process, resulting in a treatment plan for the consumer [11]. As can been seen in Figure 2.1, the model consists of:

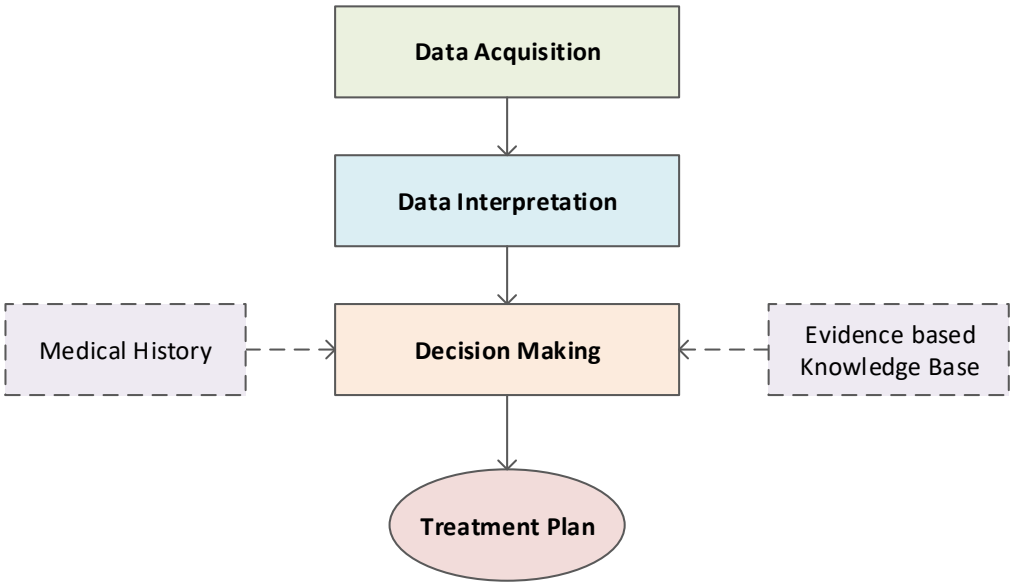


Figure 2.1 Three stage medical practice model [11]

1. **Data Acquisition:** A healthcare provider will the begin to evaluate the consumer's current health status and acquire the data via a variety of methods such as diagnostic tests, medical assessment, medical history and physical examination.

2. **Interpretation:** The healthcare provider will analyse and validate the data to identify disease, illness or condition that is impacting the consumers health. If required further diagnostic tests are performed.

3. **Decision Making:** The decision making process is influenced by three factors: Data and results from the initial two phases, consumers medical history and knowledge derived from an evidence based knowledge based are used to formulate the necessary treatment for the consumer.

Although the medical practice model identifies the universal phases of medical practice, it does not show how healthcare is personalised to the consumer. Synderman and Drake identify six components of personalised healthcare [6]. The first two components "Evaluation

of the patient's current health status" and "Assessment and quantification of their health risk" are already represented by the first two phases in the Shieh *et al.* medial practice model. However, "Enhancement of the consumers engagement' is a critical component of personalised healthcare. Previous studies have documented that placing the consumer at the centre of their care produces the best outcomes. Utilising an inclusive approach to healthcare enables consumers to be directly involved in the process of their care alongside healthcare providers. This interaction from the consumer is vital as it influences the remaining components:

- **Development of a therapeutic plan:** After data is analysed and validate during the interpretation phase a healthcare provider will have an insight into the factors and risks that are impacting a health of the consumer. The healthcare provider will discuss both the results of the assessment along with the therapeutic needs with the consumer. The outcome is to establish shared goals and produce a therapeutic plan that meets the therapeutic needs of the consumer.

- **Tracking metrics:** Monitoring the consumer progress requires establishing metrics via bio-markers and clinically approved tracking tools.

- **Creation of a personalised health plan:** Documents the goals to be achieved, their timing, and the metrics to track progress. The personalised health plan, if necessary includes formal follow-up in which the health provider can monitor the patient's progress through relevant clinical metrics such as via wearable health technology, mobile applications, or additional visits if necessary.

Therefore, Figure 2.2, presents the alignment of Shieh *et al.* medical practice model with the personalised healthcare components identified by Synderman and Drake to represent a medical practice model for personalised healthcare.

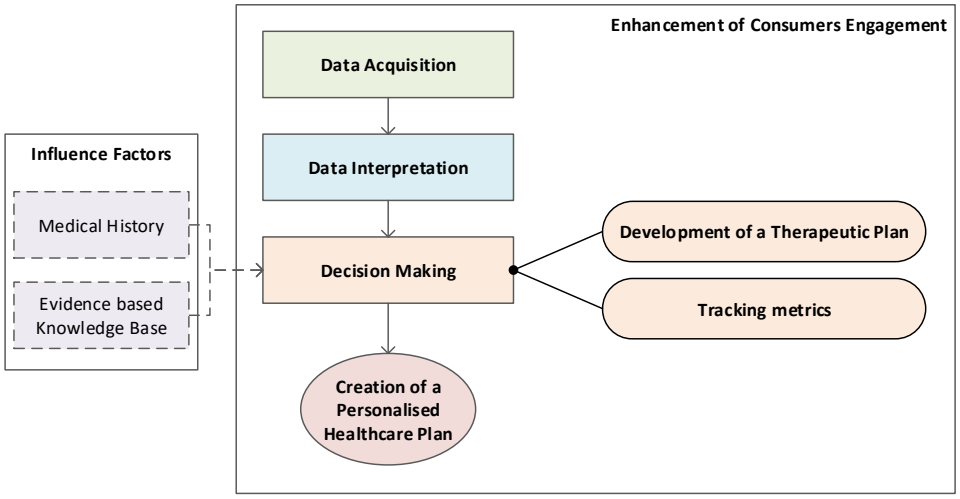


Figure 2.2 Medical practice model for personalised healthcare

2.2.2 Mobile Health Applications

Personalisation has been recognised throughout the mobile application development industry to increase the quality of mobile services and applications [35]. Throughout the literature, personalisation has many interpretations and has been approached differently to achieve varying objectives [22, 35–38]. This section discusses the key challenges that are associated with mobile application development and explores how the motivation behind personalisation and how personalisation is achieved in mobile applications.

2.2.2.1 Challenges of Personalised Mobile Application Development

Mobile devices such as a smartphone and tablet have created tremendous opportunities for the healthcare industry and have become a pervasive component of everyday life. The last decade saw the smartphone become the most successful electronic consumer product [26, 39] and statistics indicate that ownership is rising year on year [40] and they are being used throughout the entire adult (16+) age spectrum [41]. Not only do mobile devices

provide the most personal computing experience, but have now achieved such a pervasive presence in society users are becoming more reliant on them for their personal computing needs [40].

However mobile application development is a complex process. Sommerville describes the software engineering process as; theories, methods and tools which are needed to develop software [42]. A software process refers to set of interrelated or interacting activities that result in the creation of a software product and requires the collaboration of technical and managerial processes to produce a mobile application [43]. Sommerville identifies four fundamental phases of the software engineering process: Specification, Design & Development, Testing and Maintenance [44]. Arguably the most critical aspect of the software engineering process is the specification activity. Prior to developing any mobile application, it is vital to understand what exactly it is supposed to do and how it will benefit the end-user. A typical mobile application is designed to target the needs of a large demographic [45]. However, in context of this research if a mobile application is identified as a suitable clinical metric within a personalised healthcare plan, it is integral that a application is designed specifically to the personal requirements of the healthcare consumer, this also includes taking into consideration the device they own; not only does this reduce the investment in hardware from the healthcare industry [46], it also will dictate key decisions during the development process.

Developing high quality personalised mHealth applications it is imperative to understand key characteristics and challenges associated with developing such mobile applications [47]. Throughout the literature researchers have identified that the challenges of mobile application development can be traced back to three categories: Hardware, Software and Communication [47–49]. The subsequent sections discuss the key challenges of mobile application development from the perspective of personalised mHealth application development.

2.2.2.1.1 Hardware

The hardware is one of the factors that dictates the scope of an application. Although mobile applications are relatively small in size, the hardware in modern mobile devices provides the capabilities to mobile applications to perform various tasks. Nevertheless, it is inherently limited by its composition of hardware and portable form factor [48].

Battery

The demands for fast and responsive mobile applications from end-users while addressing the constraints associated with hardware is a major challenge for mobile application developers [49]. Common to all mobile devices and arguably the biggest limitation, is the battery. Developers of mobile applications believe efficient code, suitable architectures, intelligent software design can significantly minimise power consumption by reducing computation and perform tasks periodically or only when necessary to satisfy the users demands [47, 49].

Storage Capacity

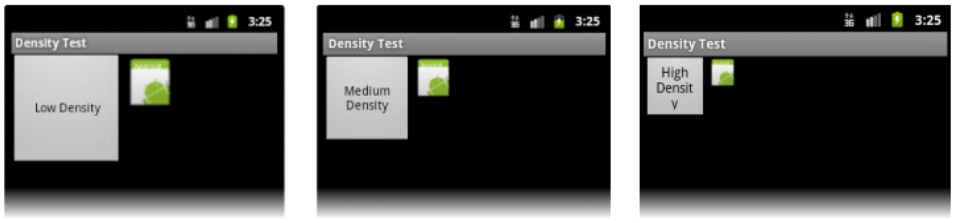
Liu *et al.* states the healthcare industry is one of the leading areas for utilising mobile technologies to gather data [50]. Although, there has been a steady growth in the storage capacity available on a mobile device to coincide with the use of high definition multimedia, the volume of applications and their associated data. As Joorabchi *et al.* states dealing with data can be problematic for developers as local storage capacity is limited, using a network connection to synchronise data or the use of offline caching for data intensive application is challenging [51]. Therefore, developers have to be conscious of the volume of data produce by the mobile application and implement suitable data management techniques.

Accessibility, Input and Output

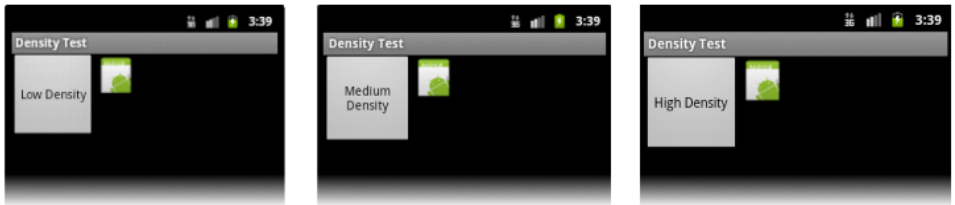
Mobile applications have a small number of input and output mechanisms such as the display, microphone and speaker. The display is typically the primary interface for users to interact with and for the system to indicate the effect of the users' interaction, making

screen real estate a premium [47]. The design of the user interface is an essential activity in the software engineering process [44]. A user interface should be simple and easy to use. It is equally important that a personalised mHealth application is designed to be accessible. The World Wide Web Consortium (W3C) has produced a working draft that describes four accessibility principles and guidelines that can be applied to mobile web content, native, web and hybrid applications [52]. The four accessibility principles are: Perceivable, Operable, Understandable and Robust.

Perceivable refers to how information is rendered on screen. Although some displays provide high resolutions, they are typically smaller compared to their desktop counterparts. There are three attributes of a display: physical dimensions, resolution and pixel density [53]. As demonstrated in Figure 2.3a, each of these attributes affect the appearance of user interface and controls in various displays. However, if an application is density independent Figure 2.3b, the UI (user interface) elements appear to take up the same proportional area of the display regardless of the pixel density of the display. Moreover, minimising the volume of information and inclusion of zoom, magnification and contrast control all improve the accessibility of information for the end-user.



(a) Screen density: How UI controls are affect by displays with different screen densities (*low*, *medium*, *high*)



(b) Screen density independence: Low, medium and high screens

Figure 2.3 Screen density

The operable principle considers how the end-user interacts with a mobile application. The placement, size and spacing of the UI controls, as well as gestures all, can impact the user's ability to interact with an application. The third principle understandable defines various good practices when designing a user interface. Although each mobile platform follows a specific set of HCI (Human-computer Interaction) principles to provide a consistent UI experience across applications coexisting on the same device [51]. Consistent layout, positioning, grouping, an indication of actionable items and instructions all improve the user's experience making it easier to navigate and interact with new applications. The robust principle focuses on data entry and supporting the characteristics of the platform. As briefly mentioned a mobile device has limited input and output capabilities, therefore data entry is often difficult. Developers can manipulate the virtual keyboard to the type of data entry required, for example, numerical values only for entering a telephone number. Developers can also utilise various UI controls such as lists, radio buttons or check boxes to reduce the volume of text entry. Therefore, it is highly recommended that mobile applications are optimised for the display and adhere to the accessibility guidelines and principles defined by WC3 to provide the end-user with a consistent and accessible user experience.

Hardware Fragmentation

The device an end-user owns runs a particular operating system that is a component of a mobile platform. In a utopian world, mobile applications that are developed to run a specific mobile platform should run on all devices that run the platforms operating system [47]. However, hardware fragmentation prevents this from occurring. Hardware fragmentation is the result of several factors; variations in hardware configurations, proprietary drivers and software [51]. These variations can cause applications to function unexpectedly even though they may work fine on other devices. This is a major concern for developers as it can have negative implications for the end users but also makes it difficult to test the application [51].

2.2.2.1.2 Software

Software characteristics refers to the interaction, development and security of a mobile application. This includes: mobile platform, application architectural style, integration with data sources, security, credibility and maintenance.

Mobile Platform

At a fundamental level, a mobile platform is a term referring to the operational ecosystem, consisting of various services, tools, products and operating systems. There are numerous mobile platforms available. In the United Kingdom, there are three major competitors: Android, iOS and Windows phone that form approximately 90 percent of the market share [54]. Each platform has its own requirements for developing native applications. Manufactures produce software development kits (SDK's) that contain all the necessary tools and libraries (API's) to develop mobile applications for that specific platform. Table 2.1 presents a comparison between the Android and iOS mobile platforms.

As can be seen in Table 2.1, each platform requires different Integrated Development Environments (IDE) and programming languages to develop mobile applications. As a mobile platform matures, changes and improvements are made to the operating system. Although this drives innovation; rather than unification occurring, fragmentation is introduced, as changes to the newer operating system, results in modifications to the libraries, tools and resources within the SDK. This is a major challenge for developers [51]. However, unlike developing a traditional application that would require the supporting multiple devices and/or multiple platforms, a personalised mHealth application within the context of this research would require an application designed to operate on the user's device, thus reducing the effect of fragment within the mobile platform.

Table 2.1 Mobile platform comparison

	Mobile Platform	
	Android	***iOS***
Manufacturer	Google Inc	Apple
Operating Systems Examples	• Lollipop • Honeycomb • Ice Cream Sandwich	• iOS 4 • iOS 3 • iOS 5
Programming Language(s)	• Java • C • C++	• C • C++ • Objective-C • Swift
Integrated Development Environment	Android Studio	Xcode
SDK	Android SDK	IOS SDK
App Marketplace	Google Play Store	App Store

Application Architectural Style

When developing a mobile application, developers can choose between three types of application architectural styles: Native, Web and Hybrid. As highlighted in Figure 2.4 each approach has its advantages and pitfalls:

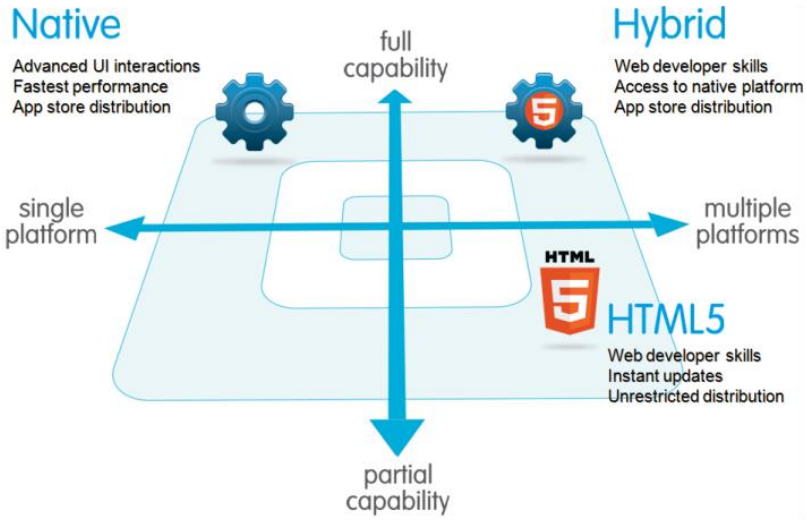


Figure 2.4 Overview of application architectural styles [55]

- **Native Applications:** are specifically developed using the programming language and development tools designed to operate specifically on a particular mobile platform, such as Android, iOS or Windows 10 mobile. Developers have access to the native application programming interfaces (APIs) allowing applications to fully utilise the device's hardware and software capabilities, to provide the richest and most compelling user experience [56, 57]. Native applications also can run offline, since the application is installed on the device and data transfers can synchronise with back-end services when a internet connection is available. Native applications are exclusive to a single platform and are distributed via the platforms app store. Any changes that are made to the mobile application, such as a patch or new functions requires re-validation by the mobile platform to ensure it meets their strict standards and regulations. Although native applications excel with regards to performance and

functionality, they are associated with longer development times, typically cost more to develop and are required to adhere to mobile platform regulations [58].

- **Web Applications:** have the greatest reach, require less financial investment, but at the expense of performance and user experience. They are built using web frameworks and technologies such as HTML5, Java script and CSS. Web applications are platform independent (with minor modifications in some cases). This means they possess the ability to run on a device regardless of the mobile platform. This is achieved since the application is located on a central server and is accessed via the devices web browser. However, as Gokhale *et al.* [59] indicates this approach has two distinct limitations: Web Applications cannot access the native APIs and require an internet connection in order to function; thus restricting their capabilities and performance.

- **Hybrid Applications:** as the name suggests, combine elements of both native and web applications. Typically the user interface is built using web technologies and contained within a native wrapper, such as Adobe Phone Gap. Allowing the application to access the devices native hardware and software [55]. The amalgamation of both the native and web technologies allows developers to reuse portions of the code for different mobile platforms, reducing development time, costs and supporting cross platform development.

The delivery of healthcare should not be restricted by the device a consumer or to a particular mobile platform. Selecting a suitable application architectural style requires developers and stakeholders have to carefully consider: the cost associated with the development, differences in the underpinning architecture, advantages and drawbacks constructed as a consequence of the required functionality, intended use and scope of the application [56, 57]. Incorrectly choosing the wrong approach can limit the applications capabilities and/or impact end-user satisfaction.

Integration with Data Sources

Mobile applications interact with various sources to send and receive data. Data is a valuable asset to consumers and healthcare providers. Interoperability is another concern for developers [16]. Many healthcare services are heavily invested in their own IT systems. For data produced by a personalised mHealth application to be effective and accessible, it is vital that they are compatible and capable of communicating with back-end services of existing systems [60, 61].

Security

Healthcare is one of the leading areas for utilising mobile technologies to gather data [50]. Due to the nature of personalised and the various environments a mobile device can operate in, security, privacy and safeguarding of personal data are prominent concerns for the healthcare industry [16, 61]. Plachkinova *et al.* emphasises that many existing mHealth applications currently available have flaws that could prove detrimental to healthcare consumers, providers and services [62]. Although mobile platforms may offer varying degrees of support for safeguarding data, it is the responsibility of the developer for protecting data. Therefore, it is paramount that security, privacy and data management mechanisms such as encryption, automatic backup and remote wipe are present in all mHealth applications and the services they interact with [63].

Credibility

The rapid growth, developments and introduction of new mobile technologies make the development of mobile applications complex. Mobile Application Developers posses the necessary skills, knowledge and familiarity of a mobile application enabling them to address and or manage challenges throughout development [43]. However, many mobile applications that are readily available to download, are designed and developed without the necessary knowledge, familiarity or experience of healthcare, thus leading to questions regarding the accuracy and credibility of the content, data and functionality provided [13, 46]. When developing mobile applications for healthcare, a developer should be conversant with aspects of healthcare they are developing the application for [64].

Maintenance

Due to the nature of mobile technology, there is a consistent introduction of new techniques and trends driving the industry forward. Alongside these new technologies and trends, requirements change, bug fixes are required as the mobile application matures. As a result, maintaining a mobile application requires developers to regularly update their skills and knowledge to ensure that they are capable of producing high quality up-to-date mobile applications [44].

2.2.2.1.3 Communication

Users of mobile devices have the freedom to navigate throughout a physical space and are reliant on wireless communication technologies for transferring and receiving information. As mobile device are operate in heterogeneous networked environments, developers must consider factors associated with wireless networks such as variable bandwidth and robustness, routeing and network failure or disconnection [65, 66]. By ignoring these factors when developing mobile applications can impede functionality and user satisfac-

tion. Therefore, requiring developers to design and implement suitable redundancies and solutions into mobile applications.

2.2.3 Motivation for Personalisation, Approaches and Related Works

Within mobile applications, personalisation can exist in various forms and can be achieved through several different approaches. The remainder of this section discusses the motivation for personalisation within the context of a user, the approaches to personalising mobile applications and concludes by presenting a series of related works.

2.2.3.1 Motivation For Personalisation

So why personalise a mobile application? Bolm's work on personalisation presents a user centric taxonomy that focuses on the motivations for personalisation and defines personalisation as 'a process that changes the functionality, interface, information content, or distinctiveness of a system to increase its personal relevance to an individual' [36]. Blom identifies two distinct categories of motivations for personalisation as depicted in Figure 2.5.

Work related motivations consist of: 'enable access to information content', 'accommodate work goals' and 'accommodate individual differences'. This group of motivations is typically centred around the mobile applications functionality. In the context of personalised mobile healthcare, these three categories may provide a consumer with: access to information regarding a specific illness, calculate insulin dosage and provide accessibility features to accommodating for a visual impairment, respectively. Whereas Socially related motivations include 'elicit emotional responses' and 'maintaining and expressing identity of a user'. Socially related motivations typically relate to an individual's character.

For example, Blom associates 'elicit emotional responses' with choosing a particular ring tone as it may symbolise happiness from the perspective of the user. Maintaining and expressing identity is described as 'a key part to social identity is the group or groups one associates with' [36]. From a mobile applications perspective this may include a user displaying a badge in their profile to represent support for a charity. Although, this work identifies motives for personalisation from a user centric perspective, it does not identify approaches used to achieve personalisation within a mobile application.

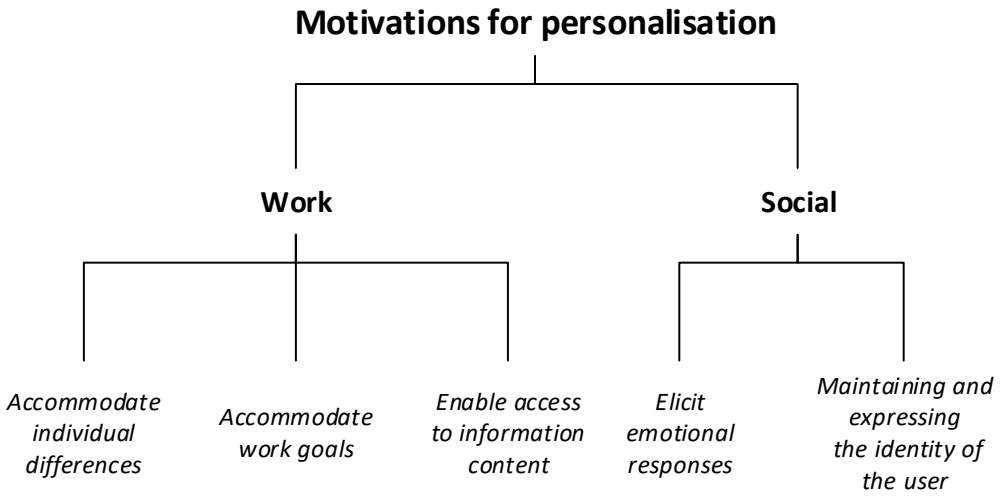


Figure 2.5 Blom's taxonomy of motivations for personalisation

2.2.3.2 Approaches to Personalisation in Mobile Applications

Within mobile applications personalisation has been approached differently. According to Jorstad [38, 67, 68], understanding how personalisation can be supported in the form of a mobile application requires considering composition of the fundamental building blocks of a mobile application. Figure 2.6 shows the fundamental building blocks of a personalised each component is described in below Table 2.3.

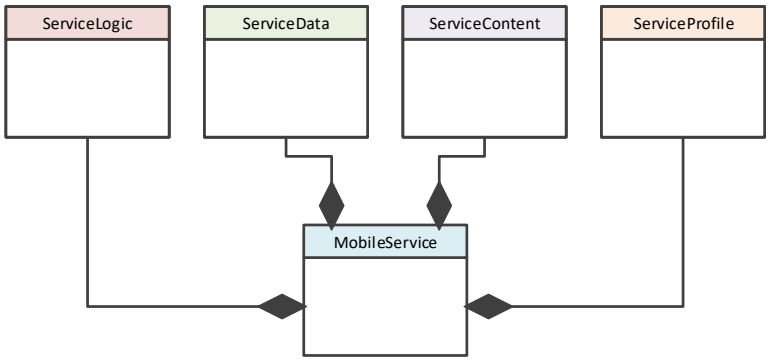


Figure 2.6 Composition of a generic mobile service

Table 2.3 Description of the fundamental building blocks of a generic mobile service

Building Block	Description	Example/s
Service Logic	Essentially code that forms to create the functions within a mobile application	Drug Monitoring, Email, Insulin calculation
Service Data	Data that is used during the execution of the service logic	Variables such as drug
Service Content	Persistent data that must exist from one session to another. It is typically consumed or produced by the user	Documents, Database
Service Profile	User configurable settings or preferences	Layout, Text size, Theme

Each building block provides an opportunity to personalise a mobile application to the specific requirements of the end user. The following works identify various approaches to personalisation. Henricksen and Indulska [37] identifies multiple approaches to personalisation in mobile applications and are as follows: End-user programming, User Profile Modelling & Machine Learning and Preference based. Ho and Bull [35] also identify that location based information can also be used to achieve personalisation. Each of the aforementioned approaches are described below.

- **Location based:** Location Based Approach Personalisation that utilises the users current location data such as GPS to tailor mobile content and services [35, 69, 70].
- **Preference based:** This technique is reliant on the users input to achieve personalisation. Users are presented with various options, rule or files that can be manipulated to influence the behaviour of a mobile application [35, 37].
- **User Profile modelling & Machine learning:** This approach utilises modelling and machine learning techniques to derive the user's requirements from historical data. The user profile model is utilised by the system or application to influence the behaviour of the application, rather than relying on the end-user [22, 37, 70].
- **End-user programming:** A technique where the end-users of a system can construct a mobile application to the desired requirements. This approach is heavily reliant on the end user to specify requirements and create functionality accordingly [14, 37].

2.2.3.3 Related Works

The following works demonstrate how the aforementioned approaches have been utilised to achieve personalisation in mobile applications.

(i) **Flexible Semantic Web Service Management Environment :**

The motivation behind the development of the Flexible Semantic Web Service Management Environment (FLAME2008) was to facilitate and supply better information to the general public during the 2008 Beijing Olympic games [71, 72]. Users would access this service via a mobile application to receive meaningful information (service content) based on the users current 'situation'. FLAME2008 achieves personalisation by combining two of the approaches to enable the system to understand the situation of the user. Location based data such as users current location, time and date are derived from sensors that are built into the mobile device which is then combined with the user preference data (Figure 2.7) inputted by user by to provide the system with various characteristics. Behind the scenes away from the end users, the services and situations supported by FLAME 2008 are described semantically in the form of multiple ontologies, see Figure 2.7. The characteristics that form the users "situation' are then fed to the inference engine, on-demand via the mobile application. The FLAME2008 system will then process the situations characteristics as described in [72], to determine the necessary required information for that particular user, resulting in the personalised services and information pushed to the user's mobile device.

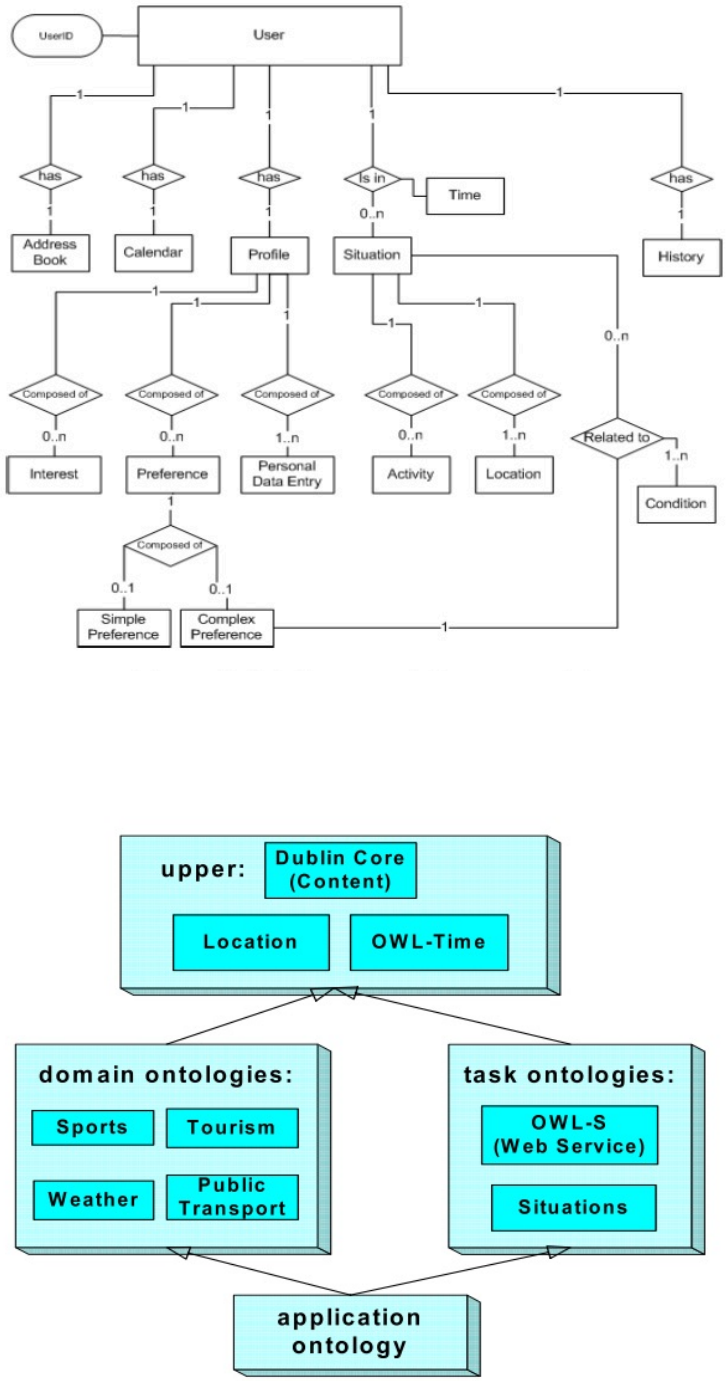


Figure 2.7 FLAME2008:User Model (*top*) and modular ontology architecture (*bottom*) [72]

(ii) **mHealth solutions for cancer survivors'-Engagement in healthy living:**

The motivation behind Myneni *et al.* research was to develop an ontology-driven cancer survivor engagement framework to facilitate rapid development of mobile applications that are targeted, extensible and engaging for adult and young adults (AYA) [20]. A AYA 'Survivorship care plan' is dynamic and documents various transitional characteristics such as: adolescence to adulthood, physiological and psychological growth, self-identity, separation from parents/family, career pursuits, and involvement in intimate relationships. Myneni *et al.* identifies in order to engage AYA's in health management requires the design of a multi-component, modular solution that can be utilised across the care continuum based on the specific personal attributes of the patient.

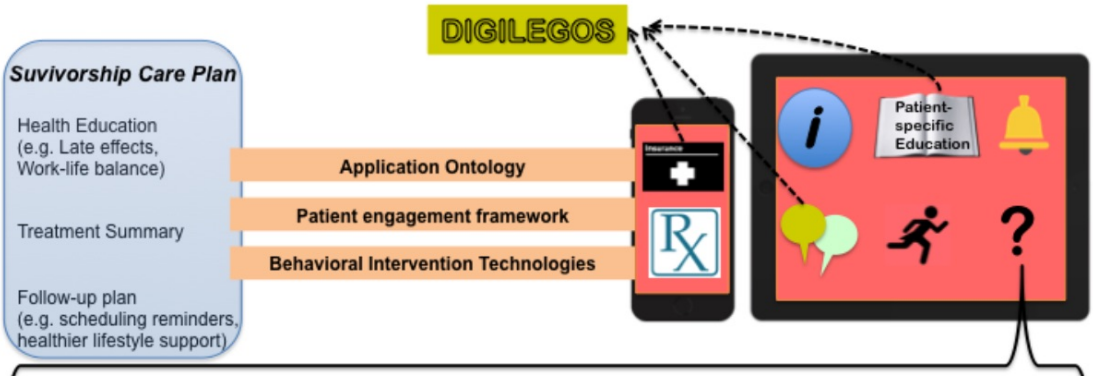


Figure 2.8 Survivor engagement framework [20]

Their proposed framework (Figure 2.8) again utilises an ontology to model the users profile. The Profile Ontology for Cancer Survivors (POCS) was designed to model and store knowledge of the patient's after treatment care plan. It is built on top of the Friend-of-a-Friend ontology [73] and integrates the Patient Engagement Framework [74] and the Behavioural Intervention Technology Model [75] to identify survivor 'Digi-Legos'. The 'Digi-Legos' provide the framework with reusable and customisable blocks that can be arranged to build a personalised application, these are represented in Figure 2.8 as the functionality within the mobile application.

Knowledge contained within the ontology can be used to evoke rule-based machine intelligence and decision-making provided via reasoning capabilities and querying of the POCS ontology to identifying the required 'Digi-Legos' for the user. Currently there are eleven 'Digi-Legos': Insurance information, Health behavioural trackers, Treatment summary, personalised late affects summarization, Follow-up care scheduling, personal profile, targeted health tips, transition assistance, lifestyle tips & care reminders, social hub and question corner.

(iii) **Assisting people with dementia in mobile environments:**

The motivation behind this work is to provide a personalised, context aware assistance services for users with dementia [21]. The mobile application is designed to assist users with dementia through activities of daily life, such as shopping and navigation. Skillen *et al.* adopts both the user profile modelling & machine learning and location based approaches to personalisation. By extending upon the existing user-profile ontology models (User Profile Ontology with Situation-Dependant Preferences Support [76] & General User Model Ontology [77]) Skillen *et al.* address the personalisation limitations within the context of this work. The inclusion and modelling of additional user-characteristics such as personal information, capabilities, interests, preferences enables for the end-user's lifestyle and healthcare requirements to influence the behaviour of the mobile application and achieve the desired level of personalised assistance.

The characteristics of the conceptual user profile model combined with the location based data enables the mobile application to utilise this information to assist people with dementia throughout various daily activities. The architecture of the mobile application is shown in Figure 2.9 and a summary of each layer is provided below, for a detailed description of the architecture see [21].

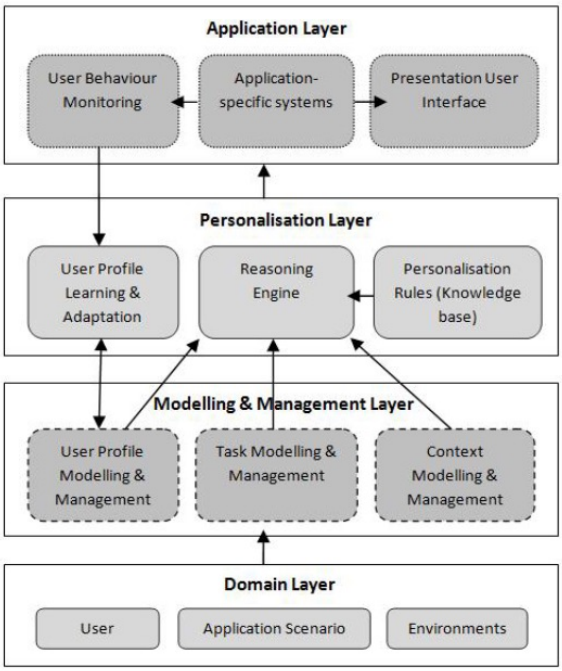


Figure 2.9 Skillen et-al. personalized, context-aware system architecture [21]

- Domain Layer: Contains three core elements: User, Application Scenario and Environments. Combined these provide the application with the necessary data and information required by the application.
- Modelling and Management Layer: Is responsible for modelling and managing the three core elements from the domain layer to model, manage and maintain the user profile ontology.
- Personalisation Layer: Contains a user profile learning and adaptation component, a reasoning engine and a knowledge base consisting of a set of rules, which enable the delivery of the personalised service to the user of the application.
- Application Layer: Focuses on the delivery of the application via the smartphone based user interface, the application- specific systems and the monitoring of user behaviours. Information is fed back from the user , enabling the application to monitor the users actions and behaviour, new information is then

used as input for the learning and adaptation component, where it is added to the user profile ontology for future reference.

(iv) **Travel Assistance: Help-On-Demand services in pervasive environments:** Skillen *et al.* work considers a systematic approach to service personalisation for mobile users in pervasive environments and presents a service-oriented distributed system architecture. Currently this work focuses specifically phase focusing specifically on travel assistance.

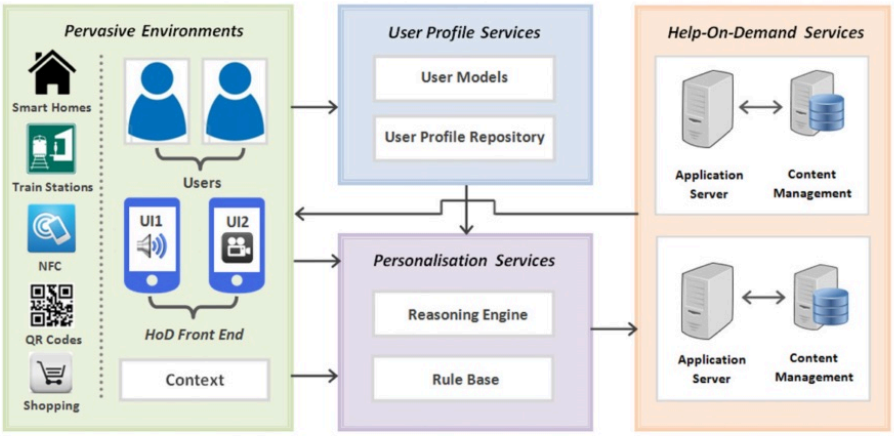


Figure 2.10 Help on Demand: Service-oriented distributed system architecture for service personalisation [22]

By modelling the changing intelligent pervasive environments, the system utilises user: requests and preferences, as well as environmental and application context, to provide personalised services based on domain knowledge, heuristics and rule-based reasoning [22]. The functionality of the personalised services where tested and evaluated using three case studies and deploying the mobile application on three Android mobile devices. Parameters linked to preferences can be modified using the application and are stored on an application server alongside the ontology model. Combined with location based information retrieved from the mobile device the application provides personalised help-on-demand assistance services for the user such as purchasing of train tickets in a foreign country, use personalised media or

provide directions to the nearest hotel in a text based format taking into account users vision and hearing limitations.

(v) **Mobile μ-Healthcare Service System:**

Lee et al. proposed the Mobile μ-Healthcare Service System (MUSS) is a application platform and development environment designed for medical specialists [13]. Following a two stage linear process, as seen in Figure 2.11,the platform enables healthcare specialists creates configurable web based applications for healthcare consumer.

The application designer allows the Healthcare specialist to select the necessary services (functionality) from a predefined selection (sensing, questionnaires, data processing, disease treatment and user feedback). To personalise the service to the healthcare consumer, healthcare specialists are required to modify the service process model as shown in Figure 2.12. Once the application is completed the application designer exports and XML scenario, that is downloaded to a mobile device. The scenario is used to configure the 'scenario player' (mobile application) to provide the personalised services to the consumer.

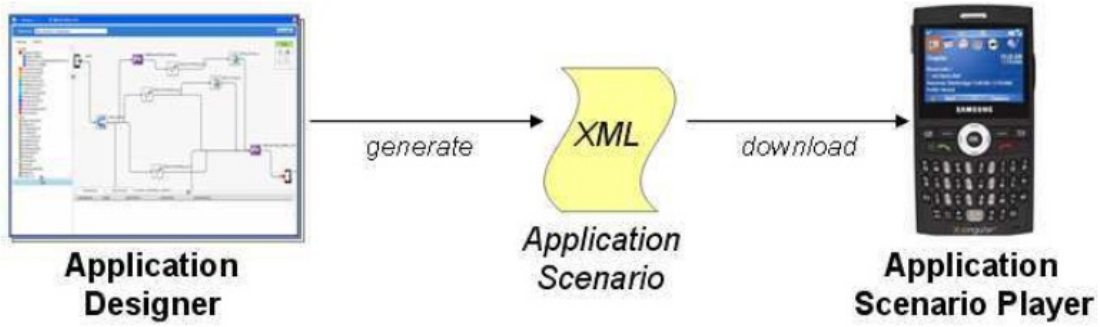


Figure 2.11 Mobile μ-Healthcare Service System: Application execution model [13]

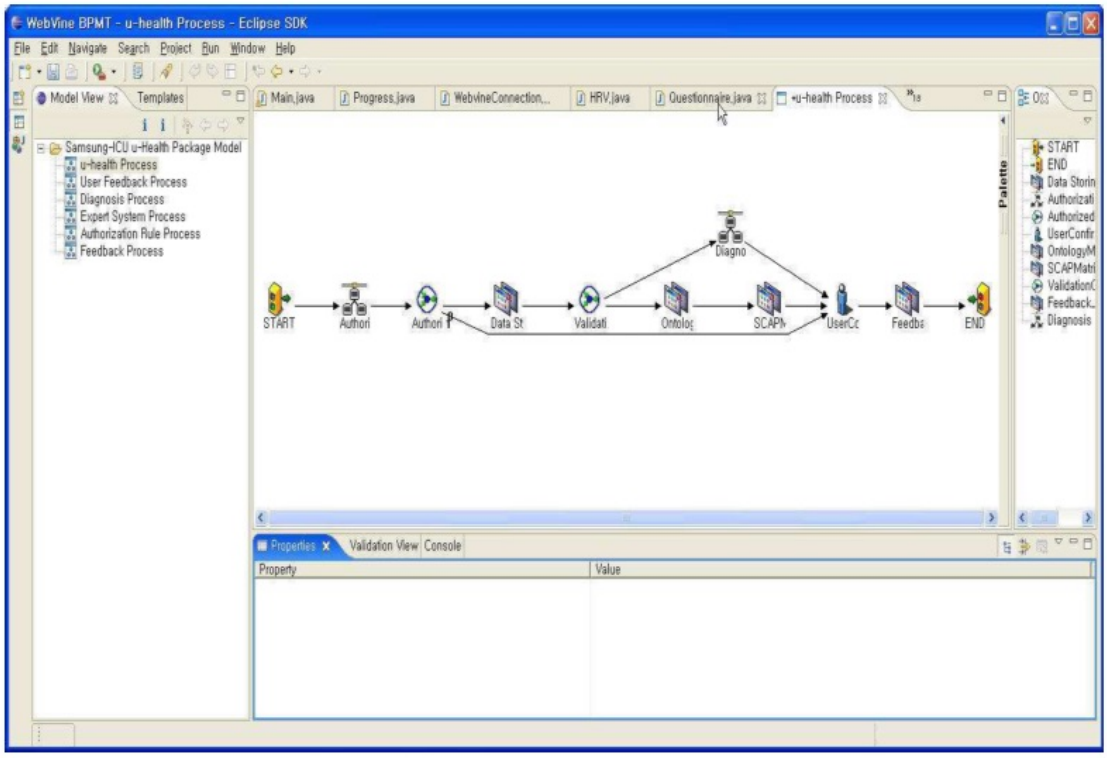


Figure 2.12 Mobile μ-Healthcare Service System: Application designer [13]

(vi) **Appy Pie :**

Appy Pie [17] is a cloud based mobile application builder for creating hybrid mobile applications for multiple mobile platforms (iOS, Android, Windows Phone, Blackberry and FireOS). The platform was designed specially to enable people who lack the necessary skills to build a mobile application but want to develop a mobile applications. Appy Pie utilises a menu driven user interface, that guides users throughout each step of a three stage linear development process. Appy pie's development process consists of Selection, Design and Build; each activity is summarised below:

(a) **Selection:** Requires the user to input a unique name for the application, select a category (domain) and predefined theme.

(b) **Design:** This activity enables the user to customise the look and functionality of the mobile application. Users can add multiple 'app pages' (functionality) to their application. The platform provides five categories: social, multimedia, contact, commerce and information, each consisting of a limited number of 'app pages'. Each 'app page' can also have selected parameters modified via the on screen controls to tailor the functionality to their requirements.

In addition to the menu system, the user can interact with a web-based representation of their application. Not only does this provide the user with an insight to how the application will look but it also can be used to quickly navigate and modify the 'app page'.

(c) **Build:** Following the on screen instructions the users submit their application to be built. Based on the subscription, users can choose what mobile platforms the application is going to be built for. This process can take several hours to complete, once ready the application will appear in the users dashboard.

(vii) **Microsoft App Studio :**

Microsoft App Studio [19] is free web application developed by Microsoft. It

has been designed to enable users to quickly develop, test, publish and maintain applications for Window Phone and Windows operating systems. Microsoft App studio deploys similar menu driven approach to seen in the Appy Pie. However, adopts a non-linear development process consisting of the following activities Start New, Content, Tiles, Settings and Finish.

- **Start New:** Requires the user to input a name for the application and choose between various templates or build a empty application from scratch.
- **Content:** Allows the user to edit the application's content by adding 'sections' such as menus, social media integration and custom API's. Each section can be customised using the various on screen controls. Here the theme can be customised, again users can opt to choose between a preconfigured theme or create their own.
- **Tiles:** Allows the user to configure how the application is displayed on a Windows Phone 10 or PC running Microsoft Windows 10.
- **Settings:** Allows the user to change the applications title and enter the app stores association details.
- **Finish:** The users application is then created, this process take a few minutes to complete.

2.3 Discussion

This section provides a discussion of two key areas. The first examines the strengths and limitations of existing approaches to personalisation. This is followed by a review of the process of personalised development.

2.3.1 Strengths & Limitations of Existing Approaches

Each existing works discussed in Section 2.2.3, demonstrates how the various approaches to personalisation have been utilised within mobile applications. The following discusses each approach to personalisation and related works highlighting the strengths and limitations. A summary of which is shown in Table 2.4.

(i) **Location-Based:**

Location based personalised services are becoming more and more widespread as a result of the onboard capabilities of mobile devices. Location data enables a mobile application to react immediately to the changes in the user's environment providing the application with the ability to deliver the right service at the right time, by modifying service data and content as demonstrated by the 'FLAME2008 project'. However, security and privacy are major concerns with this approach [78]. Also, as a stand-alone mechanism, location based data is not always a suitable approach to for personalising a mHealth application. Instead it is often used alongside user profile modelling & machine learning to provide further context of the current situation of the user as seen in [21, 22].

(ii) **Preference-Based:**

Preference-based approach allows the end-user to explicit specify their own requirements at any time based upon a predetermined series of parameters. Preference based

personalisation was used during the FLAME2008 project, to allow users to configure their interests. Again, it is typically used in conjunction with other personalisation mechanisms to provide additional context to the application [35].

(iii) **User profile modelling & Machine learning:**

The most popular approach for achieving personalisation in mobile applications is via user profile modelling & machine learning. The user model serves as a template for generating distinct user profiles for different users within the context of the application.

Although this approach is heavily centred around the user's characteristics to drive personalisation and is capable of supporting the work motivations defined in Blom's taxonomy. All of the works discussed that implement this approach, target a particular user demographic. However, user profile modelling & machine learning enables an application to support sophisticated, personalised functionality that is capable of supporting multiple users that belong to a particular demographic. Kay *et al.* [79] argues that it takes a considerable amount of time and effort to build a sufficient dynamic model that contains the necessary characteristics and attributes of the user. As a result this approach would not be sustainable for supporting a diverse range of personalised healthcare scenarios.

(iv) **End-User programming:**

The end-user programming paradigm enables domain experts, such as a healthcare provider, who do not possess the skills and expertise of a mobile application developer to become involved in the development process [14]. Each of the end-user programming solutions discussed share similar characteristics. Each adopt techniques to reduces the complexity of mobile application development, enabling the functionality (service logic) of a mobile application to be tailored to accommodate for the individual differences and work goals of the end-user.

Out of the three end-user programming solutions discussed, the Mobile μ-Healthcare Service System (MUSS) [13] is specifically designed to be used by a healthcare provider to produce a personalised mobile application for the healthcare consumer. However, to reduce the complexity of mobile application development, requires the healthcare professional to modify the 'services' process to provide personalised functionality. Although the services supported by the framework are designed specifically for healthcare. The technique to personalise functionality potential limits the scope of the application as a consequence of the healthcare professional's ability to use the system. Moreover, the framework is only capable of supporting a limited range of functionality that are currently limited to a one-dimensional pipeline, making complex logic difficult to model.

Unlike the MUSS framework, the Appy pie and Windows App studio platforms guides users through the various stages of development, via streamlined menus and options that use lay terminology. However, they are both designed to support a wide range of use cases, as a result provide simple and generalised functionality, that are not suitable for supporting the diverse range of personalised healthcare scenarios.

Table 2.4 Summary of existing and related work

Existing work	Approach(es)	Summary of Personalisation	Strengths & Limitations
Mobile U-Health Service System	• End-User Programming	Functionality is created by healthcare providers, using an application designer to form a web based mobile application	• **Strength(s):** – Applications are built by the domain expert • **Limitation(s):** – Limited functionality support – Restricted to specific set of sensors – One dimensional pipeline for personalisation
Flexible Semantic Web Service Management Environment	• Location Based • Preference Based	Provides personalised meaningful information to the general public based on the users current situation	• **Strength(s):** – Services provided by the system can be extended (server side) • **Limitation(s):** – User has to manually request each time personalised information – Designed for a specific scenario – Reliant on location based data which may not be always available

Table 2.4 Summary of existing and related work

Existing work	Approach(es)	Summary of Personalisation	Strengths & Limitations
mHealth solutions for cancer survivors'- Engagement in healthy living	• User Profile Modelling	Queries an ontology to evoke rule-based machine intelligence and decision making to identify the required functionality for the user	• **Strength(s):** – Functionality is identified based upon the users profile • **Limitation(s):** – Small number of supported functionality (11) – Designed for a specific healthcare scenario
Assisting people with dementia in mobile applications	• User Profile & Machine Learning • Location Based	Ontology based approach to provide personalised context aware assistance services for users with dementia, via the use of machine learning	• **Strength(s):** – Utilises machine learning to provide the right functionality based upon the current users situation • **Limitation(s):** – Application is designed for a specific healthcare scenario – Limited functionality support

Table 2.4 Summary of existing and related work

Existing work	Approach(es)	Summary of Personalisation	Strengths & Limitations
Travel Assistance: Help-on-demand services in pervasive environments	• User Profile & Machine Learning • Location Based • Preference Based	Provide personalised travel assistance services based on the user profile model, user requests and preferences	• **Strength(s):** – Combines various approaches to provide rich personalised services • **Limitation(s):** – Designed for a specific scenario – Limited functionality support
Appy Pie	• End-User Programming	Utilises end-user programming and aids and guides the user through the mobile application development process utilising wizards and forms.	• **Strength(s):** – Supports various mobile platforms – Streamlined development process for the layperson • **Limitation(s):** – Limited functionality support – Subscription based

Table 2.4 Summary of existing and related work

Existing work	Approach(es)	Summary of Personalisation	Strengths & Limitations
Microsoft App Studio	• End-User Programming	Utilises end-user programming and aids and guides the user through the mobile application development process utilising wizards and forms.	• **Strength(s):** – Rich functionality • **Limitation(s):** – Limited functionality support – Complication application builder

2.3.2 Process

During the creation of a personalised healthcare plan, if a healthcare provider identifies that a mobile application would be a suitable clinical metric, there are currently two options: seek a mobile application that is already available in the mobile market place that best fits the consumers needs; or focus their efforts in developing a personalised mobile application that meets the requirements of the consumer. However, personalised healthcare is driven by optimising the healthcare delivery process to provided targeted care to the unique requirements of the consumer. Therefore to increase the effectiveness of a mHealth applications it is integral that they are developed in accordance to the unique requirements of the healthcare consumer to provide the necessary functionality to support their personalised care.

Of all of the approaches discussed earlier, end-user programming not only enables healthcare providers to develop mobile applications, it can be used to facilitate the rapid production of mobile applications with functionality that is required and tailored to the healthcare consumer; bridging the gap between supply and demand of personalised mHealth applications [80, 81]. However, many existing end-user programming solutions are limited to a small number or provide simple and generalised functions that are not suitable for supporting the diverse range of personalised healthcare scenarios. In addition, many existing solutions either focus specifically on particular aspect of healthcare and do not provide any mechanism to expand the functionality they support. Therefore there is a need to explore and categorise existing mHealth application functionality to determine their suitability of the framework.

Olff argues to develop credible and effective mHealth applications requires the inclusion of a multidisciplinary team that consists of healthcare providers and mobile application developers [46]. The combination of the domain expertise is recognised as good practice

with software engineering literature [44, 64]. However, the requirements engineering activity is particularly challenging aspect of the software engineering process. LeRouge highlights that previous studies have indicated that the lack of shared understanding of the end users are among the major problems of the requirements gathering process [82]. Although the healthcare provider understands the requirements of the consumer, it can be difficult for mobile application developers to quickly identify and translate the healthcare requirements to form a personalised mobile application, as they are unfamiliar and lack the sufficient knowledge

Moreover, Lee *et al.* [13] highlights that following traditional mobile application development routes is not economical or a sustainable method for developing personalised applications. As discussed earlier, many of the approaches to personalisation either target a specific user demographic or are not suitable as a standalone personalisation mechanism that is capable of supporting a diverse range of personalised healthcare scenarios.

Recently ontologies have become an increasingly popular approach in system and software engineering. Skillen *et al.* highlights several advantages for the use of ontologies such as interoperability, knowledge sharing and reuse across several application domains [22]. Many of the works discussed earlier successfully utilise ontologies in various scenarios. For example, the FLAME2008 [71, 72] utilises several ontologies to provide users personalised services and information. Myneni *et al.* [20] uses an ontology to identify the end users required functionality based upon the user profile; Skillen et al. [21] exploit an ontology to facilitate machine learning to drive personalised assistance services. Both examples demonstrate that ontologies provide the necessary flexibility required to model a domain of discourse. As highlighted towards the beginning of this chapter, the device a user owns dictates the outcomes of several decisions throughout the engineering process, such as API version and hardware limitations defining the scope of any potential mobile application. Hence, an ontology would be a suitable mechanism to model both the

characteristics of a mobile device and functions whilst also would be capable of addressing the issue of extensibility as seen in existing works whilst also providing a system with sufficient knowledge that can be queried to assist healthcare professionals throughout the development process.

2.3.3 Sign posts

Based upon the discussions, regarding existing approaches and the process of personalised mHealth development presented throughout Section 2.3 there is a need for the development of an extensible ontology-driven framework that enables healthcare professionals to produce personalised hybrid mHealth applications for a healthcare consumer. As a result the following signposts that define the objective, purpose and challenges it aims to address is presented in Table 2.5. These signposts will be used to direct, govern and evaluate the products of this research.

Table 2.5 Sign Posts

Signpost	Description
Taxonomy	**Objective:** Analyse and Categorise existing healthcare related functions within mHealth applications designed to be used by healthcare consumers. **Purpose:** Is to identify the scope of functions that will and will not be supported by the framework and provide a tool for classifying health related functions in mHealth applications designed for healthcare consumers. **Addresses:** The limited scope of functions available to the framework as seen in existing end-user programming solutions.
Ontology	**Objective:** Is to encapsulate knowledge associated with the development of personalised mHealth applications. **Purpose:** Is to provide the framework with sufficient knowledge that can be interrogated to make critical decisions associated with the development of personalised mHealth applications, ultimately compensating for the missing/lack of mobile application development expertise of a typical healthcare professional. **Addresses:** The reliance of mobile application development domain knowledge by the healthcare professional. Also addresses the issues with extensibility with regards to expanding support of new functions
Framework	**Objective:** Is to provide a platform that enables healthcare professionals to build personalised mHealth applications on demand for healthcare consumers. **Purpose:** Is enable healthcare professionals to develop, without intervention from mobile application developers, effective personalised mHealth applications for healthcare consumers. **Addresses:** The limited personalisation capabilities of existing works. Reduces unnecessary complexities during the creation of a mobile application. The restrictions of existing solutions that only support a particular healthcare scenario.

2.4 Summary

To summarise, this chapter has introduced, discussed and analysed relevant concepts, challenges and ideas that are considered significant and influential to the foundations of this research. As a consequence of the discussions throughout this chapter there is a clear indication of the need for an ontology driven framework that enables healthcare professionals to create mHealth applications that contain the necessary functions that are required and personalised for a individual healthcare consumer.

Chapter 3

Research & Development Methodologies

This chapter presents the justification of the adopted research and development methods; tools and techniques to achieve each of the objectives defined in Section 1.2 to ultimately accomplish the aim of this research. The chapter is organised into six sections each concentrates on a particular aspect of this research and is organised as follows. The first section provides an overview of the research design and how each objective influences the route the research took. The next three sections each focus on a individual original contribution to knowledge, with respect to the research & development methods tools and techniques used exclusively to developed the taxonomy, ontology and framework, respectively. The penultimate section outlines the tools and techniques used throughout this research. This chapter concludes with a table that summarises each objective, the contribution to knowledge (where applicable) and the methods, tools and influence it has on the research.

3.1 Research Design

The aim of this research is to develop an ontology-driven framework that enables healthcare professionals to produce personalised mHealth applications for a healthcare consumer. In order to achieve the aim, requires the completion of four objectives (see Section 1.2). Each objective is a step that shapes the individual products of this research and are vital in achieving the overall aim. As a result, Figure 3.1 is a representation of the flow of this research. For each of the artefacts; the methods, tools and techniques used to create and evaluate are discussed in further detail later in this chapter.

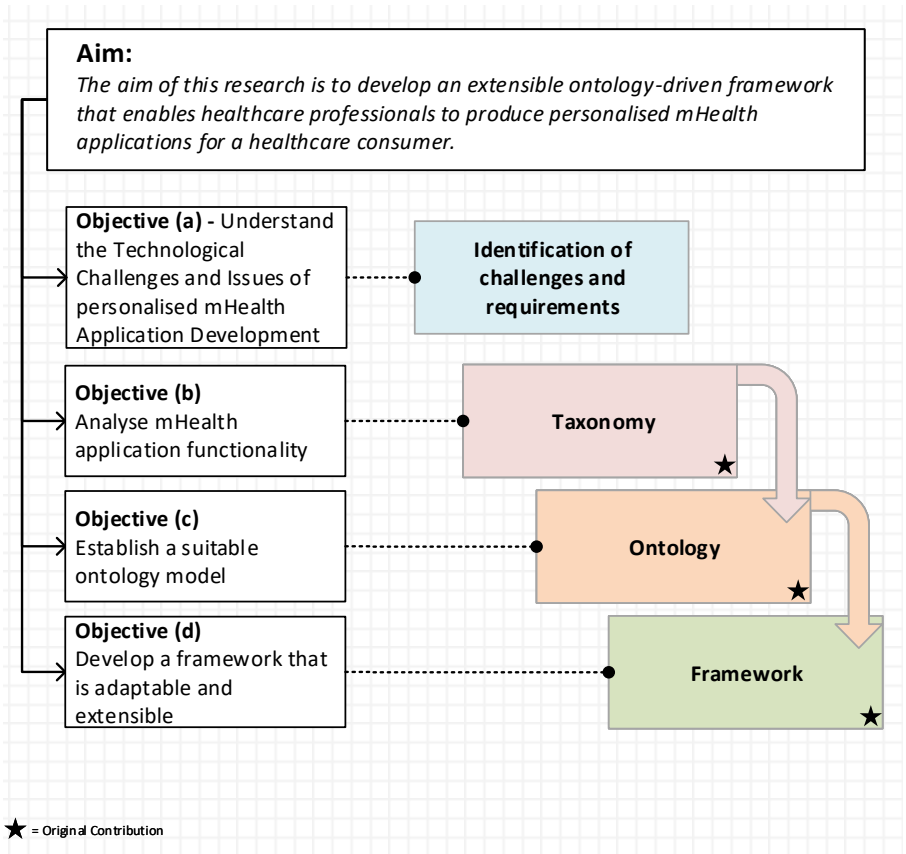


Figure 3.1 Research flow

3.2 Taxonomy

A taxonomy is the science and practice of classification, enabling users to classify entities of interest in a particular domain in a hierarchical structure [83]. When developing a taxonomy, it is important to consider appropriately the separating elements of a group into subgroups that are mutually exclusive, unambiguous, and as a whole, include all possibilities. For a taxonomy to be applicable in the real world, it must also be uncomplicated and easy to understand and use [84].

Bailey [85] provides a thorough review of taxonomy development and identifies two main techniques to achieve classification, empirical and conceptual. An empirical taxonomy derives classification from empirical data utilising statistical methods, such as cluster analysis. A conceptual taxonomy represents types of concepts rather than empirical cases; however, such data may be brought in at a later stage.

Throughout the literature, researchers have formally and informally categorised mHealth applications in various ways, such as the type of end user, security & privacy and intended purpose [31, 62, 84]. All of these works focus specifically on the mobile application as a single entity rather than the individual functions it provides the end user. As discussed in Section 2.3, for a personalised mHealth application to be effective requires functions that are suitable to support the needs of the healthcare consumer. Therefore, this research has chosen to adopt the use of a conceptual taxonomy to classify the *individual* mHealth functions within mHealth applications.

3.2.1 Taxonomy Development Approach

Described in the subsequent sections are the guidelines, development, testing and evaluation methods adopted for the development of the taxonomy.

3.2.1.1 Nickerson's Taxonomy Attributes

Nickerson argues that a good taxonomy should possess five key attributes [83], each of which are discussed below.

- **Concise:** It should contain a limited number of dimensions or a limited number of characteristics in each dimension, because an extensive classification scheme with many dimensions and many characteristics would be difficult to comprehend and difficult to apply.
- **Mutual exclusivity:** No function falls into more than one category.
- **Sufficiently Inclusive:** It should contain enough dimensions and characteristics to be of interest.
- **Comprehensive:** It should provide for classification of all current objects within the domain under consideration.
- **Extendible:** It should allow for additional dimensions and new characteristics within a dimension when new types of objects appear.

Works such as [31, 83, 84] have utilised Nikerson's attributes for various purposes including governing development decisions and as criteria during evaluation. Therefore Nikerson's attributes have been adopted for the development of the taxonomy and are to be used as guidelines during development and as criteria during the evaluation of the taxonomy.

3.2.1.2 Development Method

Therefore, to ensure that these attributes are present within the taxonomy requires combining qualitative data derived from sources of existing grounded theory and empirical data, via a systematic review of existing literature and analysis of current mobile applications. The development process begins by first establishing the purpose and scope of the and will determine the boundaries and use of the taxonomy. The next step requires identifying the sources for data collection, definition of inclusion/exclusion criteria and techniques for data extraction and documentation. During the extraction process the source are analysed and 'codes' (words phrases, descriptions) are to be extracted and documented. The next stage analyses the 'codes' to determine relationships and identify and define mutually exclusive categories. It is not the intention of the taxonomy to present an absolute classification scheme, but rather to be used as a tool for examining the individual mHealth functions and identify those suitable of being supported by the framework. This way the taxonomy has practical use outside of the scope of this research.

3.2.1.3 Testing and Evaluation

Finally, to test taxonomy, a sample of mHealth applications were chosen at random. As already mentioned an mHealth application consist of various mHealth functions, each function will be documented and categorised into one on the distinct categories defined in the taxonomy. The results will be analysed against Nikerson's criteria defined earlier to determine the robustness of the taxonomy.

3.3 Ontology

To ensure that the framework contains sufficient knowledge to build personalised mHealth applications and also assist in addressing the technological challenges discussed in Chapter 2, required selecting a suitable technique capable of encapsulating the necessary knowledge. Therefore this research has opted to develop an 'skeleton ontology model'.

Unlike software engineering, there is not a single definitive, standardised or mature methodology for developing an ontology [86, 87]. Therefore there has been several independent, ad-hoc methodologies developed such as [88–91]. Each methodology was designed to address challenges such as clarity, reuse and extensibility [92, 93].

The methodology chosen for the development of the mHealth Application Function ontology model is the Uschold and Grunninger Skeletal Methodology [88]. The methodology was formulated using the experiences from multiple authors. Several reasons influenced this decision. Rather than providing a precise step-by-step instructions, the methodology illustrates a clear set of guidelines that govern the development process as a whole, thus the development process is flexible [87]. This allows for a fine balance between control during the development of the ontology and flexibility surrounding choices surrounding modelling approach, conventions, tools and ontology language.

3.3.1 Skeletal Methodology Overview

Figure 3.2 represents the Skeletal Methodology. The various colours highlight 5 different stages of the ontology development process. Documentation is a vital continuous process throughout the design and development of the ontology and is responsible for documenting key aspects of the ontology's development to support knowledge sharing and reuse.

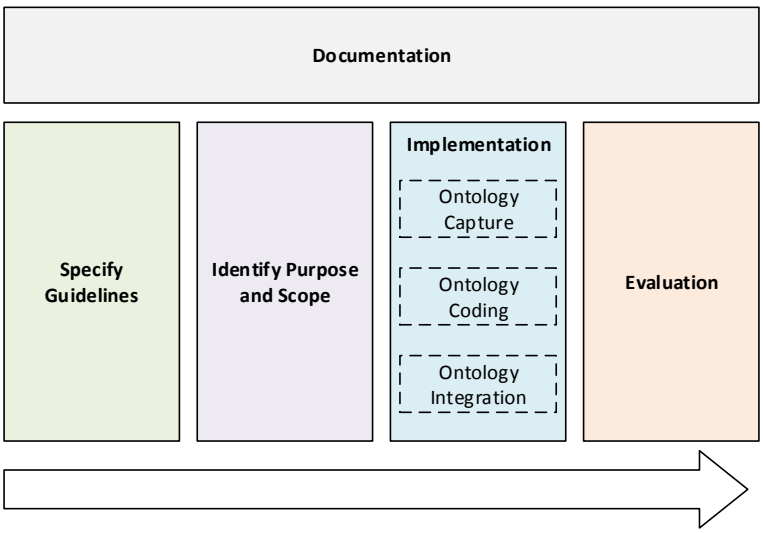


Figure 3.2 Uschold and Grunninger Skeletal methodology [88]

1. **Specify Guidelines:** The purpose of this stage is to establish a series of agreed guidelines that govern the development process as a whole.

2. **Identify purpose and scope:** The objective of this stage is to understand and define why the ontology is being developed, its possible uses, intended users and boundaries.

3. **Implementation:** Consists of 3 stages, 2 of which are compulsory stages Capturing & Coding that relate to the design and development of the ontology. The third is optional and is associated with the integration or reuse of existing ontologies.

(a) ***Capture*** -Identification, organisation and structure key entities within the domain such as concepts, terms relationships to form a conceptual model of the domain of interest.

(b) ***Coding*** - Explicit representation of the conceptualisation captured in the previous phase using a formal language. This requires committing basic terms used to specify the ontology, choosing a representation language and finally writing code.

(c) ***Integrating Existing Ontologies*** - *Optional* Integrate or reuse existing ontologies.

4. **Evaluation:** Although the ontology will be routinely tested throughout the development process to ensure that the knowledge modelled is satisfiable. Evaluating ontologies is still an developing area and there are various different approaches for evaluating an ontology [94, 95]. Therefore the ontology will be assessed from three different perspectives. The first will utilise the Ontology Pitfall Scanner, to identify the presence of any potential pitfalls. The second utilises the HermiT reasoner [96] and description logic queries to demonstrate the correctness of the knowledge modelled within the ontology. Finally, the competence of the ontology will be assessed against the criteria established during stage 1 of the ontology development.

3.4 Framework

To achieve the final objective requires combining the products of this research into in the form of a framework that represents an ontology driven approach to personalised mHealth application development, that enables healthcare professionals to produce such application for the healthcare consumer. It is not the intention of this research to implement a fully operational platform, but rather a framework which represents the architectural components of such a system. Therefore, this research has chosen to adopt and adapt key technical processes defined in the ISO15288 Standard (System and software engineering - System life cycle processes) [97] to produce the Personalised Mobile Application Development (PMAD) Framework.

3.4.1 Development Approach

As presented in Figure 3.3, the design and development of the framework consists of two stages and four sequential processes:

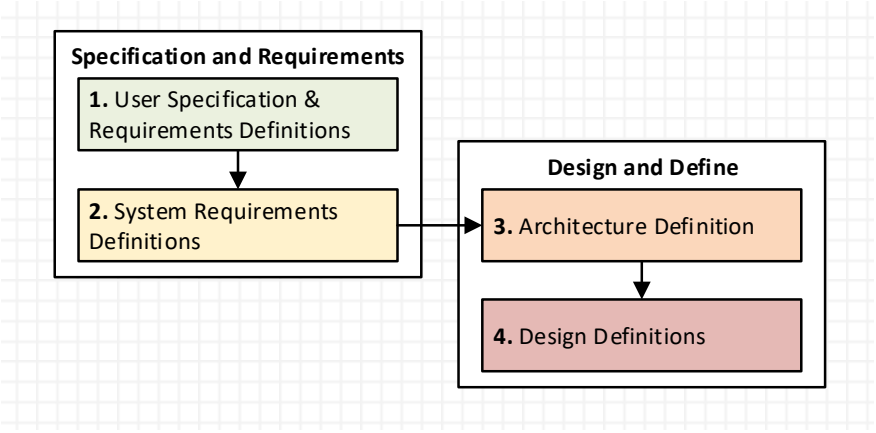


Figure 3.3 Framework development process

Specification and Requirements

1. Stakeholder Needs & Requirement Definitions: The purpose of this process is to define the stakeholders requirements to identify the required capabilities of system. This process will utilise the challenges discussed in Chapter 2 alongside the stakeholders requirements to identify the necessary interaction and operational requirements of the system from the perspective of the users.

2. System Requirements Definitions: Is the process of transforming the stakeholders needs into a technical requirements that meets the needs of the user.

Design and Define

3. Architecture Definitions: Is the generation of a system architecture that meets the users and technical requirements of the system. During this process UML Activity diagrams will be utilised to identify key framework components.

4. Design Definitions: The purpose of this process is to provide sufficient detailed information about the system and its components as defined in the model views of the system architecture.

3.4.2 Evaluation

As the main aim of this research is to create a framework, it is vital that the frameworks personalisation capabilities are evaluated. The evaluation of the framework will be conducted on a small-scale software concept that consists of the frameworks critical components. The following tests are designed to determine if the framework contains the necessary characteristics to achieve the overall aim of this research.

(a) **Application Feasibility**: This test assesses the frameworks capability to determine the feasibility of the application regarding the development characteristics (platform, hardware dependencies, etc.). This requires the creation of hypothetical use cases to simulate both the positive and negative outcomes. Within this context a positive outcome would be the mobile devices characteristics are sufficient in addressing the requirements of the personalised mHealth application. Where as a negative outcome would be the opposite, i.e the mobile devices characteristics are not sufficient in addressing the requirements.

(b) **Personalisation:** This test will assess the personalisation capabilities of the framework. The software concept will be used to "*build*" each of the personalised applications defined in each of the use case profiles. The concept will generate a design specification which will be compared against the requirements in the scenario.

3.5 Tools and Techniques

This section outlines, in no particular order the tools and techniques used throughout this research. A detailed description of each tool/technique and how it was utilised is discussed in the relevant chapters.

- **Taxonomy:** A taxonomy is the science and practice of classification. Taxonomies enable users to classify entities of interest in a particular domain in a hierarchical structure [83]. When developing a taxonomy, it is important to consider appropriately the separating elements of a group into subgroups that are mutually exclusive, unambiguous, and as a whole, include all possibilities. For a taxonomy to be applicable in the real world, it must also be uncomplicated and easy to understand and use [84].

- **Systematic Review:** A systematic review is a type of literature review that collects and critically analyses multiple research studies or papers. A review of existing studies is often quicker and cheaper than embarking on a new study.

- **Coding:** Coding is a method for arranging entities in a systematic order to make something part of a system or classification [98]. Coding will be used to identify the distinct taxons (classes) during the development taxonomy.

- **Use Case Scenarios:** Modelling an entire domain is a time-consuming task. As this research has a limited time frame for completion, a set of fictitious use case scenarios where created, based on grounded theory and are outcome of the taxonomy chapter. These represent a healthcare consumers healthcare plan where a mobile application is recommended as a clinical metric. Each profile consists of background information regarding the health status of the consumer, attributes, treatment and a description of the mHealth functions required. The set of use case profiles will be

used as a tool used to assist in the design, testing and evaluation of the ontology and to evaluate the framework.

- **Unifiied Modelling Language:** The Unified Modelling Language (UML) is a general-purpose, developmental, modelling language in the field of software engineering, that is intended to provide a standard way to visualise the design of a system or application [99]. UML consists of various models that provide system and software engineers insights into different aspects of an application or system. Described below are the diagrams used during this research:

  - ***Use Case Diagram:*** During the development phase of the ontology, use case diagrams were used alongside the use case profiles for two purposes. The first is to quickly represent the usage requirements of the personalised mHealth application and the second is to be used to identify functionality of the framework.

  - ***Activity Diagram:*** Used by System and Software Engineers to model the flow of computational and organisational through a series of stepwise activities using a standardised series of nodes to depict different functions [100] [101]. Throughout the frameworks development, activity diagrams were used to gain a quick understanding and envisage how components of the framework would function.

- **Android Developer Tools:** The Android platform is the fastest growing mobile platform that powers millions of devices around the world [102]. The platform provides a freely available Android Developer Tools(ADT). The ADT's provides a full Java IDE and features for developing , debugging and packaging Android Applications. Also the tools allows virtual devices to be created that are capable of emulating various hardware configurations. When necessary, the ADT's were used

to develop small scale applications to be used as a tool to assist in the modelling of the various ontology components.

- **Ontology:** An ontology in computer science is defined as a "explicit specification of a conceptualisation" [103]. Researchers at the time adopted a naïve realistic approach when developing ontologies, choosing to denote a systematic representation of only the necessary knowledge required to perform a particular task, referred to as the universe of discourse. The universe of discourse represents a set of named entities that are described using primitive concepts and formal axioms that constrain the interpretation and use of these terms to form an ontology [103, 104, 94, 105]. From a simple perspective, developing an ontology typically requires defining: individuals, classes, properties and relationships [106], these will be discussed in more detail in Chapter 5.

- **OWL:** An ontology language defines which language constructs can be used in an ontology and also defines the formal semantics of that language. There are several ontology languages that are discussed throughout the literature, all of which have been designed to represent semantic information, so it can be shared, processed and interpreted by machines. OWL has become an increasingly popular choice for developing an ontology within the literature and has is the ontology language adopted for this research. OWL is a semantic web language designed to represent rich and complex knowledge about things, groups of things and the relationships between them [107]. OWL is part of the growing stack of W3C recommendations related to the Semantic Web and is built on top off the Resource Description Framework (RDF) schema, adding more vocabulary for describing properties and classes. This is discussed in further detail in Section 5.1.5.

- **Protégé** is a freely available open source ontology editor developed by the Centre for Biomedical Informatics at the Stanford University School of Medicine. It provides a

rich environment with full support for OWL, including visualisation tools to interact with the ontology throughout development, advanced support for tracking down inconsistencies and operations to modify the various ontology components[108]. The version used throughout this research was Protégé 4.30. Again this is discussed in more detail in **??**.

- **OWL API:** The OWL API is an open source Java based API and reference implementation for creating, manipulating and serialising OWL ontologies [109]. The OWL API version 4.0.1 is to be utilised during the testing of the framework.

- **Glossary of Terms:** Is a central repository that include all the relevant terms of the domain such as concepts, instances, attributes, relations between concepts; a description in natural language, synonyms and acronyms [110]. The glossary of terms will be used to document the terms used within the ontology and as a tool to help build the ontology model.

- **Concept Map:** A concept map is a graphical conceptual representation that depicts the relationships between concepts and is used to organise and structure knowledge [111]. Concept maps will be used to organise codes during the development of the taxonomy and also utilised throughout the development skeleton ontology model.

- **Systematic Review:** A systematic review is a type of literature review that collects and critically analyses multiple research studies or papers. A review of existing studies is often quicker and cheaper than embarking on a new study.

- **Coding:** Coding is a method for arranging entities in a systematic order to make something part of a system or classification [98].

- **Use Case Scenarios:** Modelling an entire domain is a time consuming task. As this research has a limited time frame for completion, a set of fictitious use case

scenarios where created, based on grounded theory and are outcome of the taxonomy chapter. These represent a healthcare consumers healthcare plan where a mobile application is recommended as a clinical metric. Each profile consists of background information regarding the health status of the consumer, attributes, treatment and a description of the mHealth functions required.The set of use case profiles will be used as a tool used to assist in the design, testing and evaluation of the ontology and to evaluate the framework.

- **Unifiied Modelling Language:** The Unified Modelling Language (UML) is a general-purpose, developmental, modelling language in the field of software engineering, that is intended to provide a standard way to visualise different perspectives system or application [99].
  - ***Use Case Diagram:*** A use case diagram specifies a set of interactions between actors and use cases in order to achieve a particular goal [112, 113].
  - ***Activity Diagram:*** Used by System and Software Engineers to model the flow of computational and organisational through a series of stepwise activities using a standardised series of nodes to depict different functions [100] [101].
- **Android Developer Tools:** The Android platform is the fastest growing mobile platform that powers millions of devices around the world [102]. The platform provides a freely available Android Developer Tools(ADT). The ADT's provides a full Java IDE and features for developing , debugging and packaging Android Applications.
- **OWL:** An ontology language defines which language constructs can be used in an ontology and also defines the formal semantics of that language. OWL is discussed in further detail in Section 5.1.5.

- **Protégé** is a freely available open source ontology editor developed by the Center for Biomedical Informatics at the Stanford University School of Medicine. The version used throughout this research was Protégé 4.30. Again this is discussed in more detail in Chapter 5.

- **OWL API:** The OWL API is an open source Java based API and reference implementation for creating, manipulating and serialising OWL ontologies [109].

- **Glossary of Terms:** Is a central repository that include all the relevant terms of the domain such as concepts, instances, attributes, relations between concepts, a description in natural language, synonyms and acronyms [110].

- **Concept Map:** A concept map is a graphical conceptual representation that depicts the relationships between concepts and is used to organise and structure knowledge [111].

3.6 Summary

In summary, this chapter has presented and discussed the considerations regarding the tools, techniques, research and development approaches adopted to achieve the objectives of this study, a summary of which is presented in Table 3.1.

Table 3.1 Summary of contributions, methods, tools and influence they have on this research

Objective	Contribution	Method(s)	Tool(s)/ Technique(s)	Influence on this research
a	n/a	• Literature Review		• Identification of challenges for developing personalised mHealth applications • Identifies framework requirements
b	Taxonomy	• Classification	• Taxonomy • Coding • Concept Maps	• Determine the feasibility of mHealth functionality • The design of the use case scenarios • Identifies components in Ontology
c	Ontology Skeleton	• Skeletal Methodology	• Ontology • Use Case Profiles • UML: Use Case Diagrams • Concept Maps • Glossary of Terms • OWL • Protégé	• Encapsulation of knowledge associated with development of personalised mHelath applications
d	Framework	• ISO15288 (processes)	• OWL API • Conceptual model • Use Case Profiles • UML: Activity Diagrams	• Fulfilment of the aim of this research

Chapter 4

mHealth Application Function Taxonomy

The purpose of this chapter is to present the design and development considerations for the mHealth Application Function Taxonomy, which has been developed to gain an understanding the functions and their associated characteristics within mHealth applications. The chapter begins by presenting an overview of the development process of the taxonomy. The following sections discuss each process in detail from the data collection and analysis activities. Section 4.4 presents the taxonomy as a completed artefact including the definitions for each of the distinct function categories. Followed by testing and evaluation of the results and a discussion surrounding the types of functions to be supported by the mHealth Application Development framework. To conclude this chapter summarises the key and influential components presented in this chapter.

4.1 Method Overview

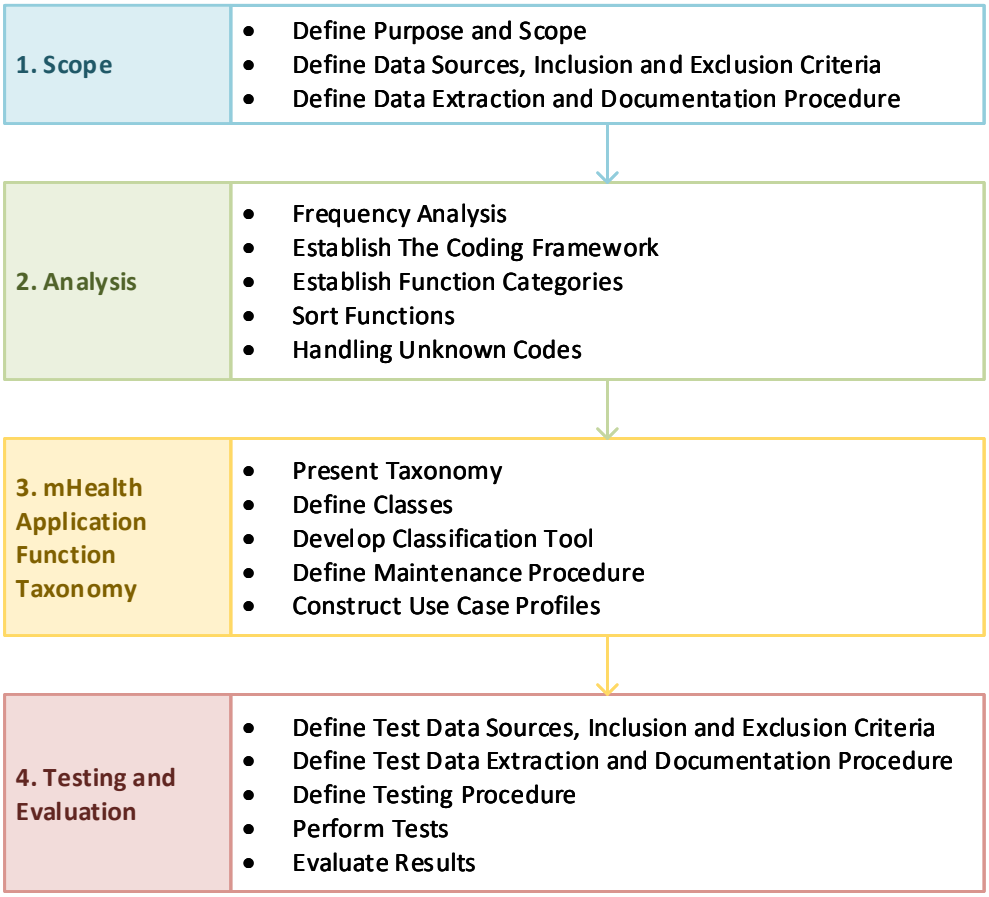


Figure 4.1 Taxonomy development approach

The development of the mHealth Application Function Taxonomy followed the approach described in Section 3.2. As can be seen in Figure 4.1 the development consists of four stages Scope, Analysis, mHealth Application Function Taxonomy and Testing and Evaluation. Each stage consists of various activities and are described in detail throughout Sections 4.2 to 4.5.

4.2 Taxonomy Scope

The focus of this section is to discusses and justify the considerations and decisions made during the data collection phase of the taxonomy. The subsequent sections describe: the scope and purpose of the taxonomy, identifies and justifies the data sources, inclusion and exclusion criteria and data extraction techniques for data collection.

4.2.1 Purpose and Scope

Categorisation of mHealth applications is a technique that has been applied throughout the literature to gain an understanding of mHealth applications and how they are used within a particular context. Works such as [31, 84] categorise mHealth applications based on the intended purpose, user or the overall functionality they provide. However, functionality is not a single entity within a mobile application. The ISO/IEC 9126 standard defines functionality as 'the capability of the software product to provide functions which meet stated and implied needs when the software is used under specified conditions' [114]. Therefore, an mHealth application consists of various *functions* that combine to provide the necessary functionality to enable the healthcare consumer to perform various operations. As highlighted in Chapter 2 for a personalised mHealth application to be effective as a clinical metric, requires providing functions within a mobile application that are suited to the specific healthcare requirements of the consumer. Hence, gaining an understanding of the different types of healthcare functions and their common associated characteristics, rather than the kind of application is a critical factor in achieving the aim of this research.

As identified in Chapter 3, to gain an understanding of the functions and their characteristics this study chose to develop a conceptual taxonomy to classify the *individual* mHealth functions within mHealth applications. The mHealth Application Function Taxon-

omy (mHAFT) was developed as a tool that can be used to analyse, identify and categorise the different type of health related functions available within mHealth applications, that are designed to be used by healthcare consumers. The taxonomy is based on the principles of a functional hierarchy diagram. A functional hierarchy diagram is used by software engineers to organise the overall functionality into functions arranged by mutual attributes to gain an insight into the needs of system [115]. However, rather than grouping the functions based upon their relation to one another in a system, the taxonomy groups functions based upon the service they provide to the user. Nickerson *et al.* describes this as the meta-characteristic on the taxonomy [83].

The objective of the taxonomy is not to produce an exhaustive classification schema, but rather a high-level representation of the core types of mHealth functions. Therefore adopts a top-down development strategy for the development of the taxonomy, as this will identify the core categories and prevent unnecessary complexity [83, 116]. This allows the taxonomy also to be deployed as an appropriate tool that can be used by others outside the context of this work such as, mHealth application developers during the development of mHealth applications. Within the boundaries of this research, the taxonomy is utilised to assess each type of function defined within the taxonomy and determine the scope of support by the framework. The taxonomy also is exploited during the creation of use-case profiles and the development of the mHealth Application Function Ontology Model; as discussed in further detail in Chapter 5.

4.2.2 Data Sources, Inclusion and Exclusion Criteria

Gathering a rich dataset that provides a broad representation of the current functions that are available is vital to the development of any taxonomy. Therefore, a decision was made to manually gather data from two sources, articles and mHealth applications. Data from articles was based upon grounded theory and mHealth applications produce empirical data. Deciding what data to include and exclude from the study required establishing criteria. Listed below are a list of the inclusion criteria for each respective data source. If a potential source did not meet the criteria defined below, it was excluded from the study.

Table 4.1 Inclusion criteria

Type	Source	Inclusion Criteria
Article	• PubMed • Google Scholar • IEEE Xplore	Articles that discussed: the use of, design, development, evaluation, categorisation or personalisation of mHealth applications that are intended to be used by healthcare consumers
Application	• Google Play • App Store (iOS)	Applications that are freely available from the mobile application marketplace and are intended to be used by healthcare consumers

The reasoning behind combining two types of data is to: embrace the terminology that is commonly used throughout the related literature and to ensure the categories identified are synonymous to the functions within the mHealth applications. This also will help accelerate the taxonomy development process. But also provides the coding process with empirical data based on observations of current mHealth applications, since the taxonomy is intended to be used as a tool. Additionally this will also help ensure the taxonomy possesses the attributes defined earlier in Section 3.2.1.1.

4.2.3 Data Extraction and Documentation

The objective of the data extraction and documentation phase was to identify codes[1] so that they could be analysed to form a coding framework. The purpose of the coding framework is to establish thematic relationships to reveal distinct and mutually exclusive function categories based upon the data that was extracted from both the article and applications sources. The procedures that follow, describe how and what data was manually extracted from each type of source.

- **Articles** which met the inclusion criteria defined in Table 4.1 were read in full to extract key information from the article. This included documenting the bibliographical information such as the articles title, keywords used to index the article and abstract. Also documented during this process was any words and or phrases used to describe of the application(s), functionality and functions (if available).

- **mHealth Applications** mHealth applications which met the criteria defined in Table 4.1 were then downloaded to a compatible mobile device and the functionality and functions that are specifically health related to would be observed and documented. The following data would be recorded: application name, developer, application description, description of function(s), alongside any words and or phrases used to within the application or application description to be used as codes.

As a consequence of the inclusion criteria defined in Section 4.2.2, a total of 41 sources consisting of 21 articles and 20 mHealth applications were eligible for this study. Following the extraction process defined above, the 41 sources that met the inclusion criteria combined to produce a sample size of 213 individual mHealth functions following the process defined above. Meaning on average each source consisted of 5 health related functions. Since

[1] only one instance of a code was recorded per source (article/application)

the inclusion and extraction processes were performed manually and the objective of the taxonomy is to define at a high level types (categories) of functions present in mHealth applications and not a exhaustive granular classification system. A sample size of 213 mHealth functions provided a suitable representation of the domain whilst also still being at a size that is manageable. The data obtained from this process was then organised into the retrospective data sets in a table format and can be found in the following appendix Appendix A.4.

4.3 Analysis

The purpose of this stage is to analyse each dataset to identify a series of distinct mHealth application function categories. As discussed in Section 4.2.2 and Section 4.2.3 the development of the taxonomy is composed of two complimenting datasets. Therefore, it was important that data was processed systematically and in manageable iterations, to allow greater control over the developing categories. The subsequent sections provide details surrounding the analysis process demonstrating how the mHealth function categories were devised.

4.3.1 Frequency Analysis

In its raw form, the data extracted was qualitative. Therefore, the first step towards identifying suitable mHealth function categories, required quantifying the data, via frequency analysis. Using a python script (see Appendix A.1) each dataset was processed, calculating the frequency each code occurred. The data shown in Table 4.2 represents descriptive statistics for each dataset and was extracted using a python data analysis library, Pandas [117]. Combined, both of the datasets produced a combined total of 186 codes, of which

72 were unique. This is an initial indicator that there is some common ground between the two data sets, as there are 13 instances of codes appearing in both datasets. The data from each datasets was aggregated forming a single coherent dataset as represented in the stacked bar chart in Figure 4.2.

Table 4.2 A summary of the descriptive statistics relating to the codes extracted from the *Article* and *Application* data sets. Statistics were produced by the Pandas data frame describe function

Statistic	Article	Apps	Combined
Count	83	103	186
Unique	39	46	72
Mean	2.100	2.239	2.749
Min	1	1	1
25%	1	1	1
Median	1	1	1
75%	2	3	3
Max	13	10	14

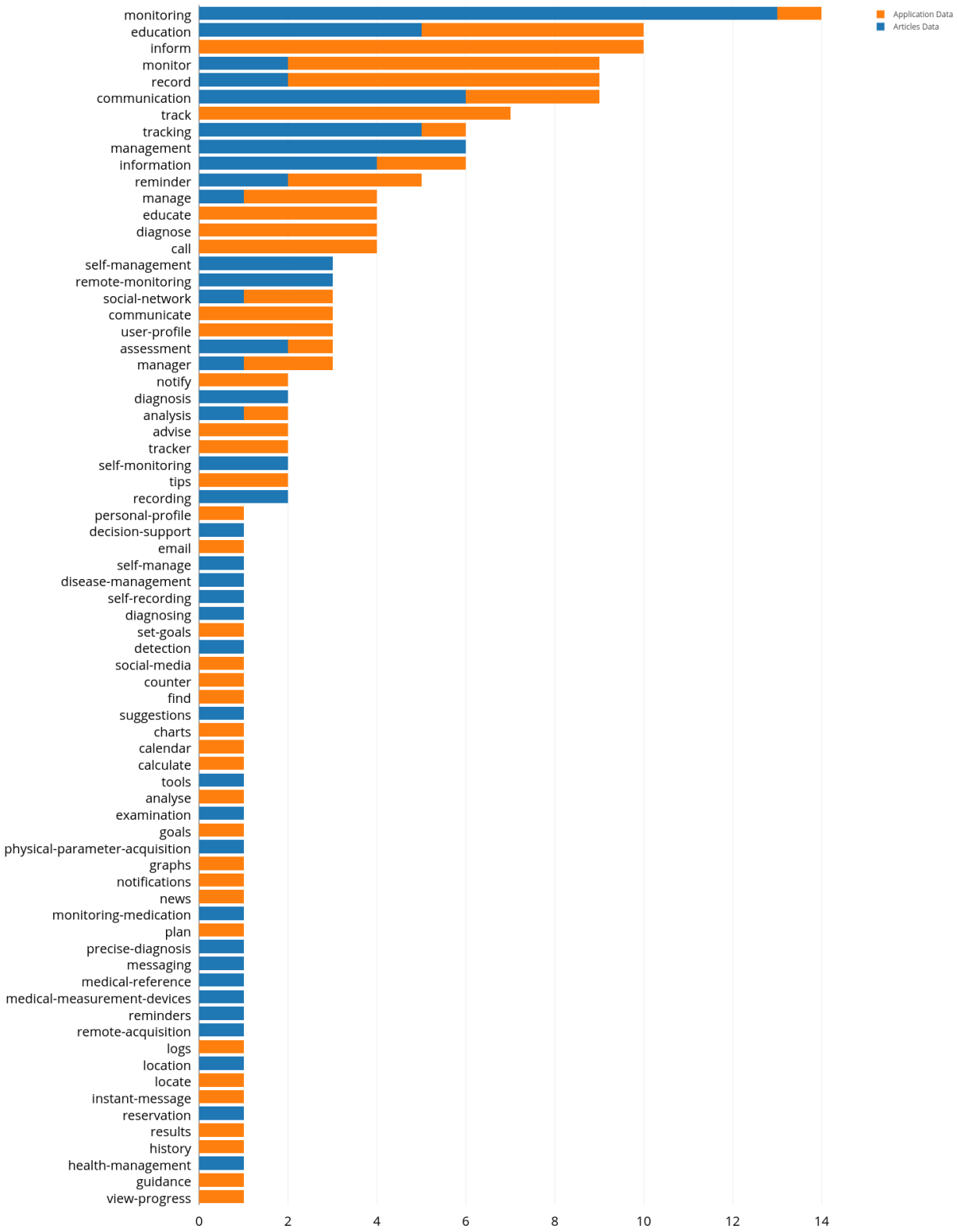


Figure 4.2 Frequency analysis - Article Data (orange), Application (blue)

4.3.2 Coding Framework: Establishing The Foundations

The objective of the second stage required establishing an initial set of thematic relationships that will be used to form the basis of a coding framework. Therefore, it was important to understand and model the lexical semantics between the codes gathered during the data extraction phase. A decision was made to analyse the articles dataset first, since these codes represent themes from grounded theory. In addition, priority was also given to codes that lie within the upper quartile (see Table 4.3) as this consisted of codes that occurred most frequently. The data was processed in several phases and is described below.

Phase One: Identifying Common Origins

The first step was to identify and group codes that shared common origin word and documenting the relationships using a concept map. Figure 4.3 presents two independent concept maps as a result of this process. The relationships between codes represent binary relationships. Words contained within a rectangle illustrate how the two codes are related, while the arrow represents the direction of the relationship.

Table 4.3 Article dataset: Upper quartile codes

Code	Frequency
monitoring	13
management	6
communication	6
education	5
tracking	5
information	4
remote-monitoring	3
self-management	3
assessment	2
monitor	2
diagnosis	2
recording	2
record	2
self-monitoring	2
reminder	2

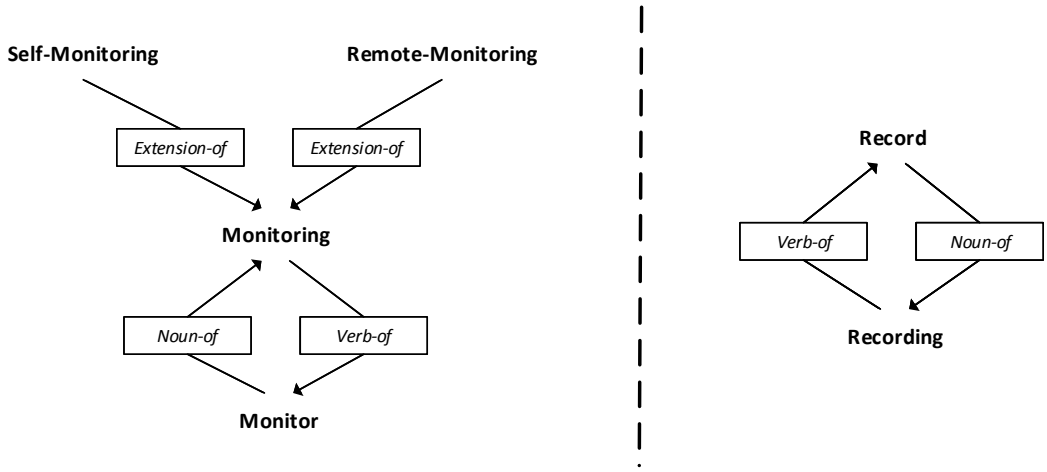


Figure 4.3 Examples of concept maps showing the relationship between a common codes

Phase Two: Identify Synonyms

The goal of phase two was to further reduce the number of concept maps and begin to define further relationships by identifying codes that are synonymous (words that have the same or similar meaning as one another). As can be seen in Figure 4.4, four independent concept maps have now merged together. Again this helps towards achieving usefulness & utility attribute defined in Section 3.2.1.1.

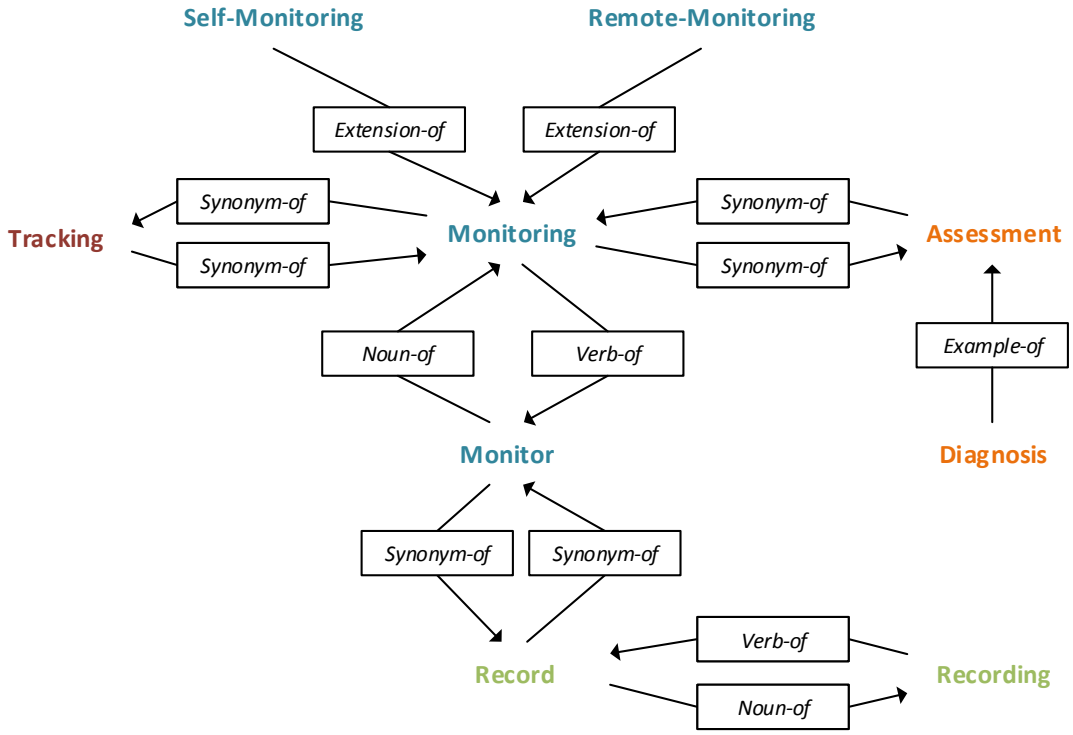


Figure 4.4 Concept map - Synonym relationship

Phase Three: mHealth Functions

The objective of phase three was to establish relationships between codes that represent examples or components of mHealth functions. During this stage in the analysis, it was critical to check that each code was exclusive to a single set as this ensures that the sets remain mutually exclusive from one another, as discussed in Section 3.2.1.1. At this stage in the analysis, the 15 individual codes formed 4 mutually exclusive sets, labelled *A to D* as illustrated below in Table 4.4. These sets along with the associated semantics provide the foundations of the coding framework which is used to help process the remaining data.

Table 4.4 Sets within the initial coding framework

Set A	Set B	Set C	Set D
tracking	management	communication	education
assessment	reminder		information
recording	self-management		
monitor			
record			
self-monitoring			
monitoring			
remote-monitoring			
diagnosis			

Phase Four: Coding Framework: Further Development

The purpose of this stage was process the remaining data from both datasets and further develop the coding framework. Following the same methods as described in the previous section, the remaining data from the articles data and then the application dataset was processed. Through each iteration, new semantics were added, refining and refactoring the coding framework, again taking care to ensure that each code remained exclusive to a single category. Figure 4.5 shows a comparison between the entities belonging to *Set B* throughout the different stages of development.

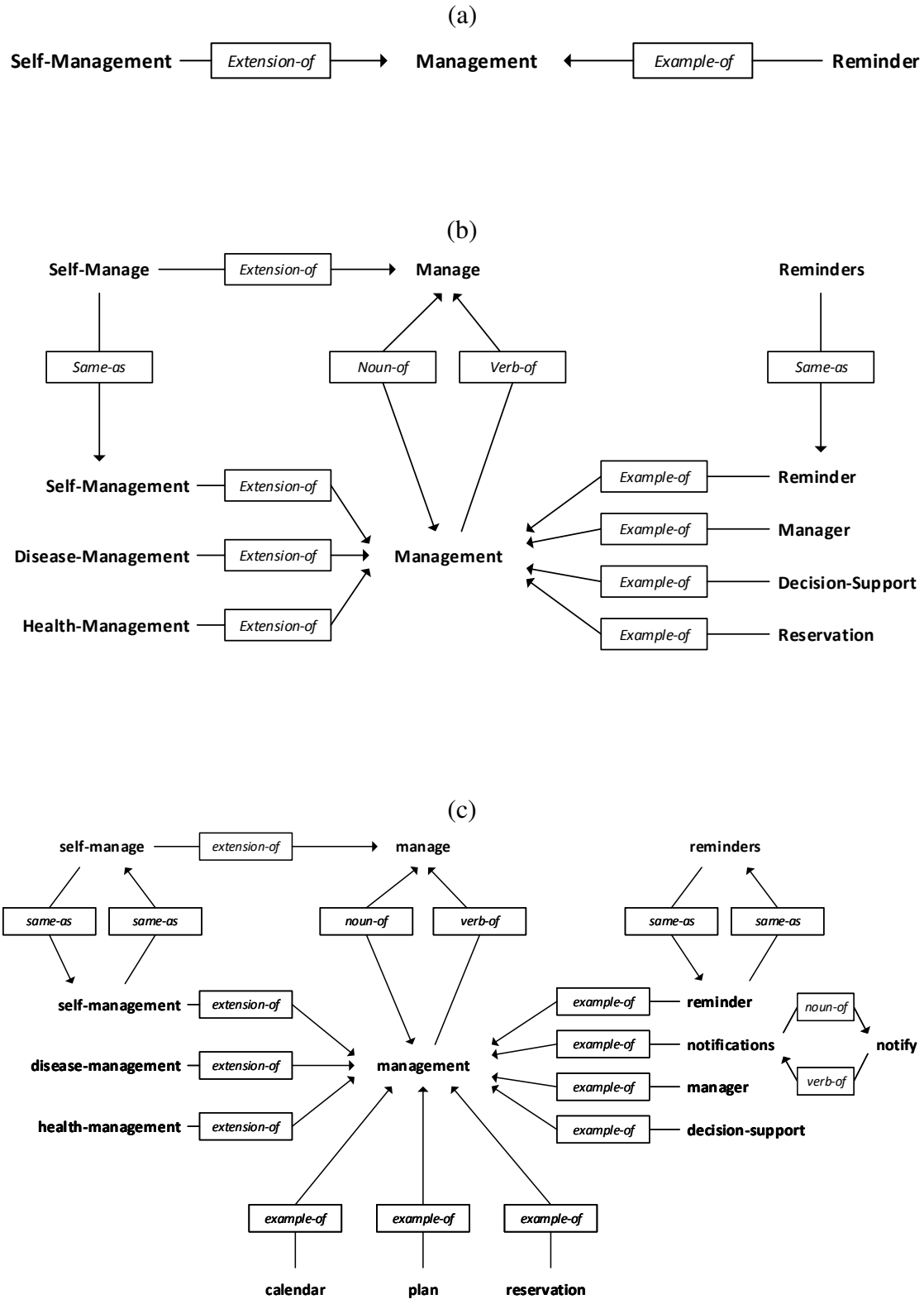


Figure 4.5 Comparison of SET B concept maps through varying stage of analysis: (a) Initial, (b) Article Data, (c) All data

4.3.3 Establishing Categories

The final stage of the analysis process to establish a suitable name and definition of each category within the coding framework. The process in described throughout the subsequent sections.

4.3.3.1 The Importance of Naming Each Set

PageRank is a link analysis algorithm that is used to measure the relative importance of a web-page within a web-graph [118]. A web-graph is directed graph constructed of nodes and transitions A node represents a single web-page, whilst a transition defines how the node is connected to a neighbouring node. Transitions can have one of two properties: in-degree (inbound link) and out-degree (outbound link) in relation to a particular node. The basic principle of the PageRank algorithm considers each in-degree to a page as-a vote. A web-page is generally considered of greater importance if the sum of all its votes is higher.

Although PageRank is commonly used within information retrieval, the principle has been applied to this work in order to name each set within the coding framework. Furthermore, by applying this technique when naming the categories ensures that the taxonomy can still evolve and be refactored in the future. In the context of this work, the structure of the concept map is substituted to form a web graph. As shown in Figure 4.6, codes in the concept map represent nodes, while the relationships represent transitions. For example, information has 4 *in-degree* and 1 *out-degree* transitions.

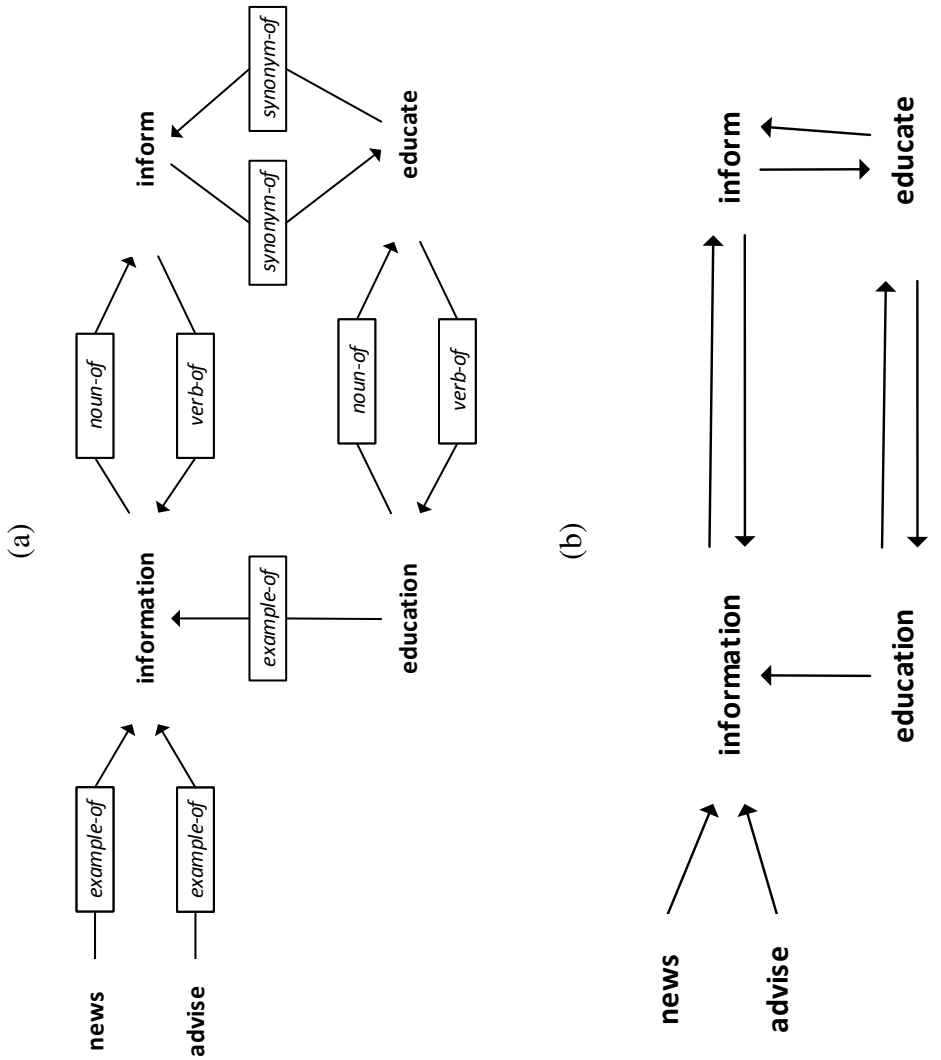


Figure 4.6 Demonstrating the structure similarity between a (**a**) concept map and (**b**) web graph

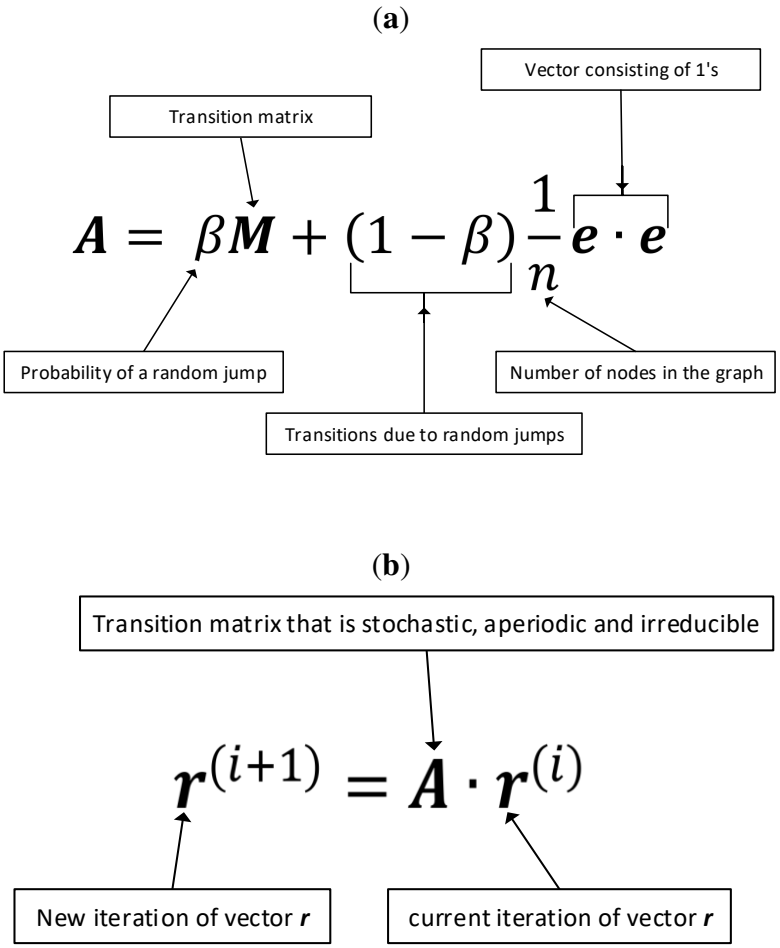


Figure 4.7 PageRank: Matrix notation - (**a**) expression used to calculate matrix A, (**b**) expression used to calculate PageRank

The PageRank algorithm is calculated using a simple iterative algorithm that can be expressed using the following matrix notation as shown in Figure 4.7. It is assumed that during the initial iteration r^0 of the algorithm, the PageRank value for each node is uniformally distributed, meaning that each node begins with the PageRank value $\frac{1}{n}$, n being the number of nodes within the graph. The algorithm continues calculate the PageRank values until the data reaches a state of convergence. To compute the PageRank scores for

each set with the coding framework a python script was developed (see, Appendix A.3) that implemented the PageRank algorithm. Table 4.5 provides an summary of the PageRank process for each set. For each set, the table identifies the most important code and the Page Rank value it received as well as the number of iterations required to reach a state of convergence and the number of codes that belong to it.

Table 4.5 Table showing a summary of the PageRank process for sets A, B, C & D

Set	**Most important code**	**PR**	**# Iterations**	**# Codes**
A	*monitoring*	0.24	14	22
B	*management*	0.40	20	15
C	*communication*	0.43	17	8
D	*inform*	0.34	17	6

Figures 4.8 to 4.11 depict the PageRank values for each of the sets within the coding framework. Using Figure 4.8 as an example, you can see initially the PageRank values fluctuate. However, several iterations later, the data reaches a state of convergence. As can be seen in this example, the code *monitoring* has the highest PageRank score and is therefore considered the most important code within *Set A*. Thus *monitoring* is established as the name to represent functions that belong to *Set A*.

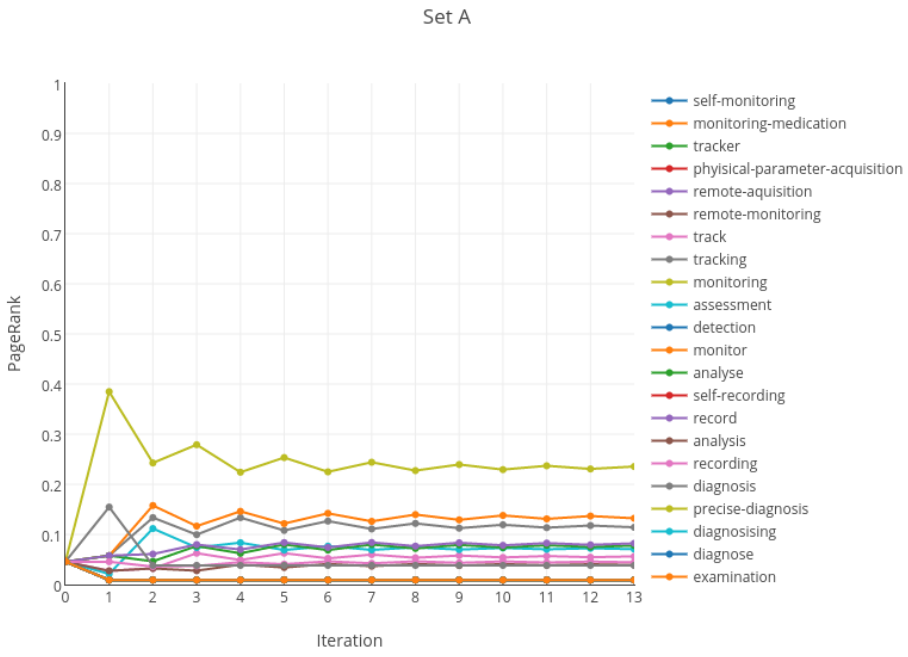


Figure 4.8 Set A - PageRank values

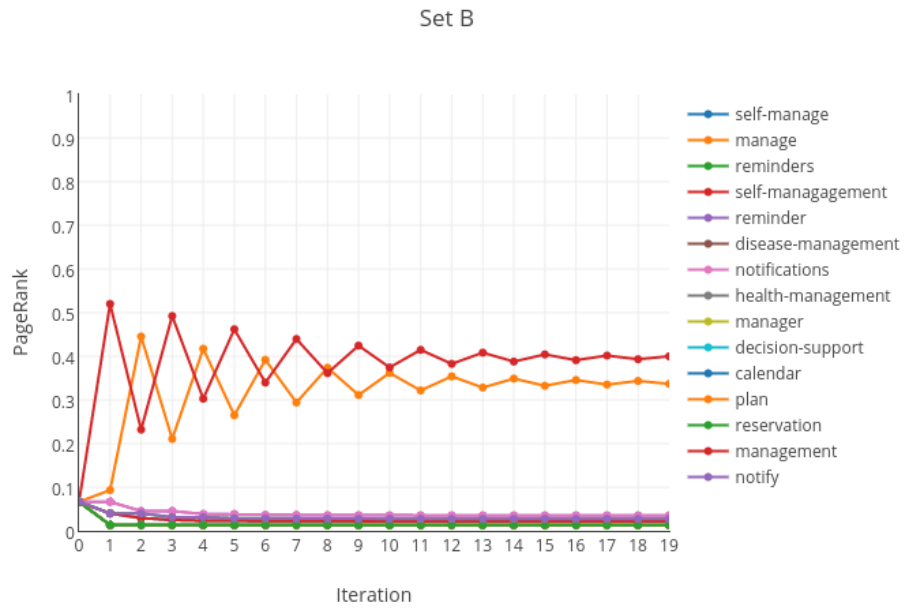


Figure 4.9 Set B - PageRank values

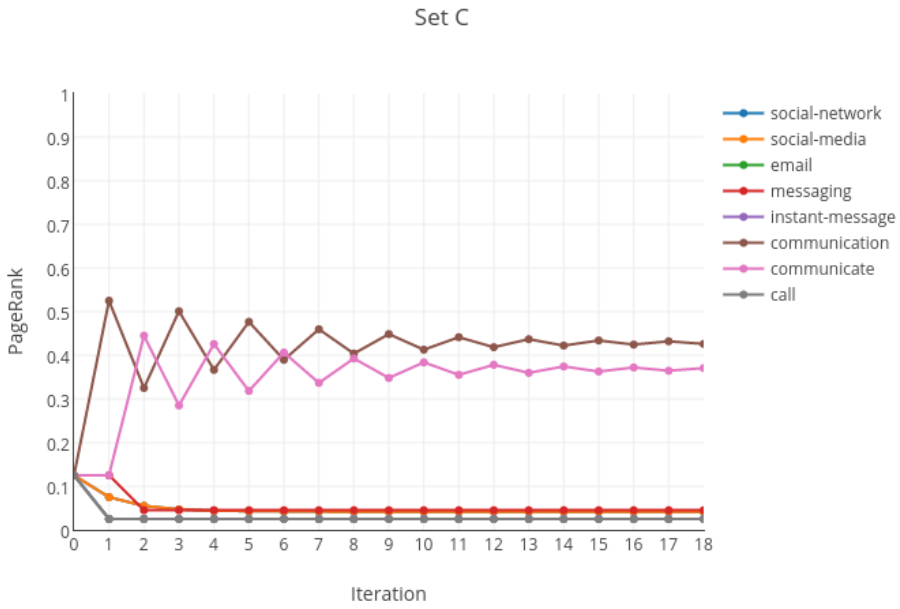


Figure 4.10 Set C - PageRank values

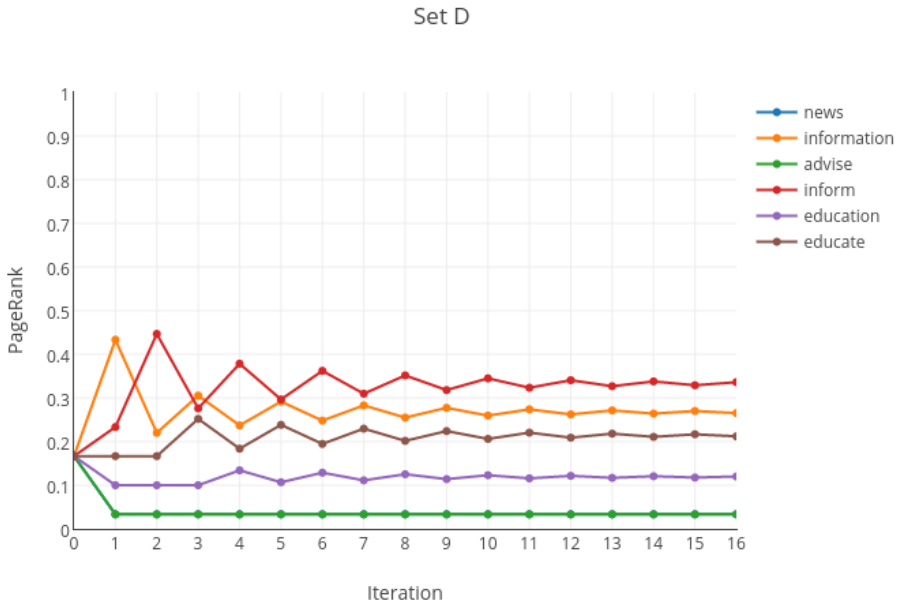


Figure 4.11 Set D - PageRank values

4.3.4 Sorting Functionality

The penultimate step of the analysis process required sorting each of the functions collected during the extraction and documentation phase, to assist in defining the categories: monitoring, management, communication, diagnosis and inform. The behaviour of each function was assessed in conjunction with the semantics defined in the coding framework to assign the function a type (see Appendix A.4). Figure 4.12 represents the distribution of the 213 mHealth functions that where surveyed.

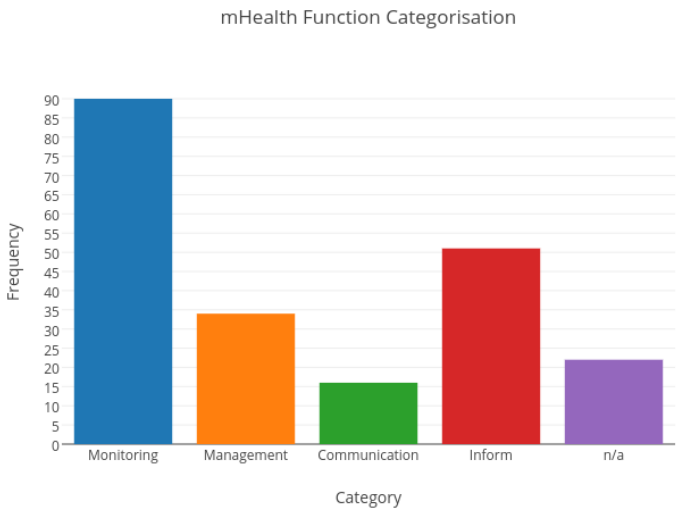


Figure 4.12 A bar chart depicting the frequency distribution of mHealth functions from both the article and application datasets

However, 22 of the functions could not be classified for one of the following reasons:

- *The function was derived from an article that didn't provide sufficient context regarding the functions behaviour*
- *The function was 'locked' as it was a premium feature within a free/lite version of the application thus fell beyond the scope defined earlier in Section 4.2.1*

The remaining 191 functions included within this study were successfully categorised. Each category is discussed below:

Monitoring:

Approximately 47% of the functions assessed where categorised as monitoring, making it the most frequently used function within the scope of this work. Monitoring functions are often used in conjunction with *Inform* functions as a mechanism to visualise the data for the user. The page rank analysis also reviled two distinct forms of monitoring functions.

- ***Tracking***: enables a user to observe, collect and store data regarding a specific aspect of their health. Examples include, tracking number of steps, blood glucose monitoring and heart rate.
- ***Examination***: Focuses on assessing an aspect of the user's health to identify the nature of an illness or other health related issue. These typically required data from the user, such as a responding to a series of questions, followed by an assessment to reach a conclusion, such as a diagnosis.

Management:

Management functions are designed to assist the user in managing an aspect of their health. Out of all the function categories identified, management is the most diverse. In the applications surveyed, management functions existed in various forms, examples included: convenient calculators, others provided reminders to the user to take their medication or the ability to create a shopping list. Interestingly there were instances that used games and challenges to positively influence the users health.

Inform:

As can been seen in Figure 4.12, inform functions are used frequently throughout mHealth applications. The intent of this type of function is to provide the user with health-related

information. In the applications surveyed information existed in five distinct forms: informative, educational, advisory, instructional and statistical.

- ***Informative***: Many applications utilised informative functions to provide the user with news on current events.
- ***Educational***: As the name suggests, this type of function provides the user with information that is designed to educate the user of a particular aspect of their health.
- ***Advisory***: Advisory functions offer suggestions to the user to positively influence and aspect of their health.
- ***Instructional***: The nature of instructional functions is to provide the user with step by step instructions regarding a specific aspect of their health. This was often in the form of text, images and video.
- ***Statistical***: Many applications utilised statistics to present data to the user. As mentioned earlier this type of function is often used in conjunction with *monitoring* function to present data to the users

Communication:

As the name of this category suggests, the objective of this type of function is to provide the user with a mechanism to communicate. In the applications that were observed, email, call and instant messaging where all common methods used by mHealth applications, often used to contact health care service providers. Moreover, applications also used social media as a channel for communication.

4.3.5 Handling Unknown Codes

During the earlier stages of the analysis process, several codes shown in Table 4.6, could not be initially assigned to a set and required further attention. Therefore, the final step in the analysis process aims to utilise the newly acquired definitions to place them into the relevant categories.

Table 4.6 Codes requiring additional work

Unknown			
user-profile	tools	medical-reference	suggestions
location	view-progress	calculate	charts
counter	find	locate	logs
goals	graphs	guidance	history
personal-profile	results	set-goals	tips
medical-measurement-devices			

Upon initial review, the codes required context from its original source, this information was aggregated to form a new table. Using the contextual information and the coding framework relationships could then be established enabling the unknown codes to be categorised, as shown in Figure 4.13. In this example the code *result* when combined with its context, was able to define a relationship with code *tracker*. Therefore, the assumption can be made that *result* is-a function that is an example of a tracker.

Yet, despite the additional work to establish relationships of these particular types of codes, the following codes where disregarded as they only occurred once within the datasets and for the reasons defined in the table Table 4.7.

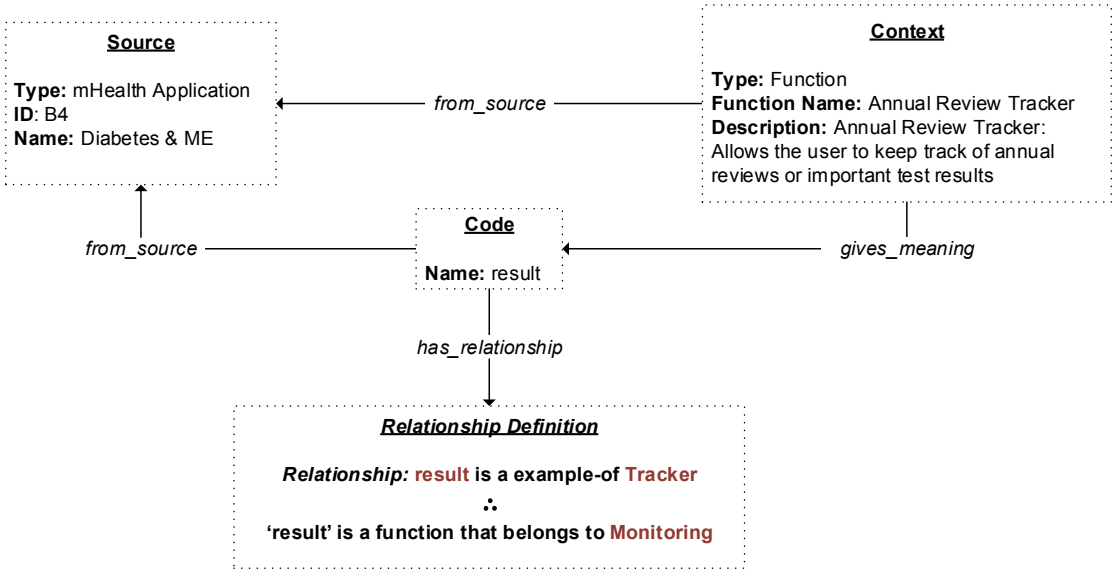


Figure 4.13 Reasoning process behind the definition of a relationship for the code *result*

Table 4.7 Codes disregarded and justifications

Code	Frequency	Justification
Tool	1	Used to describe a broad spectrum of mHealth application functions.
Medical Measurement Devices	1	Refers to additional devices/sensors that are capable of monitoring the users physical parameters and not the specific functionality.

4.3.6 Observations

Presented in the subsequent sections is a discussion surrounding some of the observations made during the analysis of the mHealth applications.

4.3.6.1 Personalisation Spectrum

After analysing and classifying functions it was evident that functions from a personalisation perspective existed in two forms: generalised and specialised. At first glance, functions within a single mHealth application appear to be all designed for a specific healthcare scenario. However, surveying a large collection of applications revealed that many functions provide the same or very similar service to the user. For example, medication tracking was a popular function found in the applications surveyed. Although, they typically tracked medication that related directly to the context of the application. From a personalisation perspective the service logic was fundamentally the same, it was the service data/content (personalisation component) that made the service they provided to the user unique. Functions such as this can be considered as general functions since they can be used in a large variety of application scenarios. At the other end of the spectrum, functions were the service logic and service data are designed specifically for a single healthcare scenario for example *Peak Detection* function provide a very narrow margin for personalisation and can only be used in very specific and specialised contexts, in the case of the example atrial fibrillation screening. Therefore these type of functions are considered as specialised functions.

4.3.6.2 Composition of a Function

The second observation, extends slightly upon the first and reflects specifically on the composition of a personalised mHealth function. The previous observation identified two core components of a function. At the heart of a function is the logic. The logic represents the code necessary in order to perform a specific task. From a personalisation perspective, the logic of function can be tailored to the user via personalisation components. Personalisation components represent the specific feature or features of a function that can be personalised, this may include data or user consumable content. The final component of a function relates to its specific hardware dependencies. There were cases throughout the application surveyed that require a mobile device to have a specific hardware feature present in order for the function to operate. Examples from the applications surveyed included: *Find Nearest A&E Service* which required location based hardware such as a GPS to function; *Monitor Blood Pressure* that required Heart rate sensor and *Call NHS* required telephony hardware in order to make a call. From a mobile application development perspective understanding each of these components is vital when designing personalised mHealth applications.

4.4 mHealth Application Function Taxonomy

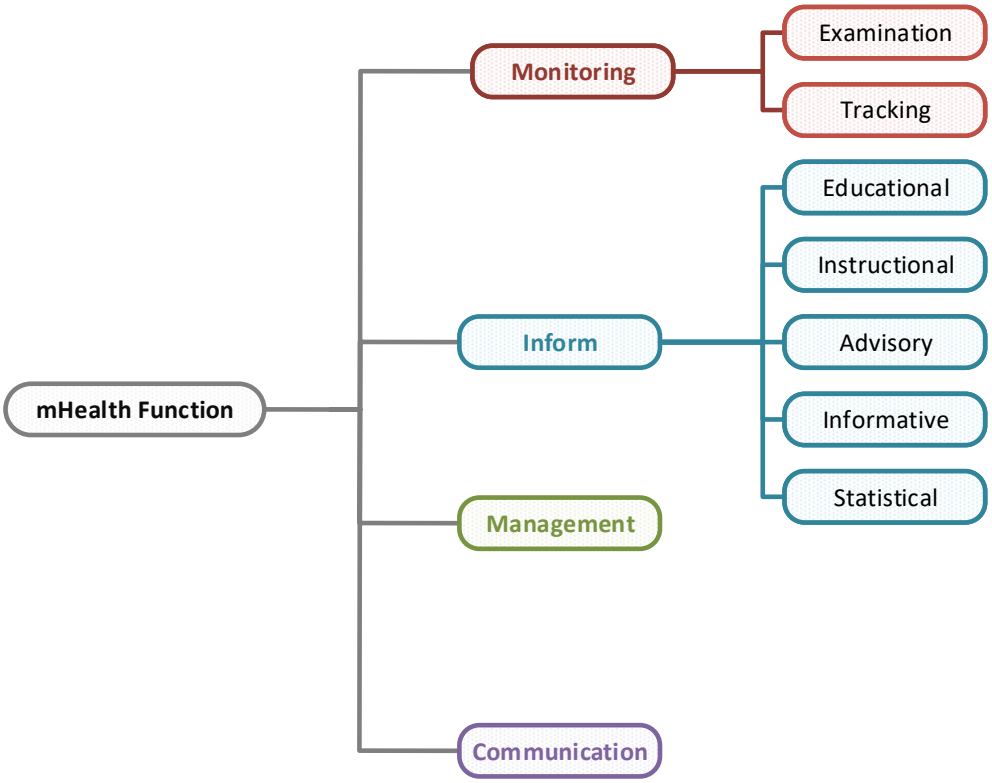


Figure 4.14 mHealthe application function taxonomy

The mHealth Application Function Taxonomy (mHAFT) was developed as a tool that can be used to analyse, identify and categorise the various types of health related functions that are available within mHealth applications that are intended to be used by healthcare consumers. As shown in Figure 4.14, the taxonomy is constructed of 4 dimensions, each representing a distinct type of service that a function provides to a healthcare consumer and are defined in Section 4.4.1.

4.4.1 Class Definitions

This section presents the definition for each class within the mHealth Application Function Taxonomy.

Monitoring: A function that allows the user to monitor an aspect of their health.

- **Assessment:** A function that assess an aspect of the user's health status to identify the nature of an illness or other health related issue.
- **Tracking:** A function that enables a user to observe and collect data regarding specific aspect and of their health.

Inform: A function that provides the user with health related information.

- **Educational:** A function that provides educational information to the user
- **Informative:** A function that provides Information that provides the user with news on current events.
- **Instructional:** A function that provides instructional information to the user.
- **Advisory:** A function that provides advice to the user.
- **Statistical:** A function that provides statistical information to the user.

Management: A function that is designed to assist a user with managing an aspect of their health.

Communication: A function that provides a user with a mechanism to communicate.

4.4.2 Classification Tool

Figure 4.15 is a classification tool that has been developed alongside the taxonomy to assist user's when categorising mHealth functions. The tool has been designed to systematically classify mHealth functions. It ensures that the taxonomy is being used correctly, removes potential ambiguity, ensures that a function only receives a single classification and identifies potentially new categories. As can be seen, the tool consists of a series of simple questions, which the user is required to answer. The answer to each question will lead to one of three outcomes: the user is required to answer another question, determine the type of mHealth function or identify a potential new function category.

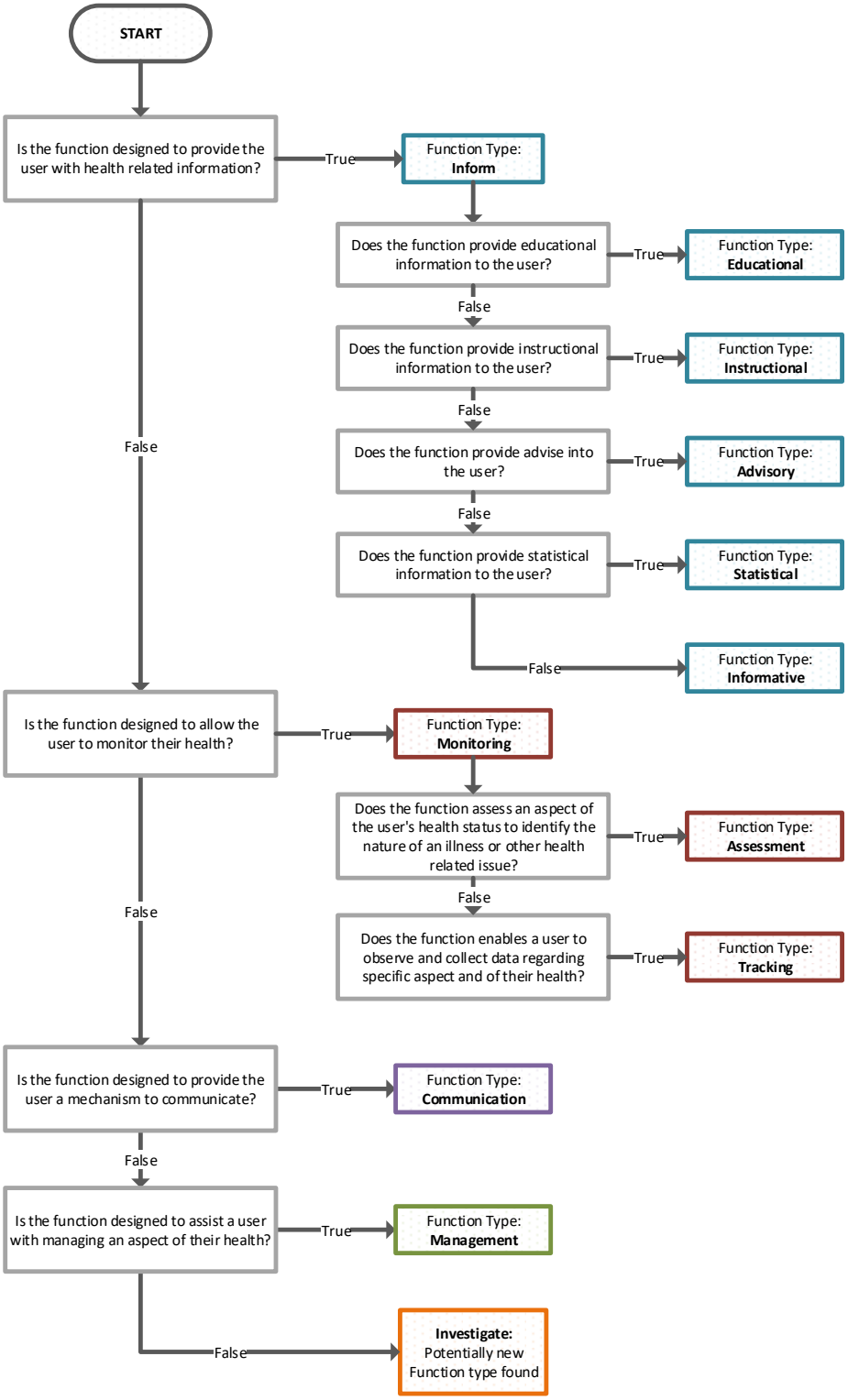


Figure 4.15 Flow chart for classifying functions

4.4.3 Maintenance

To ensure that the mHealth Application Function Taxonomy remains comprehensive, it is critical that it is maintained. A vital aspect during the development was to utilise techniques that enable the taxonomy to evolve as more data is analysed and processed or when a potential new function or subcategory is identified. As a result the procedure shown in Figure 4.16 is a modified version of the development processed followed to create the initial revision of the taxonomy. The maintenance process is designed to build upon the existing relationships.

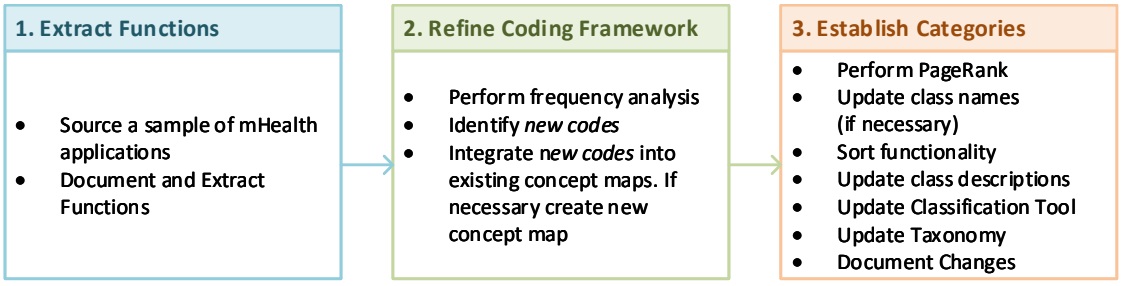


Figure 4.16 Maintenance procedure for the taxonomy

1. **Extract Functions:** As with before, a selection of mHealth applications that met the criteria described in Section 4.2.2 would be required and be subjected to the same extraction and documentation processes.

2. **Refine Coding Framework:** Next would be to refine and refactor the the coding framework, identifying any new codes and integrate them into the existing concept maps. If for example a new, type of function is identified this would require the creation of new concept map.

3. **Establish Categories:** The third phase in the maintenance process requires following the steps described in Section 4.3.3. From a comprehensive perspective this is vital step to ensure the taxonomy is remains representative of current the trends and

themes. To accommodate changes to the taxonomy all functions including those previously assigned a category will require classification. The final step would be then to update the class descriptions, classification tool and taxonomy. Again, this process must be documented.

By following the maintenance process described above helps ensures that new data is processed the same, but also helps quality assure the information prior to revising the taxonomy.

4.4.4 Construction of Use Case Scenarios

The development of the taxonomy also provided the opportunity to construct a collection of use case scenarios, that are based upon themes and trends found within the literature and applications surveyed. The objective of the use case scenarios is to simulate a potential scenario of a healthcare consumer that requires a personalised mHealth application. A summary of the contents of each use case profile is presented below:

- A narrative describing the current health status of the healthcare consumer.
- An outline of a personalised healthcare plan.
- The Healthcare consumers mobile device[2].
- A description of the required functions[3] for a personalised mHealth application.

[2] All device used in the use case scenarios are Android devices.
[3] Functions extracted from mobile applications were used in the use case scenarios

4.5 Testing and Evaluation

The following sections provide details surround the testing procedure and evaluation aspect of the mHealth Application Function Taxonomy. Section 4.5.1, describes how the test data was collected and the procedure it was subjected to. Section 4.5.2 discuses the results and evaluates the taxonomy against the criteria defined by Nickerson (see Section 3.2.1.1).

4.5.1 Test Data Collection and Procedure

Using the same inclusion criteria defined in Table 4.1, a sample of 10 mHealth applications (Taxonomy Test Dataset) were chosen at random. Each application is then subjected to the following process:

1. Record the Applications name, platform, description.
2. Download and install the application on the appropriate device.
3. Observe, describe and document the functions.
4. Attempt to categorise each function using the method described in Section 4.4.2 record the results.

The complete test dataset (and results) are available in Appendix A.5.

4.5.2 Tests and Evaluation of Results

The following discusses the results and each of the 5 criteria described by Nickerson.

Comprehensive: *Can the taxonomy classify all current mHealth functions?*

With regards to the comprehensive nature of the taxonomy, its not practical nor feasible to test all available mHealth functions. However, to but things into perspective, the taxonomy and its components are the product of the analysis of 213 mHealth applications functions, representing trends from both literature and currently available mHealth applications. In total the taxonomy has successfully classified 191 out of 213 (22 were excluded from the classification process for reasons stated in Section 4.3.4) functions during its development and a further 63 functions during testing(please refer to Figure 4.17 for results). At no stage during the development or testing of the taxonomy has a function not been able to be classified using the classification tool, suggesting that the taxonomy is indeed comprehensive. Furthermore, it is acknowledged that this is a rapidly developing area, as a result the taxonomy has been developed using techniques that can facilitate evolution when new data is processed, additionally the classification tool has been designed to also identify potentially new function types and subtypes to help ensure the taxonomy remains comprehensive as time progresses.

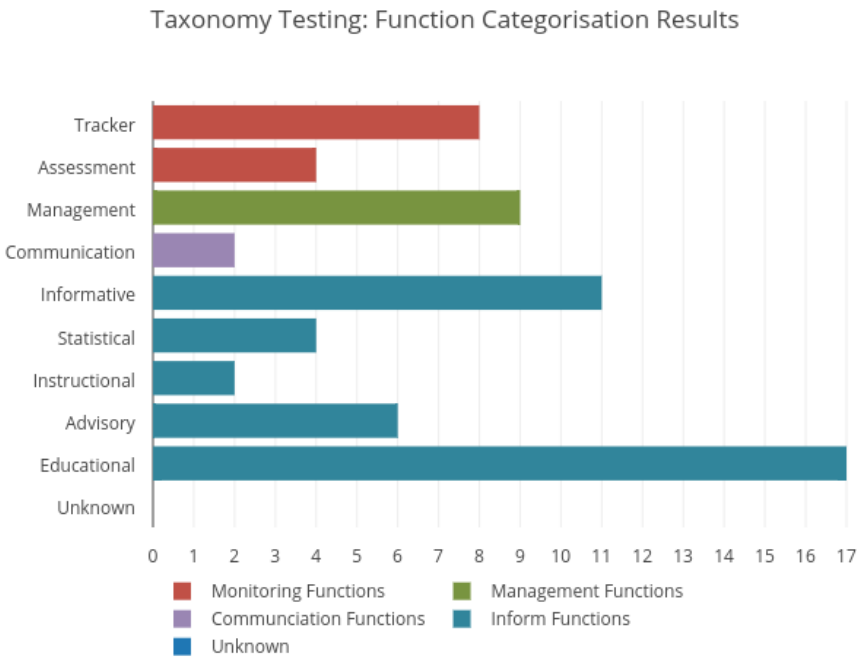


Figure 4.17 A bar chart to show the classification results of the test data used to test the taxonomy

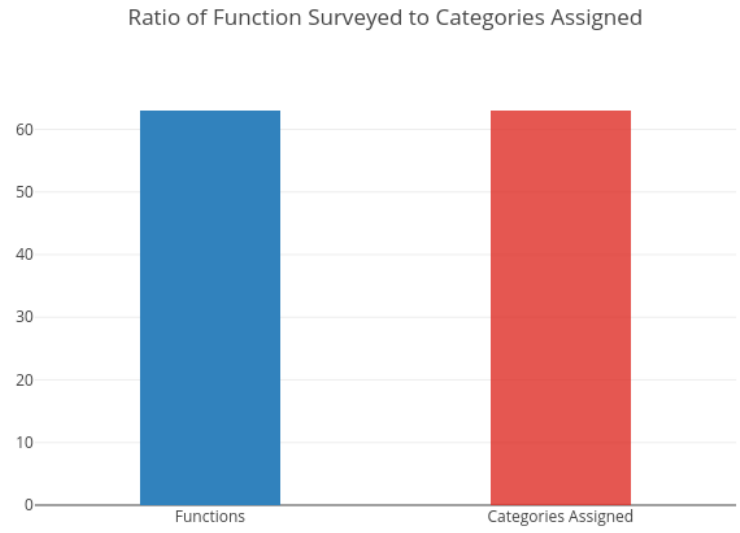


Figure 4.18 A bar chart to show the comparison between the number of functions vs the number of categories.

Mutually Exclusive: *No function falls into more than one category.* A simple test was conducted that compared the number of functions against the number of categories assigned during the testing of the taxonomy. As can be seen in Figure 4.18, the number of functions (63) is equivalent to the number of categories assigned (63) during testing. This infers that no function was assigned more than one category. Suggesting that the classes are mutually exclusive. This is also reinforced by the sequential design of the classification tool which prevents a function being assigned to more than one category. Therefore classes within the mHealth Application Function Taxonomy are considered to be mutually exclusive.

Extendibility: *Does the taxonomy allow for additional dimensions and new characteristics within a dimension when new types of objects appear?* In addition to taxonomy, the primary purpose of Figure 4.15 is to assist a user when classifying functions, its secondary purpose is also identify potential new categories as they emerge. If a new category is discovered, further investigation into its characteristics would be required prior to establishing it as a new distinct category or sub category. The maintenance process was designed to support new iterations of the taxonomy, by following the procedure shown in Figure 4.16 not only allows new data to further refine existing categories, for example renaming of a class. But once there is sufficient evidence to define the new category, the taxonomy and the classification tool can be updated to reflect the newly processed data or function type.

Concise: *Is the taxonomy succinct?*

With regards to the concise nature of the taxonomy, Table 4.8 represents various characteristics of the mHealth Application Function Taxonomy. As can be seen in the table, the four main types of mHealth functions were constructed from 55 unique codes. This allowed 191 of the initial functions to be classified. Both monitoring and inform have child classes that are associated to them. This is because there was significant evidence that they provided a distinctly different service to the user. However, Management and Communication do not. Communication functions although enable the user to commu-

nicate through different mechanisms, they fundamentally only provide a communication service to the user. By adding child classes to communication for example *Call*, *Email* and *Instant Message* would not provide any further benefit to the taxonomy. As these are considered as instances of communication and could indeed affect the the concise nature of the taxonomy. Management functions on the other hand are quite the opposite. As already mentioned Management functions were the most diverse. This made it difficult to identify distinct subclasses using the data that was initially collected. However, this is something that could be addressed future revisions of the taxonomy.

Throughout the development of the taxonomy careful consideration was taken to remove any potential for ambiguity. Class definitions are succinct, short sentences. The classification tool utilises a flow chart and short true or false questions to guide users through the classification process. The benefits of these decisions during the development can be seen in the results from the test data. Figure 4.17 shows the percentage frequency distribution of the test data. As can be in the test data provided a range of functions that covered all aspects of the taxonomy. Overall the details provided above along with the results from the test data suggest that the taxonomy is concise.

Table 4.8 mHealth application function taxonomy characteristics

	Monitoring	**Management**	**Communication**	**Inform**
Number of unique codes during development	22	15	8	6
Number of functions classified during development	90	33	17	51
Number of subclasses identified	2	0	0	5
Number of functions classified during testing	12	9	2	40

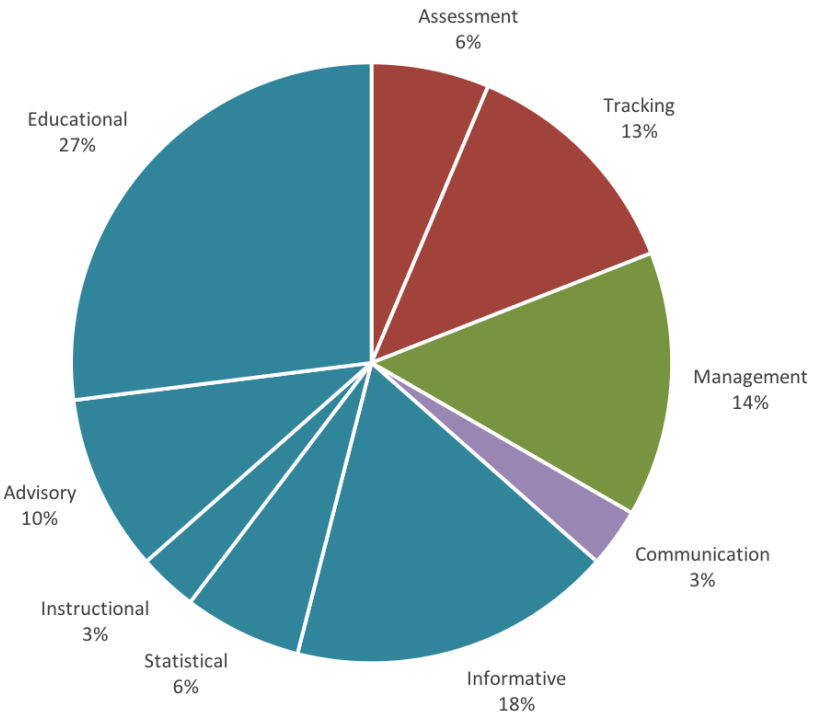


Figure 4.19 Pie chart representing the percentage frequency distribution of mHealth functions within the test data

Inclusive: *Does the taxonomy contain enough dimensions and characteristics to be of interest?*

The inclusive attribute is difficult to quantify since it was primarily developed for use within the scope of this research. However, the taxonomy has several applications both within and out of the scope of this research. The taxonomy, with regards to this research has several uses. The first use was to obtain an understanding of the types of health related functions that are designed to be used by healthcare consumers, including their associated characteristics that make them distinct. This is critical in achieving the second objective of this research. As discussed in Section 4.4.4, the taxonomy and the data associated to it was also used to develop use case scenarios that are to be utilised, during the development of the ontology and evaluation of the framework.

Moving outside of the boundaries of this research, the taxonomy also can be utilised from a practical standpoint during the development of a mHealth application. The taxonomy along with the classification tool can also be utilised by mobile application developers to identify the types of healthcare related functions required for a specific application during the design phase, since the taxonomy is based upon the principles of a functional hierarchy diagram.

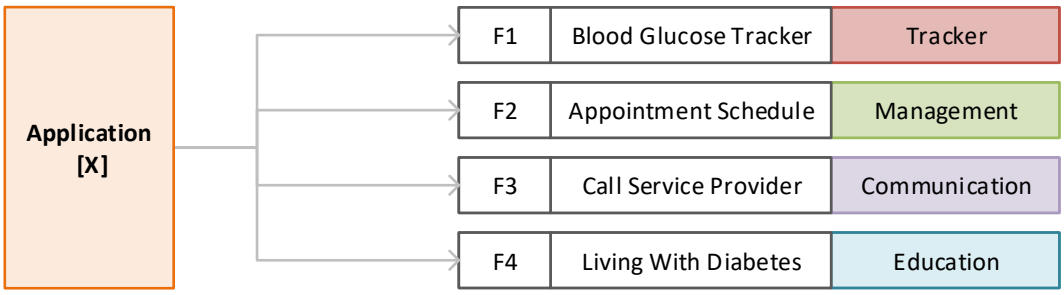


Figure 4.20 Demonstration of the practical use of the mHealth application function taxonomy during the development of a mHealth application

4.5.3 Results Synopsis

The taxonomy developed as part of this research is representative of the trends and themes found within the literature and from the United Kingdom's two most popular platforms[4] app stores. The testing and evaluation of the taxonomy has demonstrated that the taxonomy is capable of classifying health related functions that are found in mHealth applications that are designed to be used by healthcare consumers. Furthermore, the results obtained from the testing and evaluation of the mHealth Application Function Taxonomy provides sufficient evidence to suggest that the taxonomy in its current from possess all of the attributes described by Nikerson.

4.6 Summary

In summary, this chapter presents the development considerations for the creation of the mHealth Application Function Taxonomy, a tool that is capable of categorising health related functions within mHealth applications designed specifically for healthcare consumers. The mHealth Application Function Taxonomy was developed as part of this research was to gain a solid understanding of the trends and themes of functions and the distinct attributes that made them unique. The analysis of the functions also provided an insight into the composition of a function from a personalisation perspective, leading to the creation of the generic function model. The product of which, led to the construction of several use case scenarios that simulate healthcare consumer personalised requirements for a mHealth application. These scenarios are to be utilised throughout the key stages of development in both the PMAD Ontology and PMAD Framework. The testing of the taxonomy was conducted using functions extracted from a random sample of mHealth applications. Using the attributes defined by Nikerson as evaluation criteria, both the

[4] Android and iOS

taxonomy and results were then evaluated. The outcome from the testing and evaluation of the results demonstrated that the taxonomy is capable of classifying health related functions that are found in mHealth applications. Furthermore, the evaluation of the mHealth Application Function Taxonomy provided sufficient evidence to suggest that the taxonomy in its current form, possess all of the attributes of a good taxonomy as described by Nikerson. Overall, the development of the taxonomy has satisfied the requirements of Objective (b) of this research.

Chapter 5

PMAD Ontology

The objective of this chapter is to present the theoretical, design and development considerations for the Personalised Mobile Application Development Ontology (PMAD Ontology). The chapter is composed of two parts. The first part presents the theoretical considerations, exploring what an ontology is in computer science and how it differs from its traditional origins of philosophy. This section also discusses the relationship between various semantic models and how they can be used to develop an ontology and how knowledge is represented by ontologies represent knowledge. The theoretical considerations conclude with a discussion of ontology languages and tools used during the development process. The second part of this chapter provides a detailed insight into the design and development process of the PMAD Ontology. The development process follows the skeletal methodology of Uschold and Grunninger and begins with an overview of the development process, the guidelines that were established and scope and purpose of the ontology. The implementation is divided into two stages; the first is associated the capturing and organisation, the second focuses on the encoding of the knowledge captured. The penultimate section presents an overview of the PMAD Ontology. The chapter concludes with a summary of the key points raised.

5.1 Ontology Theory

5.1.1 Ontology Definition

Many of the works discussed in Section 2.2.3.3 successfully implemented an ontology to provide intelligent and personalised services to users of mobile applications. Ontologies are also heavily utilised in areas such as artificial intelligence, web technologies and information systems. Their ability to address challenges such as interoperability and knowledge sharing has seen the interest and application of ontology in the realm of computer science increase tremendously in recent years. To help understand what an ontology is in computer science, it is worthwhile contemplating its original application within the area of philosophy. In its original context, the term ontology refers to the "science of being" [119]. Philosophers strived to understand and systematically model abstract concepts that related to the meaning of existence, its prerequisites, conditions, origins and future horizons. In other words philosophers questioned what it means to exist [104, 105, 120].

During the 1990's computer scientists began to recognise the potential of ontology and began implementing ontologies in areas such as artificial intelligence, natural language processing and knowledge-based systems [121, 122]. However, unlike the philosophical interpretation of an ontology, computer scientists the time adopted a naïve realistic approach [104]. Meaning researchers choose only to denote a systematic representation of only the necessary knowledge to perform a particular task. This body of knowledge is referred to as the universe of discourse and represents a set of named entities that described concepts and formal axioms that constrain the interpretation and use of these terms, forming an ontology [104, 105, 103].

Throughout the literature, there have been many attempts to define an ontology, the most widely accepted definition 'formal explicit specification of a shared conceptualisation' [103, 123] was proposed by Gruber. Although concise, the definition represents a series of requirements (see below) for the use of ontologies within the field of computer science.

- **Conceptualisation** is an abstract model about a domain, represented by objects, concepts and entities as well as the relationships that exist among them.
- **Explicit** Concepts and the restrictions applied to them are clearly defined.
- **Formal** The ontology must be defined using a formal language so it can be read and interpreted correctly by a machine (machine understandable).
- **Shared** Reflects the notion that an ontology captures consensual knowledge – it is not restricted to some individual but is accepted by a group.

5.1.2 Ontology and the Semantic Spectrum

However, throughout many different subject disciplines, an ontology has been portrayed using various semantic (knowledge-based) models, each model has varying degrees of complexity with regards to its structure and the semantics they represent. The *Semantic Spectrum* (see, Figure 5.1) was developed by McGuinness to compare the 'semantic richness' of such models as they increase in expressivity and complexity[125].

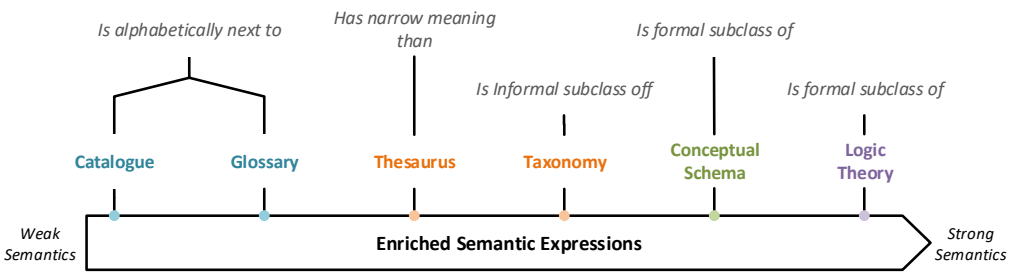


Figure 5.1 The Semantic spectrum [124]

A *Catalogue* and *Glossary* (shown in blue) are examples of simple controlled vocabularies, as they are basically a list of terms structured alphabetically. Whereas a *Thesaurus* and *Taxonomy* are considered as examples of complex controlled vocabularies (shown in orange) since they are more expressive with regards to the semantics they represent. Thesauri provide some additional semantics such as synonym, homonym, narrower than (NT), broader Than (BT) and related to (RT) relationships associated with terms within the domain[125]. As demonstrated in Section 4.4, taxonomies are based upon a glossary of terms and thesaurus and is primarily used for classification. Entities within a taxonomy form a hierarchical tree structure that represents the relationship between a parent and child. However, it is common that relationships defined in complex controlled vocabularies are informal, meaning there is often some ambiguity regarding the relationships they represent [124, 126]. A conceptual schema, on the other hand, formalises the relationships between the concepts. A conceptual model is a model of a domain that represents the primary entities of a domain, the relationships among them, the properties and property values of the entities. A conceptual model may also model rules that are associated with entities, relationships and properties [126]. Conceptual models are often represented using Entity Relationship diagrams (ERD's).

The aforementioned semantic models are all machine-readable, meaning the model can be represented in a format that a machine can read, such as XML. However, they are not machine-interpretable, meaning that they cannot be used by a machine to make valid inferences and enforce semantic constraints (axioms) [127]. In order for a machine to make valid inferences and enforce semantic constraints, the model must be described formally using logic theory (knowledge representation languages) such as First Order Logic (FOL) or Description Logic (DL). This is so the knowledge modelled is in a format that is both readable and interpretable by a machine, thus making it machine-understandable. Reflecting upon Gruber's requirements more specifically the *Formal* requirement, we can clearly see that an ontology (with regards to computer science) is restricted to the

logical theory end of the Ontology Spectrum [128]. Furthermore, McGuinness states for something to be considered an ontology within the field of computer science, that it must have least the following properties [125]:

- *Finite controlled (extensible) vocabulary*
- *Unambiguous interpretation of classes and term relationships*
- *Strict hierarchical subclass relationships (is-a) between concepts*

These three properties indicate the relationships and similarities between controlled vocabularies and an ontology [95, 124]. Table 5.1 represents attributes of each of the semantic models presented in Figure 5.1. As can be seen, each model has a specific purpose and presents knowledge from a domain differently with regards to its structure and semantics. Since the complexity (structure and semantics) of each semantic model increases, as we progress throughout the ontology spectrum, we can consider each of the semantic models to the left of the logical theories, as mechanisms to assist in solving complex problems throughout the development of a formal ontology [95, 128].

Table 5.1 Summary of semantic models

Model	Glossary	Thesaurus	Taxonomy	Conceptual Model	Ontology
Based on	Catalogue	Glossary	Thesaurus	Taxonomy	Taxonomy
Purpose	Define Terms within a domain	Information Retrieval	Classification	Enterprise Modelling	To capture and represent knowledge of a specific domain
Structure	List	Tree	Tree and Hierarchy	Hierarchy of knowledge	Relationships between categories
Relationships	• Term • Definition	• Synonym • Homonym • Narrower than • Broader than • Related to	• Parent • Child	• Entity • Relationships • Values • Rules	• Classes • Instances • Properties • Restrictions • Axioms
Presents	List terms and definitions	Presents the relationships between terms	Classification of concepts, terms and things	A Conceptual view of knowledge within a domain	Represent complex semantics of concepts and the relationships between them within a domain of discourse
Machine Readable	Yes	Yes	Yes	Yes	Yes
Machine Interpretable	No	No	No	No	Yes

5.1.3 Ontology Representation

The knowledge encapsulated in an ontology is constructed based on semiotic theory and consists of three overlapping areas: Syntax, Semantics and Pragmatics [129]. Each area is summarised in the list below.

- **Syntax:** Syntax is concerned with the rules (form, format and structure) used for constructing, or transforming the symbols and words of a language.
- **Semantics:** Is the analysis of the relationships between signs in reality.
- **Pragmatics:** Pragmatics is the study that relates signs to the agents who use them to refer to things in the world and to communicate their intentions about those things to other agents who may have similar or different intentions concerning the same or different things.

Again referring back to Gruber's definition, the purpose of an ontology is to share (communicate) a common understanding of a conceptualisation between agents. To help understand the meaning of *sharing a common understanding* first, requires understanding how we as humans communicate *meaning* in natural language.

The semiotic triangle (Figure 5.2), commonly referred to as the *triangle of meaning*, was developed by Ogden and Richard and illustrates the relationship between objects and concepts and the indirect relationship between terms and objects [130]. The lower left of the triangle represents terms that are bound by rules that are dictated by the syntax used (language). Terms can represent a single word, phrase or even a sentence. However, alone they have no meaning until associations are made with the other angles in the triangle. For example, if you were asked the meaning of the term 'AA45@SD', you would not be able to relate it to a concept in order to identify the object the term refers to. This is because concepts are a fundamental aspect of proposition. However, if the same question was asked

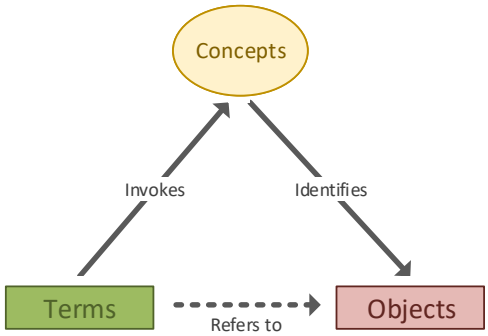
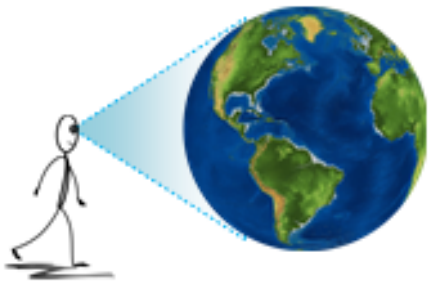


Figure 5.2 Semiotic triangle

about the meaning of the term ‘jaguar’ one could make semantic associations that refer to objects in the real world. For instance, the term ‘ jaguar’ could refer to a ‘car’, ‘animal’ or ‘the operating system’.

As illustrated in Figure 5.3a, the real world with regards to its scope is huge, hence why the term ‘jaguar’ is ambiguous and can be interpreted differently depending on the context and experiences (pragmatics) of an agent. This is often referred to as a weakly defined relationship. This is why in computer science an ontology only denotes the necessary knowledge of a conceptualisation required to perform a particular task as depicted in Figure 5.3b. Therefore, constrains the vocabulary, reducing how terms, concepts and objects can be interpreted. For example, if we constrain the view of the real world to just that of cars, then the term ‘jaguar’ in this context refers to a type of car and we are not concerned with the other interpretations of the concept ‘jaguar’. This not only reduces the ambiguity between agents but also strengths the logical relationship [131]. Because of this constrained interpretation of the world (universe of discourse), Guarino and Oberle revisited the semiotic triangle, modifying it to reflect upon the strengthened logical relationships and the constraints as shown in Figure 5.4.

This view of the semiotic triangle can also be used to illustrate the relationships between semantic models discussed earlier and an ontology. From a development perspective,



(a) Perspective of the whole world and everything in it

(b) Narrow perspective of only the necessary parts of the world

Figure 5.3 Perspective of a conceptualisation

the left portion of the triangle is associated with the semantically weaker (and limited) models such as a glossary, thesaurus and taxonomy. A glossary and thesaurus provide the vocabulary (terms) along with their semantics of the domain of discourse. A taxonomy allows the classification of terms. Transitioning to the right of the triangle, concepts within an ontology are organised in a strict is-a hierarchy, a structural property inherited from a taxonomy. It is at this point we can begin to distinguish between controlled vocabulary and an ontology. A controlled vocabulary such as a thesaurus only models simple relationships between terms [132]. Whereas an ontology represents a series of logical conceptual semantics (axioms) that are expressed in a logic-based knowledge representation language so that complex, accurate, consistent and meaningful distinctions can be made between classes, instances, properties and their relations [133].

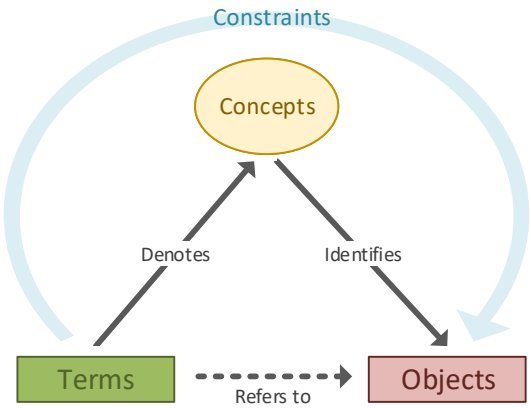


Figure 5.4 Semiotic Triangle - including the constraints of a domain [131]

Therefore, taking into consideration both Gruber's requirements of an ontology and the properties described McGuinness: we can say that an ontology in computer science is the product of engineering, consisting of a controlled axiomatic vocabulary, that consists of explicit interpretations of concepts arranged in a strict is-a parent-child relationship, their properties instances and relationships of a specific area of a specific domain.

5.1.4 Types of Ontologies

Ontologies are utilised to achieve many different objectives and as such can be classified based upon certain characteristics such as the generality of the conceptualisation, their coverage of a domain, intended purpose and structure [128, 133–135]. Guarino defines four types (see, Figure 5.5) of ontologies based upon the levels of generality (dependence on a particular task or perspective) [134]. The arrows in the diagram represent specialisation relationships, this is made clearer in each of the definitions in Table 5.2

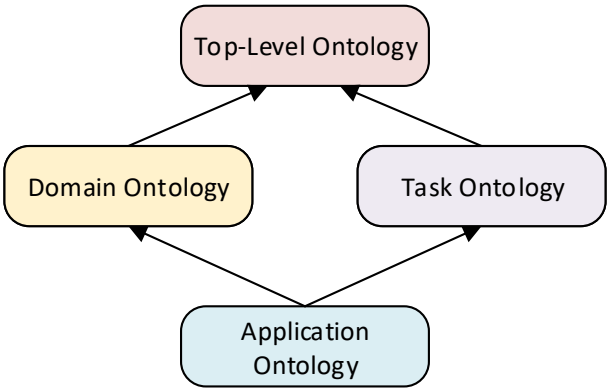


Figure 5.5 Guarino's ontology classification [134].

Table 5.2 Guarino's ontology classification definitions

Type of Ontology	**Definition**
Top Level Ontology	Describes very general concepts like space, time, matter, object, event, action, etc., which are independent of a particular problem or domain.
Domain Ontology	Describes the vocabulary related to a generic domain (like medicine or vehicles).
Task Ontology	Describes an ontology that relates to a specific task, such as diagnosing.
Application Ontology	Describes concepts depending both on a particular domain and task, which are often specializations of both the related ontologies. These concepts often correspond to roles played by domain entities while performing a certain activity, like replaceable unit or spare component.

5.1.5 Ontology Languages and Development Tools

When deciding to develop an ontology, there are several challenging questions that must be addressed: What *ontology language* to use to encode knowledge associated with the domain of discourse? Which *ontology tools* to use to facilitate the development of the ontology? [136]. Therefore this section provides an explanation of what an ontology language is and the purpose of ontology tools during the development of ontology. Also included is a discussion of the ontology language and tools chosen for the development of the PMAD Ontology.

5.1.5.1 Ontology Languages

As highlighted in Section 5.1.1, an ontology within computer science must be defined formally in order to enable a machine to interpret the knowledge within the ontology. An ontology language is a formal language used to construct ontologies. They provide mechanisms for creating all of the necessary components for encoding knowledge with the domain of discourse and often include reasoning rules that support the processing of that knowledge [135, 136]. A number of ontology languages exist such as Cycl, DAML+OIL and Web Ontology Language and can be categorised as logic based, frame-based or graph based [137]. Due to its increased interest in academic and commercial applications the language chosen to encode knowledge within the PMAD Ontology was the Web Ontology Language, or as it is commonly referred to as OWL.

5.1.5.1.1 Web Ontology Language

The web ontology language (OWL) is a family of semantic web languages for authoring ontologies. OWL is designed to represent rich and complex knowledge about things, groups of things and the relationship between them [138] and is the W3C's recommended

ontology language for the Semantic Web since 2004 [139, 140]. The design of OWL was influenced by three areas: Description Logics, Frames Paradigm and the Resource Description Framework [141].

- **Description Logics:** aimed to bring reasoning and expressive power to the Semantic Web. Description logics (DL) are a family of formal knowledge representation formalisms that are used to represent the knowledge of a domain [139]. The formal language constructs and language features of OWL are derived from description logics. These are discussed in context of OWL and in more detail throughout Section 5.4. A description logic knowledge base is composed of two components, TBox and ABox [139]. In the TBox the terminology of the knowledge base is defined and is associated with concepts and what they denote. Whereas the ABox contains assertions about individuals in the knowledge base. The architecture of a description logic system is shown in Figure 5.6.

- **Frames Paradigm:** OWL also provides a surface syntax based on the frames paradigm. Frames group together information about each class, making ontologies easier to read and understand, particularity for users not familiar with description logics [141].

- **Resource Description Framework:** To maintain maximum upwards compatibility with existing web languages, the OWL ontology language extends the syntax and semantics of the Resource Description Framework adding numerous constructs and semantics for describing properties and classes: among others, relations between classes, cardinality, equality, richer typing of properties, characteristics of properties, and enumerated classes [142].

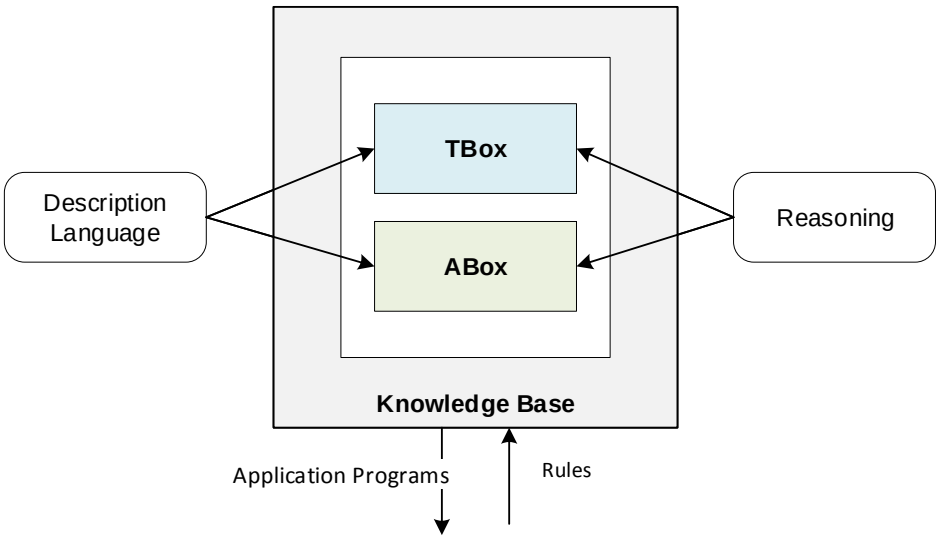


Figure 5.6 Description logic architecture: TBox and ABox

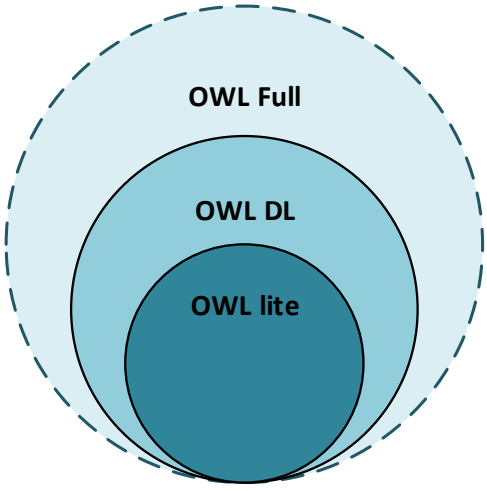


Figure 5.7 OWL species

OWL Sub-Languages

As a result of the various demands from specific communities and users of OWL, led to the specification of three 'species'; OWL Full, OWL DL (description logic) and OWL lite. Each sub language of OWL supporting has compromises between expressiveness and computational tractability [143].

- **OWL Full:** as indicated by the dashed lines in Figure 5.7, OWL Full has no expressiveness constraints, but also does not guarantee any computational properties. It is formed by the full OWL vocabulary, but does not no impose any syntactic constrains, so that the full syntactic freedom of RDF can be used.
- **OWL DL:** OWL DL is a syntactic subset of *OWL Full*, but supports those users who want the maximum expressiveness while retaining computational completeness[1] and decidability [2]. OWL DL includes all OWL language constructs, but they can be used only under certain restrictions.
- **OWL Lite:** is a syntactic subset of *OWL DL* but only supports classification hierarchies and simple constraints.

[1] all conclusions are guaranteed to be computable
[2] all computations will finish in finite time

OWL Components

The components of an OWL ontology are summarised in the table below. Each component is discussed in further detail in later sections of this chapter.

Table 5.4 Summary of OWL ontology components

Entity	Description
Individual	Also referred to as instances, individuals represent objects within the domain of discourse.
Classes	Represents a collection of objects that describes the precisely the requirements for membership of that class.
Properties	Are used to describe relationships, they exist in two forms: • ***Object Properties*** - object properties describe the relationship between individuals. • ***Data Type Properties*** - describe the relationship between individuals and a given data type.
Restrictions	Describe an anonymous classes that contains the individuals that satisfy the restriction.
Axioms	Are used for expressing propositions that are always true.

5.1.5.2 Ontology Tools

It is well documented throughout the literature that building an ontology is a complex and time-consuming process. With the rise of the Semantic Web and increasing academic and commercial interests of ontologies, saw the development of numerous ontology tools. Ontology tools are designed to help an alleviate a diverse range of challenges related to various activities associated with the ontology development process. Extensive work from Gómez-Pérez *et al.* identified six categories of ontology tools [122]. Each category is described in Table 5.6.

Table 5.6 Description of ontology tools [122]

Ontology Tools	**Description**
Development	This group includes tools and integrated environments that can be used to build a new ontology or edit an existing one.
Evaluation	They are used to evaluate the content of ontologies and their related technologies.
Merge and Alignment	These tools are used to solve the problem of merging and aligning different ontologies in the same domain.
Annotation Tools	With these tools users can insert instances of concepts and of relations in ontologies and maintain (semi) automatically ontology-based markups in Web pages
Query and Inference	These allow querying ontologies easily and performing inferences with them. Normally, they are strongly related to the language used to implement ontologies
Learning	They can derive ontologies (semi)automatically from natural language texts, as well as semi-structured sources and databases, by means of machine learning and natural language analysis techniques.

The following ontology tools, have been chosen for the development of the PMAD Ontology:

- **Protégé:** is a freely available open source ontology editor developed by the Center for Biomedical Informatics at the Stanford University School of Medicine [108]. It provides a rich frame based environment with full support for OWL and other ontology languages. Protégé also includes visualisation tools to interact with ontology throughout development, advanced support for tracking down inconsistencies and operations to modify the various ontology components. The version used throughout this research was Protégé 4.30.

- **Hermit Reasoner:** Developed by the Shearer *et-al;*, the Hermit reasoner is written using the OWL and can perform various automated checks including the consistency, satisfiability and subsumption of an OWL ontology [96] HermiT is the first publicly-available OWL reasoner based on a novel "hypertableau" calculus which provides much more efficient reasoning than any previously-known algorithm. Ontologies which previously required minutes or hours to classify can often by classified in seconds by HermiT. HermiT is also the first reasoner able to classify a number of ontologies which had previously proven too complex for other reasoners.

- **OntOlogy Pitfall Scanner:** OntOlogy Pitfall Scanner(OOPS!) is a web based tool crated by Villalon *et al.* for detecting common pitfalls in OWL based ontologies [144]. The OOPS! ontology validator detects potential pitfalls within the ontology, details of all the pitfalls are documented here [145]. The OntOlogy Pitfall Scanner will be used to during the evaluation of the PMAD Ontology.

- **OWL API:** The OWL API is an open source Java based API and reference implementation for creating, manipulating and serialising OWL ontologies [109]. In addition the OWL API has the following dependencies: Java SE development kit 7 [146], Simple Logging Facade for Java (SLF4J 1.7.12) [147] and Jfact reasoner 4.0 [148]. The OWL API version 4.0.1 was utilised during the testing of the framework.

5.2 Establishing Guidelines

The first stage in the development of the ontology required establishing a series of guidelines that will govern aspects of the design and development of the ontology. The subsequent sections describe the initial series of guidelines that were established at the beginning of the PMAD Ontologymodel development. This includes an overview of the activities and methods used during each stage of the development lifecycle, design criteria and documentation procedure.

5.2.1 Development Overview

Unlike other methodologies, the Skeletal Methodology of Uschold and Grunninger does not explicitly define a series of activities to follow but rather suggests an outline to help guide throughout the design and development process. Although this allows for flexibility throughout the development, it was worthwhile investing time to construct a plan that provides details surrounding the key stages in the design and development of the ontology. The plan is shown in Table 5.8 was constructed, using knowledge discussed earlier in Section 5.1 and the OWL web ontology language reference material published here [149]. The plan outlines the precise activities, tools and methods associated with each of the development stages outlined in the methodology. Each activity is discussed in further detail throughout the respective sections of this chapter. Alongside the development several other guidelines were also adopted, this included Gruber's ontology design criteria, Rector's modelling approach and a strict naming convention. These are discussed in the sections that follow.

Table 5.8 Overview of the activities, methods and tools associated with the development of the PMAD Ontology model.

Development Stage	**Activity**	**Methods / Tools / Products**
1. Purpose & Scope Definition	Define the purpose of the ontology	Statements using natural language
	Define the scope of the ontology	Statements using natural language
	Define objectives	Statements using natural language
2. Implementation: Coding	Acquire Knowledge	Extract knowledge from various reputable sources
	Define Vocabulary	Build glossary of terms
	Organise Concepts	Build Class Taxonomy
	Identify Relationships	Document relationships
	Create Class Expressions	Document Class Expressions
3. Implementation Capturing	Formalise conceptual model	Using the OWL ontology language
		Protégé development environment

5.2.2 Design Criteria

The list below represents a series of ontology design criteria originally proposed by Gruber [103]. As can be seen, the criteria identify several important characteristics of an ontology. For this reason, the design criteria were adopted to serve as a framework for making key decisions throughout the design and development of the PMAD Ontology. Furthermore, the criteria will also be used to evaluate the ontology.

1. **Clarity:** An ontology should effectively communicate the intended meaning of defined terms. Definitions should be objective, formal, and documented with natural language.

2. **Coherence:** Is a vital criterion in evaluating the consistency of the ontology, an ontology should sanction inferences that are consistent with the definitions.

3. **Extendibility:** An ontology should be designed to anticipate the uses of the shared vocabulary. It should offer a conceptual foundation for a range of anticipated tasks, and the representation should be crafted so that one can extend and specialise the ontology monotonically. In other words, one should be able to define new terms for special uses based on the existing vocabulary, in a way that does not require the revision of the existing definitions.

4. **Minimal Encoding Bias:** Encoding bias should be minimized, because knowledge-sharing agents may be implemented in different representation systems and styles of representation.

5. **Minimal Ontological Commitment:** An ontology should require the minimal ontological commitment sufficient to support the intended knowledge sharing activities.

5.2.3 Modelling Approach

It is common practice for ontology engineers to adopt a modelling approach to help reduce some of the constraints that are associated with developing an ontology [150]. The modelling approach that has been adopted for the development of this ontology is referred throughout the literature as *normalisation*. Normalisation from a relational database perspective is the process of refactoring and organisation of attributes and tables to minimize data redundancy and increase reusability and maintainability of the data contained within the database. This is a formal standardised process, that has a series of strict guidelines and objectives for database engineers. Although there is currently no standard formalised process of normalisation available to ontology engineers, there has been several techniques have been suggested [150–152].

One approach, proposed by Rector [152], aims to normalise domain ontologies by untangling the complexities of ontology modelling. This technique aims to achieve explicitness and modularity in an ontology to support re-use, maintainability and evolution [153]. The principle behind Rector's technique is to divide the ontology into two parts; the primitive skeleton and defined concepts. The primitive skeleton contains basic concepts structured in a taxonomy. Concepts within the primitive skeleton form the foundations of the ontology and act as 'building blocks'that are used to define concepts.

Rector also specifics criteria that the primitive skeleton must retain in order to guarantee the accuracy and explicitness of the ontology. These are as follows:

1. The branches of the primitive skeleton of the domain taxonomy should form trees.
2. Each branch of the primitive skeleton of the domain taxonomy should be homogeneous and logical.
3. The primitive skeleton should clearly distinguish between:

(a) Self-standing concepts - objects from the physical or the conceptual world, for example, `Hardware`, `BluetoothFunctionLogic` ect.. . Self-standing concepts should be disjoint, but use open world assumption. Meaning that the list of primitive children should not be considered as exhaustive, since things within a domain can never be guaranteed.

(b) Partitioning concepts - Are value types and values which partition conceptual spaces. For example "small, medium, large" and are used to refine concepts. Partitioning concepts may or may not be disjoint and should be exhaustive, meaning the 'values' fully covers the value type.

4. The Axioms range and domain constraints should never imply that any primitive domain concept should subsume by more than other primitive domain concept.

This approach not only helps towards achieving the objective of the ontology, but also provides an effective modelling technique for defining concepts within the ontology in significant amount of detail, whilst also enabling the ontology to remain modular and extendible.

5.2.3.1 Naming Conventions

To improve the overall clarity of the ontology a naming convention for entities were established, since OWL does not specify such conventions. Therefore, it was important from a design, development and maintenance perspectives to clearly and easily distinguish between the different entities within the ontology. By establishing a strict series of distinct rules that dictate the character sequences for entities (individuals, classes, object-properties, data-type properties) at the beginning, promotes consistency throughout the design, development and maintenance of the ontology. This also has a direct impact on the readability (from a human's perspective) of the ontology, strict conventions improve

clarity between entities during instances of potential ambiguity or problems that may arise. Therefore, the following conventions presented in Table 5.10, define a series of rules that dictate the character sequences for naming entities within the ontology.

Table 5.10 Naming convention

OWL Entity	**Conventions**	**Example**
Class	Upper Camel Case Notation	ExampleClassName
Object-Property	Lower Camel Case Notation	exampleObjectProperty
Data-Property	Lower Camel Case Notation	dataProperty
Annotation-Property	Lower case, single word	annotation
Individual	Sentence Case	Individual

5.2.4 Documentation

Documentation is a continuous activity throughout all aspect of the design and development of an ontology. Moreover, documentation is a critical component in assuring that the ontology adheres to the clarity and coherent criterion defined by Gruber. Therefore, all documentation produced must be explicit and coherent. Key documentation produced during the development of the ontology is made available throughout Appendix B.

5.3 Purpose and Scope Definition

As described in Section 3.3, the intent of this stage is to define in natural language the purpose and scope of the PMAD Ontology model. This included specifying the overall purpose of the ontology; including how it is intended to be used, its scope and the characteristics of the domain of discourse.

The purpose of the ontology is to play an integral role within a framework that enables healthcare professionals to create personalised mHealth applications to be used as part of a patients care plan. As discussed in Chapter 2, healthcare professionals recognise the benefits of mHealth applications and understand the healthcare requirements of the consumer. However, healthcare professionals are not convergent with mobile application development. Therefore, the ontology model aims to encapsulate key knowledge from the mobile application development process, so it can be made operational via the framework to compensate for the missing domain expertise. The knowledge encapsulated within the ontology will be responsible determining if the mobile device a user owns is capable of supporting the required functionality and the relevant development choices based upon the device a user owns. Furthermore, the ontology will also drive user interface elements of the framework such as menus and options. The ontology is considered as an application level ontology, since it is engineered to be used under a specific use case [122, 154]. Note, it is not the intention of the model to encapsulate the entire domain but utilise the use case scenarios developed in the previous chapter to design a suitable model that is generic, modular and extendible. As illustrated in Figure 5.8, the domain is envisaged to encapsulate knowledge associated with two critical components; Mobile Device and mHealth functions.

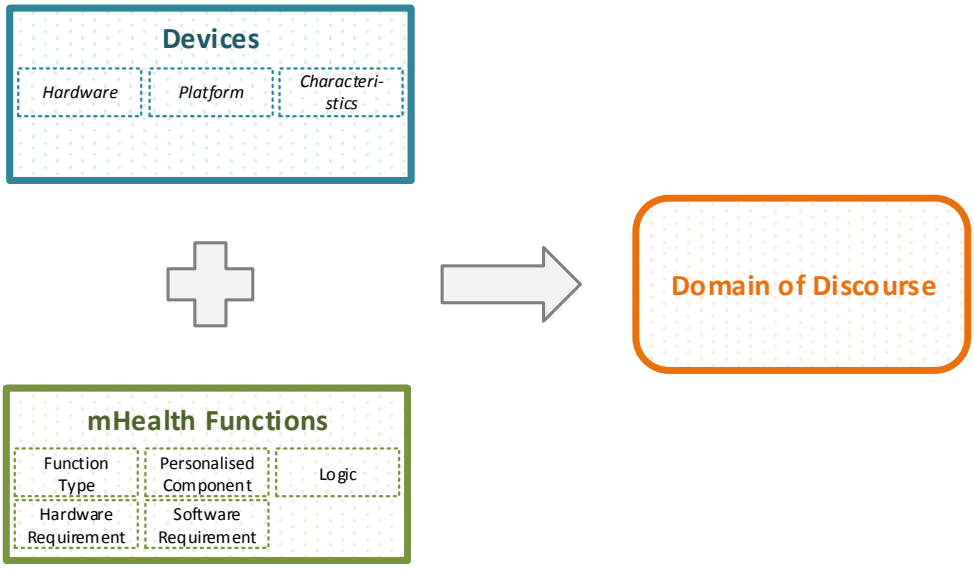


Figure 5.8 Components of the domain of discourse

5.3.1 Ontology Objectives: Competency questions

The ontology objectives are represented as a series of competency questions that play a crucial role in the ontology development life cycle. They represent a series of questions that the ontology must be capable of answering to be competent at tackling the problem it has set out to solve. They align development decisions with the scope, assist in quality assurance of the content within the ontology and provide the basis for evaluation of the knowledge contained within the ontology [155, 156]. The following questions represent a series of competency questions that were established for the development of the PMAD Ontology model. These will be used during the evaluation to determine the competence of the knowledge encapsulated within the ontology.

Table 5.11 Competency questions

Competency Questions	
CQ 1	Can the knowledge encoded within the ontology determine if a function is capable of running on a patient's device?
CQ 2	Can the knowledge encoded within the ontology determine the overall feasibility of the personalised mHealth application?
CQ 3	Can the knowledge encoded within the ontology be operationalised to build a personalised mHealth application?
CQ 4	Can the knowledge encoded within the ontology determine a suitable API based upon the patient's device?

5.4 Implementation: Capturing

The first phase of implementation focuses on capturing, organising and structuring terms to provide an informal representation of the domain of discourse. It begins first discussing the procedure of how knowledge regarding the domain was acquired. Followed detailed and systematic overview of how various semantic models discussed in Section 5.1.2 was used to organise and structure knowledge and how they addressed key semantic challenges and assisted in identifying ontological components for the second phase of the implementation activity. Also discussed in this section are the various constructs and how they are used within the context of the PMAD Ontologyontology.

5.4.1 Knowledge Acquisition

A critical stage in the development of an ontology is linked to how the knowledge that eventually is to be encapsulated in the ontology is captured. As highlighted in the previous activity, the ontology aims to encapsulate knowledge associated with two areas: Mobile devices and mHealth functions. Extracting and describing features from entities was a critical but time-consuming activity. As any mistakes made during this process would lead to inconsistencies in the knowledge encoded within the ontology. Therefore careful consideration was taken when sourcing and extracting knowledge. Knowledge extracted from a variety sources using a combination of automated and manual techniques, which are described in detail in the subsequent sections.

5.4.1.1 Hardware

Chapter 2 discussed the significance of a users device in the development of a personalised mHealth application. The device dictates the scope and limitations of the application

with regards to the platform, operating system and hardware that is available. Therefore, modelling the hardware and software characteristics of a device is a critical component of the PMAD Ontology model. The Android API guide [157] provides a comprehensive documentation list of device features. It includes several categories of device features and a description associated with each feature. This information was extracted and documented.

5.4.1.2 Mobile Device

There are also several Android devices listed in the use case scenarios. Features relating to a specific physical device, such as Nexus 5 were extracted using the Android Debug Bridge (ADB). The ADB is a command line interface between the host computer and the Unix shell located on the device [158]. The ADB provides various actions such as installing/debugging of applications, file transfer or even control the device remotely. The commands `adb shell getprop` & `adb shell pm list features` return a list of all the devices attributes and features respectively. Figure 5.9 shows a screenshot of some attributes and features of a Nexus 5[3].

One issue relying solely on a single device for platform characteristics is that they are representative of a single device. Static features of a device such as hardware do not change over time and can be documented with relative ease. However, dynamic features such as software can change over time, this can be problematic from a maintenance perspective. For instance, the particular Nexus 5 device used during had the property '[ro.build.version.sdk:] [23]' this indicates that the device currently supports up to API Level 23. However, rather than relying on a dynamic property, the static attribute '[ro.product.first_api_level]' was used since this property value the represents first API level the device was commercially launched with and is not subjected to change [159]. Therefore, this property is considered

[3] An Android device manufactured by LG

as the minimum API level recommended for developing applications for this particular device. This is discussed in further detail in the next section.

Figure 5.9 Screenshot of Nexus 5 properties provided by the Android Debug Bridge

Not all the information returned via the ADB required documenting. In fact, there was a considerable amount of redundant information. Redundant properties such as sys.boot_completed, ro.qti.sensors.game_rv and ro.bluetooth.request.master were not documented since they were either beyond the scope of the ontology or described specialised properties associated with specific hardware or software features of the device. To assist in later stages of capturing phase, concept maps were utilised to model features of mobile devices used within the use case profiles. An excerpt of the from the Nexus 5 concept map is presented in Figure 5.10.

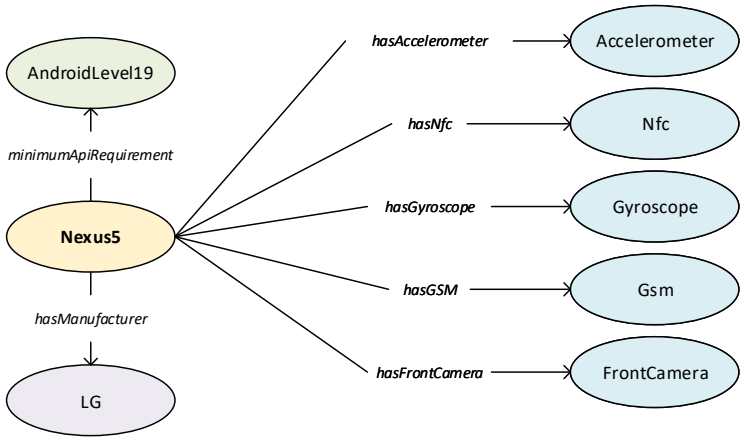


Figure 5.10 Excerpt from the Nexus 5 concept map

5.4.1.3 Android API

As discussed in Section 2.2.2.1, software development kits contain tools, resources and API's. The Application Programming Interface enables applications to interact with the underlying features of the system. Knowledge surrounding Android API was sourced directly from the official Android Developers documentation [160]. The API Level is an integer value that represents a specific version of the framework API offered by the Android Platform [161]. Updates to the framework API often introduces new functionality, for example when new hardware is introduced into the Android Platform. A recent example of is the introduction of fingerprint readers. The framework API is designed so that new revisions of the framework remains compatible with earlier versions. At the time of writing there were 20 iterations of the Android API, API Level 3 was the first publicly available API and the latest being 23[4]. Hence why we can assume that that the API Level that the mobile device launched with is capable of supporting all of its features.

5.4.1.4 mHealth Functions

Chapter 2 highlighted the need for a suitable mechanism for modelling functions. Chapter 4 saw the creation of the mHealth Application Function Taxonomy. The taxonomy provided an insight into the various types of health-related functions available within mHealth applications. Which led to a discussion surrounding several observations made during the analysis of the aforementioned functions. One observation, in particular, discussed the composition of a function with regards to its functional requirements. As discussed in Section 4.3.6, a function is composed of up to three parts: logic, personalised components and dependencies. Figure 5.11 combines the functional requirements and the function type to form the basis of a model that all functions within the use case scenarios will be assessed against.

[4]Note API Level 20 is not included since this revision was introduced to support Android wear devices

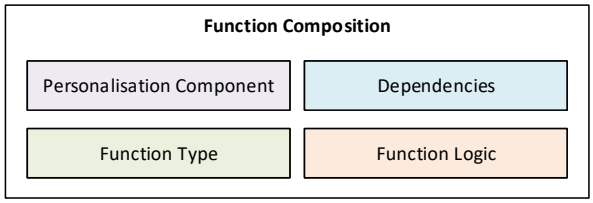


Figure 5.11 Composition of a function

The use case scenarios named functions such as `CallServiceProvider`. To identify the functional requirements of each of the named functions, required developing each of the fucntions required by the use case scenarios. Development of the fucntions was done using Android Studio, which is the official IDE (Integrated Development Environment) for developing applications for the Android platform. Once again concept maps were utilised to document these the functional requirements as shown in the example in Figure 5.12. The following sections provide details of how knowledge associated with each component was extracted.

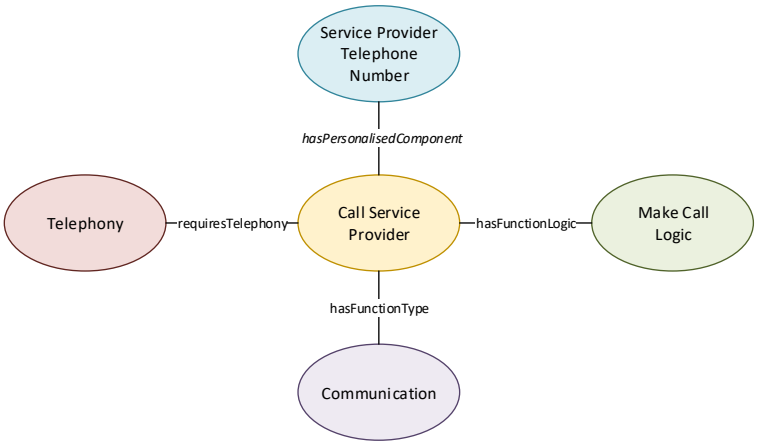


Figure 5.12 CallServiceProvider function concept map

Function Type

The design and development of the mHealth Application Function Taxonomy analysed numerous functions which led to the identification of 4 distinct types and multiple subtypes of mHealth functions. These categories along with their definitions and classification tool provide a standardised mechanism for categorising functions. Hence each function modelled within the ontology will also be assigned a function type using the classification tool and definitions provided in Section 4.4. For example, the call service provider function would be classified as a communication function.

Function Logic

As discussed in Section 2.2.3.2, the service logic refers to the code that is necessary for a particular function to operate. Figure 5.13 is a screenshot of the logic that is associated with calling a service provider. From an ontological perspective, the ontology will not model the logic so to speak, but rather acknowledge it as a concept since it will be the responsibility of the framework to handle the logic. A concept that represents logic will have a prefix the logic appended to the class name.

```
// Funciton to Call Service Provider
call.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View v) {

        try {
            String number = "tel:" + "12345678910".trim();
            Intent callIntent = new Intent(Intent.ACTION_DIAL, Uri
                    .parse(number));
            MainMenuActivity.this.startActivity(callIntent);

            //This exception will be thrown if the device doesnt have telephony hardware
        } catch (ActivityNotFoundException activityException) {
            Toast.makeText(getApplicationContext(),
                    "No Dialer App (are you using a tablet?)",
                    Toast.LENGTH_SHORT).show();
        }
    }
});
```

Figure 5.13 CallServiceProvider: Application logic example

Dependencies

As discussed in Section 4.3.6, many of the functions analysed depended on specific hardware in order to function. When developing an Android application, if a function requires access to specific hardware or software feature, the application must request permission to use it. Requesting permissions is done by declaring the `<uses-permission>` element in applications `AndroidManifest.xml` file. Figure 5.14 shows several examples of various permission requests declared in `Android- Manifest.xml`.

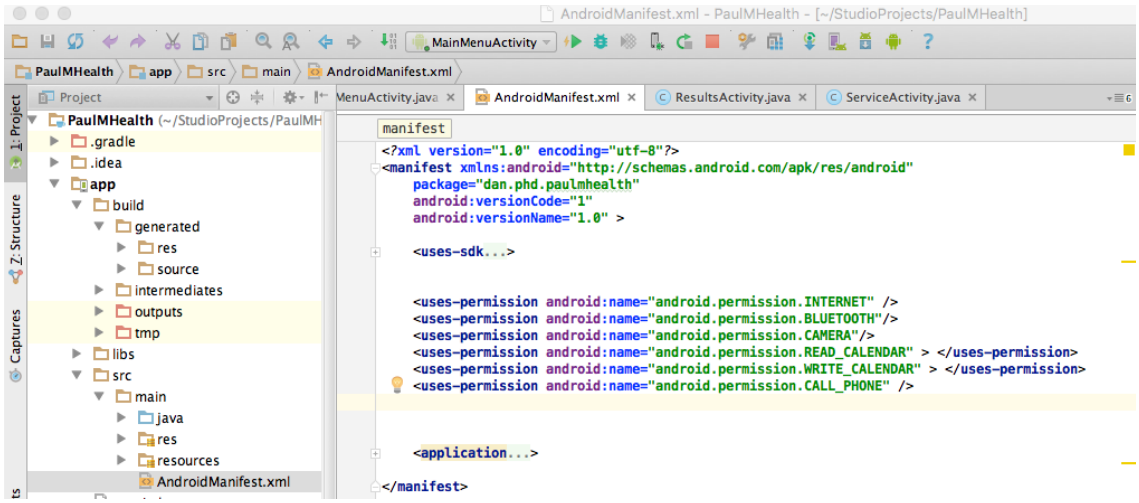
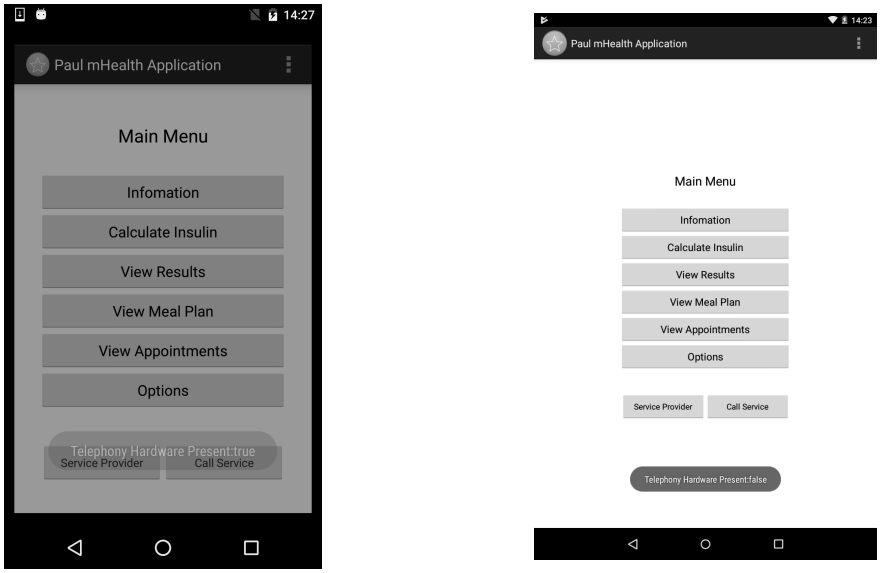


Figure 5.14 Android manifest <*uses-permission*> example

Using the Android API documentation, each of permissions requests was documented and can be mapped to specific features of a device since they imply the application requires the use of a particular feature [162]. Using Figure 5.13 as an example, the permission `android.permission.CALL_PHONE` implies that a device must have telephony hardware in order for the call service provider function to operate. To demonstrate, the same application was deployed onto two physical devices. Using the `hasSystemFeature` method from the `PackageManagerClass`, we can determine programmatically if a specific feature, in this case, telephony hardware, is present on the device. As can be seen in Figure 5.15a, if the device has telephony hardware the value returned is `true` thus the call service provider function could be implemented on this device. If telephony hardware is not present on the

device in the case of Figure 5.15b `false` is returned, meaning the device is not capable of making or receiving calls.



(a) Nexus 5 **(b)** Nexus 7

Figure 5.15 Telephony feature demonstration

Personalised Component

As discussed in Section 4.3.6 the personalised component represents the specific feature or features of a function that can be personalised. In relation to the function logic, certain variables, or documents represent personalised components. In the example shown in Figure 5.13, the string telephone number is an example of a personalised component of the function caller service provider. URL, drug name and insulin bolus are other examples of personalised components from other functions named in the use case profiles. These components allow aspects of the functions to be tailored to the healthcare consumers specific requirements. It is important here not to model 'datatypes' as this is a common mistake made by ontology engineers [145]. The personalised components will be used to form queries within the context of the PMAD Frameworkwere the description of the personalised component from a data perspective will be stored in a database.

5.4.2 Building a Glossary of Terms

As discussed in Section 5.1.3, the meaning of a term can be interpreted differently depending on a variety of factors. The creation of a glossary is the first step in controlling the vocabulary and remove the potential for ambiguity. The purpose of the glossary is to effectively communicate in a concise natural language the intended meaning of an entity within the domain. Although this activity is presented as a singular process, populating the glossary was completed naturally alongside the knowledge acquisition activity. As discussed in Section 5.2.3.1, OWL entities such as classes and object properties have been documented following strict naming conventions to help maintain consistency, alleviate issues and enhance clarity. Each entry listed in the glossary contains the following information: a unique URI (Uniform Resource Identifier), a concise description of the entity, the type of OWL ontological component, as well as any acronyms and synonyms it is associated with. The glossary consisted of over 200 entities. Table 5.12 is an excerpt from the glossary of terms associated with the PMAD Ontology model.

This space has been intentionally left blank for presentation purposes

Table 5.12 Examples of entities within the glossary of terms

Entity Name	Description	Type	Acronyms	Synonyms
...	...	...	...	...
Bluetooth	Hardware component that provides bluetooth wireless communication capabilities	Class		
Mobile Device	A portable computing device, such as a smartphone or tablet computer	Class		
Application Programmable Interface	Represents a set of functions and procedures that allow the creation of applications which access the features or data of an operating system, application, or other service.	Class	API	
hasMinimumApiRequirement	describes the minimum API requirement of a mobile device	Object Property	-	
Near Field Communication	Hardware component that enables NFC wireless communication	Class	NFC	
...	...	...	...	...

5.4.3 Building Concept Taxonomy

Using the semantic spectrum as a guide, the next logical step was to create a semi-formal taxonomy. The glossary also played a vital role in this process as the concise definitions of the concepts assisted in clustering concepts to create the hierarchy. The creation of the concept taxonomy was a manual process that utilised a combination of top down and bottom up development approaches, similar to that used in software engineering. The combination of both approaches allowed the class hierarchy to occur naturally.

Top Down Approach

The top-down approach focused heavily on the creation of primitive skeleton discussed in Section 5.2.3 and allows the creation of the high-level structure of the taxonomy that is based upon the assumptions and pre-existing knowledge regarding the components of the domain. In OWL, concepts within the taxonomy are referred to as things, hence the name `Thing` is given to the root node of the taxonomy. A further three subclasses were added to the taxonomy. The first two `MobileDevice` and `Function` represent the two main components of the domain under consideration. As suggested by Rector (see Section 5.2.3) , the hierarchy should be divided into two parts, making a clear distinction between the primitive skeleton and the defined concepts. To distinguish between the two branches a 'holding class' called `ValuePartition` was added to the taxonomy, which serves as the root node of the primitive skeleton. A description of the `ValuePartition` class was added to the glossary to maintain consistency. As stated in Section 5.3, the ontology will encapsulate knowledge that is related to the use case scenarios, hence concepts that belong to the `ValuePartition` represent self-standing concepts and should not be considered as exhaustive of the entire domain. Self-standing concepts were extracted from the glossary and organised into the x categories: Hardware, Manufacturer, Api, FunctionType, PersonalisedComonent, function logic forming a taxonomy structure. The

knowledge collected from the ADB along with the Android API reference documentation provided an exhaustive organised hierarchy of hardware features available to devices that operate on the Android platform and Android API revisions. Hardware features could be classified into one of 12 different classes such as `Audio`, `Camera` and `Telephony`. The same hierarchy was from the documentation was replicated in the value partition.

5.4.3.1 Bottom Up

This is by no means the final structure of the taxonomy; its purpose is to enable defined concepts described within the glossary to be organised by identifying facets from the bottom up resulting in the creation of high level or more generalised concepts. The focus of bottom-up approach was to further organise defined concepts such as devices and mHealth functions based upon a specific shared characteristic to establish common more generalised classes. This process identified a further 6 classes. Examples included functions that could be organised into one of four distinct types as defined by the mHealth Application Function Taxonomy , and types of mobile devices i.e Android and iOS. Once again new concepts identified as a consequence of this process required updating the glossary to ensure that the vocabulary is consistent and clearly defined. An overview of the high-level class hierarchy.

5.4.3.2 Class Hierarchy Taxonomy

The product of this phase created the structure of the class hierarchy in the form of a taxonomy. The taxonomy serves as the backbone of the ontology and will be used as testing criteria during the coding phase and evaluation of the ontology. For presentation purposes and to avoid repetition the complete class hierarchy has been divided into sections and is discussed in detail throughout Section 5.6.

5.4.4 Building the Relationship Dictionary

In OWL properties represent axioms and are used to define relationships. As discussed earlier, there are two[5] types of properties, Datatype properties and Object properties. Data properties describe the relationship between an instance and a data values. Whereas object properties describe the relationship between instances. Since there are no specific individuals identified within this domain, the focus of this activity was on object properties.

Figure 5.16 Structure of a object property

Object properties describe the binary relationship between pairs of individuals. The pairs consist of a `subject` and a `object` (also referred to as the filler) that are connected via a `predicate` as shown in Figure 5.16. Each of the main branches in the primitive skeleton has at least one association to an object property. These properties are then used to define the conditions for membership of concepts belonging to the defined concept hierarchy, this is discussed in further detail in Section 5.4.5. During the knowledge acquisition activity, concept maps were developed that consisted of named entities within the domain. These concept maps show the relationships between the subject such as `Nexus5` and its features (objects). Each of the concept maps was annotated with additional attributes such as characteristics of the relationship, domain and range. These attributes are discussed in detail in the subsections that follow.

[5]Technically there is a third type of property called *Annotation properties*, however, these are not used to create axioms, these are used to add meta-data to entities

5.4.4.1 subProperties

Similar to classes, object properties in OWL can also be arranged in a tree structure using the `subPropertyOf` construct [149]. From a description logic perspective, this is referred to as a role hierarchy [139]. Consider the two properties `hasHardware` and `hasBluetooth`. If `hasBluetooth` is a sub-property of `hasHardware`, the property extension[6] of `hasBluetooth` should be a subset of the property extension of the parent property `hasHardware`. This states that individuals described using the `hasBluetooth` property are also members of the property extension `hasHardware`. The list below represents the high-level branches of the object property hierarchy[7].

- **topObjectProperty**
  - *hasHardware*
  - *hasFunctionType*
  - *hasFunctionLogic*
  - *hasManufacturer*
  - *hasPersonalisedComponent*
  - *requiresHardware*

5.4.4.2 Object Property Characteristics

In OWL, relationships between individuals can be enriched via the use of seven different object property characteristics: Functional, Inverse-Functional, Transitive, Symmetric, Asymmetric, Reflexive and Irreflexive [163]. Each of these object property characteristics represents a distinctive type relationship that can exist between individuals. As mentioned earlier the concept maps show the numerous relationships to other concepts[8] that a named

[6] Defines the relationship between pairs of individuals

[7] note the topObjectProperty is a built-in object property in OWL that connects all possible pairs of individuals, hence why it is the root of the hierarchy

[8] classes of individuals

concept has. After analysing each concept map, six out of the seven object property characteristics were not required, since the type of relationships they represent are not present within the domain. Therefore, they have been excluded from the discussions that follow. It is worth noting, however, that an object property does not require a characteristic to be specified.

Functional

Functional properties are often referred to as *single value properties* or *features*. With regards to this ontology, functional properties formed some of the relationships. An object property is considered functional if, for a given individual (A), there can be at most one individual (B) that is related to it via the property. Figure 5.17 illustrates the characteristics of a functional property. The solid lines represent the asserted relationship, whilst the dotted lines show inferred relationships. However, if individual A has more than two or more relationships to other individuals (B & C) via the same functional property then this would infer that they are the same individual[9].

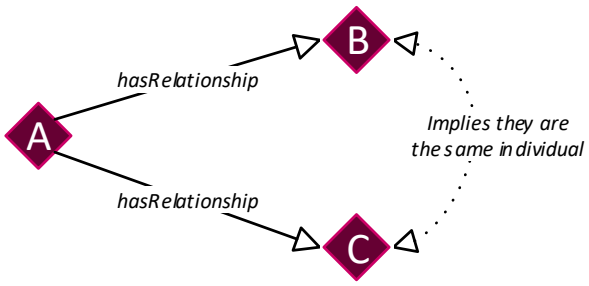


Figure 5.17 Functional property

With regards to the PMAD Ontology model, an example of a functional property would be `hasMinimumApiRequirement`. As this relationship is used to describe the minimum API requirement of the mobile device. In this example, an individual that is a member of a mobile device class can have at most one relationship with and an individual that belongs to

[9]unless explicitly stated otherwise, then the reasoner would flag the ontology has inconsistent

the `Api` class. This is due to the fact that a mobile device can at most have one relationship with an API. Other examples of a functional property include `hasFunctionLogic` and `hasManufacturer`.

5.4.4.3 Domain and Range

In addition to the object properties characteristics, object properties may also have a domain and range specified. The domain and rage of an object property represent axioms that are used by a reasoner to make inferences [139]. The domain and range of an object property constrain the relationship between individuals belonging to the domain to individuals from the range[149]. With respects to the modelling approach discussed earlier, the domain of an object property should be a class from the defined hierarchy whereas the range should be a specified as a self-standing class. To place this into context, using the earlier example the object property `hasHardware` describes the relationship between individuals that are members of a defined class `MobileDevice` (domain) to individuals that are members of the `Hardware` class (range). Thus the domain and range of the `hasHardware` object property can be set to the following:

- **Object Property:** hasHardware
  - **Domain:** Mobile Device
  - **Range:** Hardware

As a result of the restriction applied to the object property, a reasoner would infer any individual that has the relationship `hasHardware` must also belong to the class `MobileDevice`. Likewise, if `requiresHardware` object property has the domain set to `Function` and the range to `Hardware`, this would infer that any individual that has the relationship `requiresHardware` must also belong to the class `Function`.

It is important to note that particular care has to be taken when assigning restrictions to the domain of an object property. As oversights can lead to incorrect inferences being made by the reasoner. By assigning a domain restriction to an object property as seen in the examples from the previous paragraph can result in adverse side effects when describing more complex classes such as a mobile device that are defined using different object properties. As discussed in Section 5.4.1, a mobile device in this domain is interpreted as some 'thing' that is made up of hardware, has a manufacturer and has a minimum API requirement. Now consider this, if a domain restriction `MobileDevice` were applied to each of these properties and for the purpose of this demonstration; a class is defined using only one of the properties [10]. The class would be considered as a subclass of mobile device which it is not within the context of the domain.

To avoid this complication, an alternative approach is to solely rely on necessary and sufficient conditions (see Section 5.4.5.2 for more information) for determining if an individual is a member of a specific class. This approach allows the conditions for membership of classes to be described in significant amount of detail using several object properties and enables the class hierarchy to be inferred and occur naturally based on the necessary and sufficient conditions. However, adopting this approach also has its limitations as it requires an ontology engineer to take precautions when defining classes particularly and check that the inferred class hierarchy is what it is expected to be. Therefore, the domain of each object property is left blank and only the range of the property is specified, this allows the inferred hierarchy to be arranged based on the necessary and sufficient conditions (seeSection 5.4.5.2), whilst also allowing the self-standing concepts to remain as modular 'building blocks' to define concepts.

[10] only is used to represent a single object property and is not to be confused with the universal quantifier ($\forall$)

5.4.4.4 Relationship Dictionary

The product of this activity was the creation of the relationship dictionary. This document organises the relationships in accordance with the position in the object property hierarchy and describes the specific attributes of each relationship. This includes the name given to the object property relationship, the characteristic of the relationship, domain and range. An excerpt taken from the relationship dictionary is presented in Table 5.14.

Table 5.14 Excerpt From the relationship dictionary

Property	Sub Property of	Characteristic	Domain	Range
hasHardware	topObjectProperty	-	-	Hardware
hasAudio	hasHardware	-	-	Audio
hasLoudspeaker	hasAudio	Functional	-	Loudspeaker
hasMicrophone	hasAudio	Functional	-	Microphone
...	...	...	...	...
requiresHardware	topObjectProperty	-	-	Hardware
requiresAudio	hasHardware	-	-	Audio
requiresLoudspeaker	requiresAudio	Functional	-	Loudspeaker
requiresMicrophone	requiresAudio	Functional	-	Microphone
..	...	...	...	...
functionType	topObjectProperty	Functional	-	FunctionType
functionLogic	topObjectProperty	Functional	-	FunctionLogic
hasMinimumApiRequirement	topObjectProperty	Functional	-	Api
hasManufacturer	topObjectProperty	Functional	-	Manufacturer
requiresHardware	topObjectProperty	-	-	Hardware
...	...	...	...	...

5.4.5 Building and Documenting Class Expressions

Up until now all concepts within the glossary have only been expressed using natural language. Although understandable by a human, a machine is unable to interpret the knowledge encoded within the definition, thus is unable to make inferences and enforce semantic constraints. Therefore, the final activity in the capturing phase was to create and document formal class expressions in preparation for the coding phase. Class expressions were formed by utilising the concept maps created towards the beginning of this activity. However, the translation from a concept map to a class expression required an understanding of the various class constructors that enforce specific constraints on a class.

5.4.5.1 Class Constraints

In OWL the class expressions are formed using various constructs, that combine to form class axioms. Constructs are used to constrain the conditions for membership of a specific class by enforcing constraints [164]. As illustrated in Figure 5.18 there are four ways in which a class can be constrained: Class Identifier, Enumeration[11], Property Restrictions and Operators. Note, cardinality restrictions are discussed in this thesis for completeness but are not utilised within the context of this work.

[11] Enumeration is used to specify the set of individuals that belong to a specific class. Although acknowledged enumeration has been excluded from the discussion since there are no individuals specified within the domain

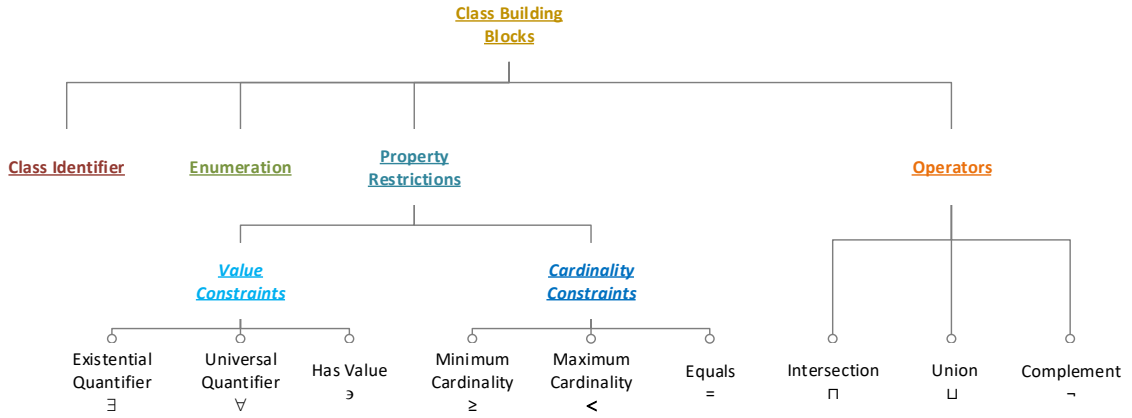


Figure 5.18 Class building blocks in OWL

5.4.5.1.1 Class Identifier

The most simple form of class description is simply the name given to the class [149]. The name given to a class is more formally referred to as a class identifier and should clearly describe the types of individuals it aims to represent. For example, `MobileDevice` represents a class of individuals that are mobile devices. The class identifier is unique in the sense alone it is also a class axiom since it states the existence of a class, whereas the remaining constructs describe an anonymous class[12] or classes by placing constraints on the class extension [149].

5.4.5.1.2 Property Restrictions

Property restrictions describe an anonymous class of individuals by placing constraints on an object and or data property. Properties can be restricted either by using value or cardinality constraints.

[12] An anonymous class represents a class of individuals that satisfy the restriction.

Value Constraints place restrictions on the relationship that the an individual participates in. They either; specify the existence of at least one kind of relationship, or specify the only kinds of relationships that can exist (if they exist), or specify the relationship of an anonymous class of individuals to a specific individual [149].

- ***Existential Quantifier*** ($\exists$) specifies the existence of *at least one* relationship along a specified property between a class of individuals to an individual of a specific class. In Protégé the construct *some* is used.
- ***Universal Quantifier*** ($\forall$) also known as, "all values from" restrictions, describe the relationship between a class of individuals, that via a given property only have relationships along this property to individuals of a specific class. However universal restrictions do not specify the existence of a relationship, they merely state if a relationship exists for the given property then is must only be to individual that ar:e members of a specific class, thus none is also valid. In Protégé a universal restriction uses the *only* construct.
- ***Has Value*** ($\ni$) describes an anonymous class of individuals that via a given property is related to either a specific individual or data property. In Protégé the construct *value* is used.

Cardinality Constraints restrict the potential number of relationships an individual can participate in for a given property and exists in three forms: maximum cardinality, minimum cardinality and equals [149].

- ***Maximum Cardinality*** ($\leqslant$) specifies for a given property the *maximum* number of relationships that an individual can participate in for a given property. In Protégé the construct *max* construct is used.
- ***Minimum Cardinality*** ($<$) specifies for a given property the *minimum* number of relationships that an individual can participate in. In Protégé the construct *min* construct is used.

- ***Equals*** ($=$) specifies for a given property *exactly* number of relationships that an individual can participate in. In Protégé the construct *min* construct is used.

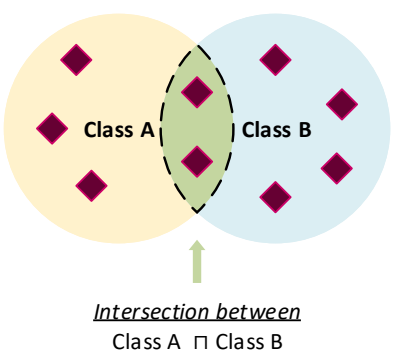
5.4.5.1.3 Operators

Operators are used to combine multiple sets of restrictions to form advanced class expressions. OWL is recognised to have three language constructs: intersection, union and compliment [149].

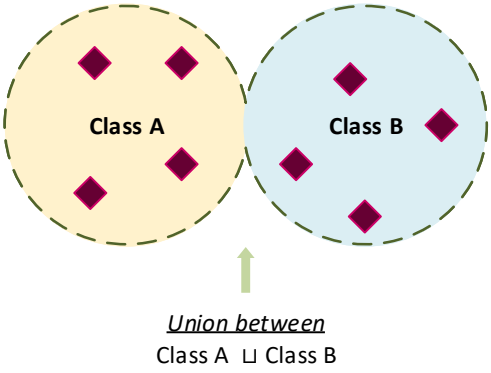
- ***Intersection*** ($\sqcap$) as shown in Figure 5.19a, the intersection between `ClassA` and `ClassB` represents an anonymous class of individuals that are members of both `ClassA` and `ClassB` or equivalently members of `ClassB` and `ClassA`. Note an intersection must contain at least two classes.

- ***Union*** ($\sqcup$) as shown in Figure 5.19b the union between `ClassA` and `ClassB` represents an anonymous class of individuals that are members of `ClassA` or `ClassB`. Note a union must contain at least two classes.

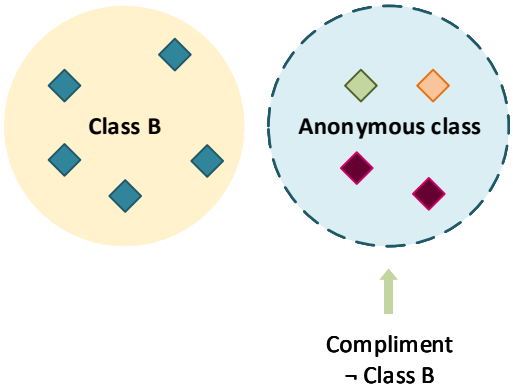
- ***Compliment*** ($\neg$) describes an anonymous class of individuals that do not belong to a specific class. For example $\neg$ `ClassB`, describes an anonymous class of indviduals that are not members of `ClassB` as shown in Figure 5.19c.



(a) Intersection



(b) Union



(c) Compliment

Figure 5.19 Visual representation of operators in OWL: Intersection, Union & Compliment

5.4.5.2 Class Axioms

Class axioms are statements that are used to describe relationships between classes. Formally defining classes is a challenging task when developing an ontology and extreme care has to be taken. As mentioned earlier, the simplest form of a class axiom is the class identifier, as it states the existence of a concept. OWL also has an additional three types of class axioms, these are `subClassOf`, `equivalentTo` and `disjointWith`. Class axioms are utilised by a reasoner to make inferences and specify the necessary and necessary sufficient conditions of class [149].

subClassOf

If *C* is defined as a subclass of *D*, then the set of individuals that are members of *C* must be a subset of the individuals that are also members of *D*. The concept taxonomy created in Section 5.4.3 illustrates the use of the subclass axiom to assert a hierarchy.

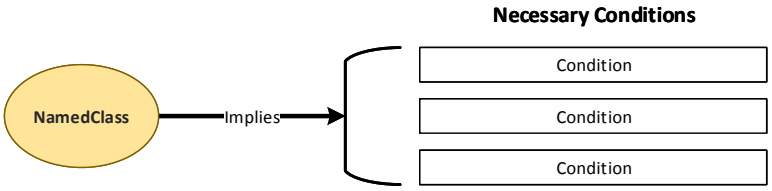


Figure 5.20 Necessary condition

The `subClassOf` construct is also used to describe primitive classes in OWL. A *primitive class* is a concept that is *described* using only necessary conditions. A necessary condition describes the conditions that an individual must satisfy in order to be a member of a named class. However, if a random individual satisfies these conditions, we cannot say that it is a member of this class since the conditions are not sufficient [164, 139]. This is as indicated by the single direction arrow in Figure 5.20. Think of necessary conditions as partial definitions of concepts that we are unable to define completely [139].

equivalentClass

For *C* to be considered as equivalent to *D*, every individual that is a member of *C* must also be a member of *D*. It is worth noting that in an OWL ontology the `equivalentClass` construct does not imply class equality[13], but rather that the classes are equivalent in terms of their class extension (conditions for membership) [149].

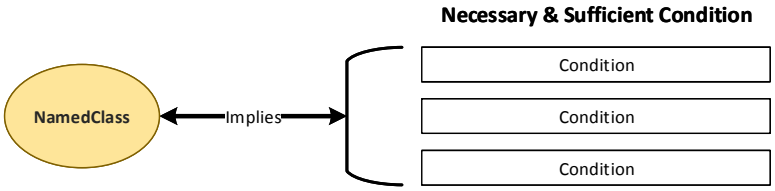


Figure 5.21 Necessary and sufficient condition -

The `equivalentClass` construct is used to express the conditions for membership of a *defined class*. A defined class is expressed using necessary and sufficient conditions. Necessary and sufficient conditions define the conditions that an individual must satisfy in order to be a member of the named class. Moreover, if a random individual satisfies these conditions then the individual must be a member of the named class, as symbolised by the bi-directional arrow in Figure 5.21.

[13] Class equality means that the classes have the same intensional meaning or denote the same concept. In OWL Class equality is expressed using the *sameAs* construct.

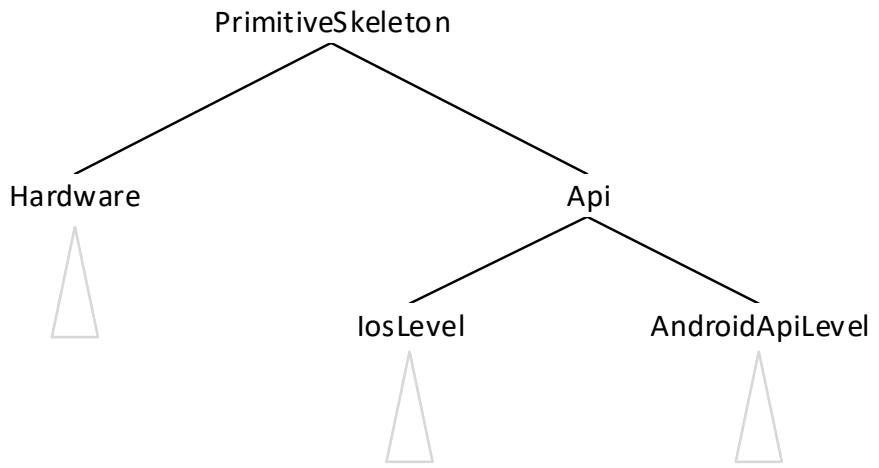


Figure 5.22 Excerpt from the primitive skeleton

$$
\begin{aligned}
\text{MobileDevice} \quad \equiv \quad & \exists\, \text{hasHardware.Hardware} \sqcap \exists\, \text{hasManufacturer.Manufactuer} \sqcap \\
& \exists\, \text{hasMinimumApiRequirement.Api} \sqcap \forall\, \text{hasHardware.Hardware} \sqcap \\
& \forall\, \text{hasManufacturer.Manufactuer} \sqcap \forall\, \text{hasMinimumApiRequirement.Api}
\end{aligned}
$$

$$
\begin{aligned}
\text{AndroidDevice} \quad \equiv \quad & \exists\, \text{hasHardware.Hardware} \sqcap \exists\, \text{hasManufacturer.Manufactuer} \sqcap \\
& \exists\, \text{hasMinimumApiRequirement.AndroidApiLevel} \sqcap \\
& \forall\, \text{hasHardware.Hardware} \sqcap \forall\, \text{hasManufacturer.Manufactuer} \sqcap \\
& \forall\, \text{hasMinimumApiRequirement.AndroidApiLevel}
\end{aligned}
$$

Figure 5.23 Class definitions represented in description logic notation

Necessary and sufficient conditions are defined using class constructors and object properties that are constrained by concepts from the primitive skeleton. Figure 5.22 is an excerpt from the primitive skeleton, `Hardware`, `Api`, `IosApiLevel` and `AndroidApiLevel` are self standing concepts. Figure 5.23 shows how the primitive skeleton and property constraints are used to define two classes. The definition of the class `MobileDevice` implies that, for a random individual to be a member of this class then it is necessary that it has:

- *at least one relationship with an individual that is a member of the* `Hardware` *class and*
- *at least one relationship with an individual that is a member of the* `Manufacturer` *class and*
- *at least one relationship with an individual that is a member of the* `Api` *class and*
- *only has relationships with an individuals that is a members of the* `Hardware` *class and*
- *only has relationships with an individuals that is a members of the* `Manufacturer` *class and*
- *only has relationships with an individuals that is a members of the* `Api` *class*

If all of these conditions are met, then the random individual must be a member of the `MobileDevice` class. This definition utilises what is known as a closure axiom. Due to OWL's open world assumption, a mobile device may have other relationships that we just don't know. Therefore, it is also necessary to 'close' the class by stating these relationships and only these relationships are necessary and sufficient to conclude that a random individual is a member of the mobile device class. The closure of a class is achieved via the combination of the existential ($\exists$) and universal ($\forall$) quantifiers on a given property. The existential quantifier denotes the existence of a given relationship, whilst the universal quantifier states the relationship can only be to the specified filler. This guarantees that if a random individual that has the relationships stated above and only those relationships then it must be a mobile device. Without the closure axiom (i.e without properties constrained by the existential quantifier), the definition would be open and would allow other random individuals that satisfy the conditions plus have other relationships to be also classified as a mobile device.

An elegant product of necessary and sufficient conditions is that a reasoner can use them to infer class hierarchy [164]. In the example above, a reasoner can infer that the class `AndroidDevice` subsumes `MobileDevice`, since the necessary and sufficient conditions of `AndroidDevice` are a subset of `MobileDevice`.

Necessary and sufficient conditions can also be used to define covering axioms. A covering axiom describes an anonymous class that cannot contain any individuals from a class other than the classes specified. Covering axioms are utilised throughout the value partition hierarchy. For example, the class `Monitoring` utilises a covering axiom:

$$\text{Monitoring} \equiv \text{Assessment} \sqcup \text{Tracking}$$

Figure 5.24 Example of a covering axiom

In the example shown in Figure 5.24, the class `Monitoring` is 'covered' by the union of `Assessment` and `Tracking` classes. All individuals in `Monitoring` must be members from the `Assessment` or `Tracking` classes. And there are no other types of `Monitoring`.

disjointWith

For C to be considered disjoint from D, the intersection between the C and D must be an empty set. The disjointWith class axiom is important in OWL as regardless of where classes exist in the hierarchy, OWL assumes that classes overlap. This can in some cases cause incorrect inferences made by the reasoner. Consider the following example, an individual I cannot be both a member of the `Function` and `Mobile Device` classes. As logically a mobile device cannot be a function. Likewise, a function cannot be a mobile device. Without the disjointWith construct, a reasoner would allow I to be a member of both classes. However, by asserting that these two classes are disjoint, prevents this from happening, instead, a reasoner would infer that I cannot exist and would be assigned

membership to the $\perp$ class, `Nothing`; which is an empty set. This indicates that there is potentially an inconsistency within the ontology. The disjoint axiom is heavily utilised throughout the `ValuePartition` hierarchy since it is a requirement of Rector's modelling approach that all self-standing concepts belonging to the primitive skeleton should be disjoint from its siblings, as shown in Figure 5.25.

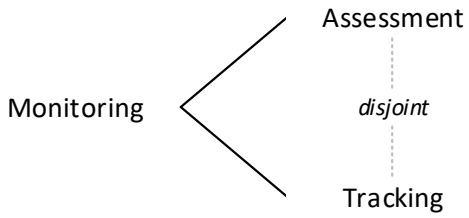


Figure 5.25 Monitoring: Disjoint sibling classes

5.4.5.3 Concept Dictionary

The product of this activity was the creation of the Class Axiom Dictionary. The dictionary contains information relating to the various class axioms discussed throughout this section for each concept within the PMAD Ontology model. This document will be utilised during the implementation and evaluation aspects of the ontology. Table 5.15 below is an excerpt from the class axiom dictionary[14].

[14] some information has been excluded for presentation purposes

Table 5.15 Excerpt from the concept dictionary

Class Identifier	**subClassOf**	**equivalentClass**	**disjointWith**
Hardware	ValuePartition	Audio ⊔ Bluetooth ⊔ Camera ⊔ Fingerprint ⊔ Infrared ⊔ NFC ⊔ Sensor ⊔ Screen ⊔ Telephony ⊔ USB ⊔ WiFi	Api, Logic, FunctionType, Hardware, Manufacturer, PersonalisedComponent
MobileDevice	Thing	∃hasHardware.Hardware ⊓ ∃hasManufacturer.Manufactuer ⊓ ∃hasMinimumApiRequirement.Api ⊓ ∀hasHardware.Hardware ⊓ ∀hasManufacturer.Manufactuer ⊓ ∀hasMinimumApiRequirement.Api	Function
AndroidDevice	MobileDevice	∃hasHardware.Hardware ⊓ ∃hasManufacturer.Manufactuer ⊓ ∃hasMinimumApiRequirement.AndroidApiLevel ⊓ ∀hasHardware.Hardware ⊓ ∀hasManufacturer.Manufactuer ⊓ ∀hasMinimumApiRequirement.AndroidApiLevel	IosDevice
...	...	...	...

5.4.6 Capturing: Summary

To summarise, each activity in the capturing phase focused on documenting or modelling vital components of the ontology. Combined these documents serve as the schematics for the PMAD Ontology model. A summary of each document/model produced by each activity is provided in Table 5.17 below.

Table 5.17 Summary of the design documentation produced during the capturing phase

Document	**Description**	**Purpose**
Glossary	Document containing an unambiguous description of entities within the domain	To facilitate the intended meaning of entities within the domain.
Class Hierarchy	Taxonomy that models the intended hierarchical structure of classes within the ontology	Will be used during the testing and evaluation of the ontology to determine if the correct hierarchy has been inferred
Relationship Dictionary	Document that describes the specific details of the relationships present within the ontology	This document will be utilised during the coding phase, specifically during the creation of object-properties activity.
Concept Dictionary	Document that lists the class axioms for each class within the domain	This document will be utilised during the coding phase, specifically during the definition of concepts activity.

5.5 Implementation: Coding

Up until now the knowledge regarding the domain has been defined, organised, structured to form several key design documents: Glossary of terms, Class Hierarchy (Taxonomy), Relationship Dictionary and Concept Dictionary. The objective of this stage is to encode the knowledge contained within the design documents to create the ontology. The PMAD Ontology model was encoded using the OWL ontology language using the ontology editor Protégé [15]. As shown in Figure 5.26, the coding process consisted of five manageable activities that systematically constructed specific components the PMAD Ontology model. Each of the activities is discussed in the subsequent sections.

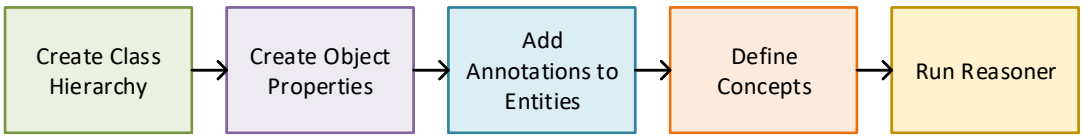


Figure 5.26 Activities with the 'Coding process' of the ontology's development

5.5.1 Create Class Hierarchy

The class hierarchy was built using the *create class hierarchy* tool built into Protégé to the specification modelled in the taxonomy created in Section 5.4.3. Figure 5.27a, shows a screenshot of how the class hierarchy is built using the create class hierarchy tool. Tabs are used to indicate subclasses. The result produces the asserted hierarchy of the ontology. Figure 5.27b shows a screenshot of the value partition hierarchy taken from the class hierarchy viewer. At this stage in the ontology's implementation, it was important to ensure that the structure of the value partition hierarchy was correct, as mistakes here would impact the knowledge encoded.

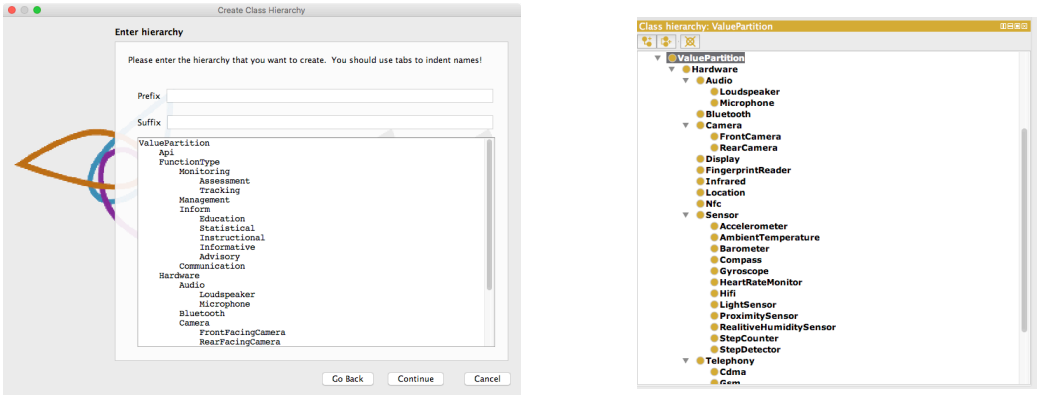
[15] Protégé 4.3 was used, that included the HermiT 1.3.8 reasoner

5.5.2 Create Object Properties

Creating the object properties was completed in two stages. The first stage built the object property hierarchy (as shown in Figure 5.28a) as documented in the relationship dictionary. The second stage added the constraints and characteristics to each of the object properties. The screenshot presented in Figure 5.28b shows the `hasAccelerometer` object property, as can be seen, this object property is functional, a sub-property of `hasSensor` and has the rage specified as `Accelerometer`.

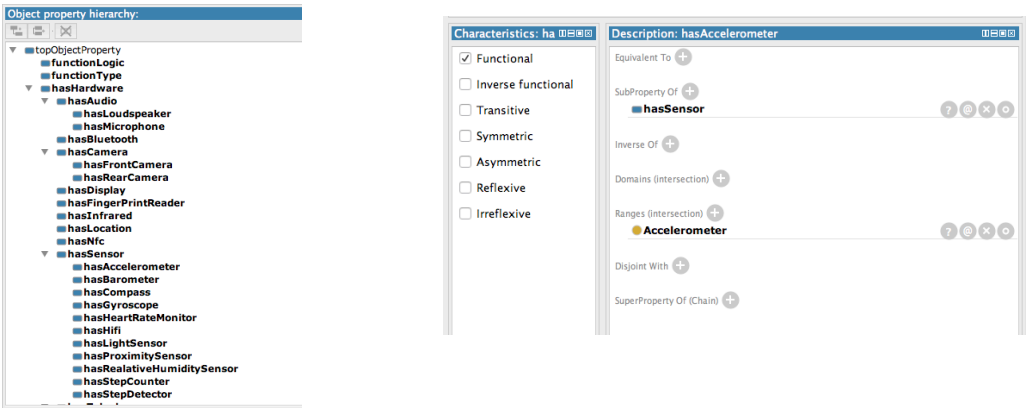
5.5.2.1 Add Annotations to Entities

In OWL *Annotation properties* provide a mechanism for attaching metadata to entities within the ontology. They are not used to assert or infer knowledge within the ontology. For the sake of clarity and to assist throughout the coding and maintenance of the ontology. It was decided that each entity within the ontology will be annotated with the annotation property `description`. This annotation property will be used to provide the natural language definition of the entity from the glossary as shown in Figure 5.29. The reasoning behind this additional step is to further increase readability of the ontology and assist in the description and definition of classes. But also be operationalised via the framework to provide descriptions of components of the ontology to the user, for example, a description of a function could provide more information to the user rather than just a class name.



(a) Create class hierarchy tool in Protégé **(b)** Value partition hierarchy

Figure 5.27 The process of creating the Asserted class hierarchy in Protégé



(a) Object property hierarchy **(b)** Object property constraints and description

Figure 5.28 The process of creating Object properties In Protégé

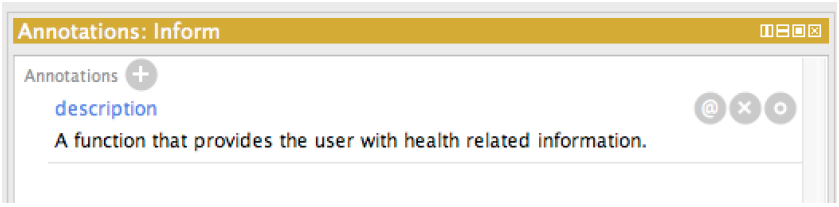


Figure 5.29 The process of annotating entities in Protégé

5.5.3 Defining Classes

The penultimate activity in the coding phase utilised the concept dictionary to add class specific axioms: subclass of, disjoint with and equivalent class to the ontology.

5.5.3.1 Sub ClassOf Axiom

With regards to the subClassOf axiom, the creation of the class hierarchy auto-populated this axiom. Again the defined concepts such as `MobileDevice`, `Function` etc., were all placed as a subclass of `Thing`, as they will be placed into a inferred hierarchy via a reasoner, once they have been defined using necessary and sufficient conditions.

5.5.3.2 DisjointWith Axiom

As stated earlier the disjoint with axiom was heavily utilised in the value partition hierarchy as it was a requirement of Rector's modelling approach. The disjointWith axiom was also used in the defined concept hierarchy to prevent any individual from being a member of both the `MobileDevice` and `Function` classes. Below is a screenshot showing multiple uses of the disjointWith axiom.

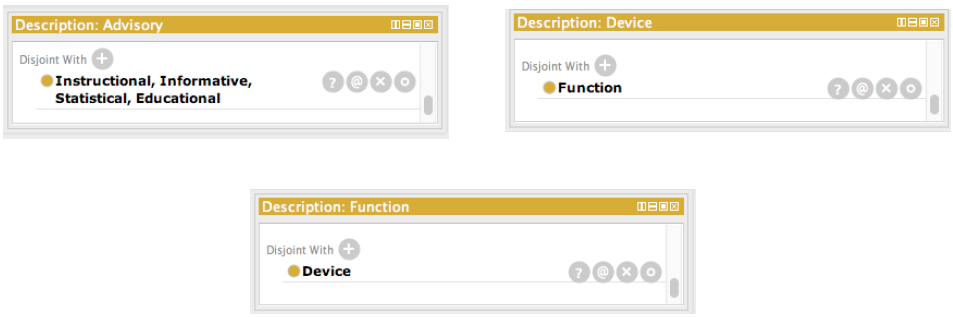


Figure 5.30 Several uses of the disjoint class axiom in Protégé ontology editor

5.5.3.3 Equivalent Class Axiom

As discussed in Section 5.4.5.2, the equivalent class axiom is used to specify the necessary and sufficient conditions of a class. In Protégé class descriptions are presented using the Manchester syntax, which is a user-friendly syntax used for descriptions [165]. Below are several examples of necessary and sufficient conditions that were discussed earlier in Section 5.4.5.2:

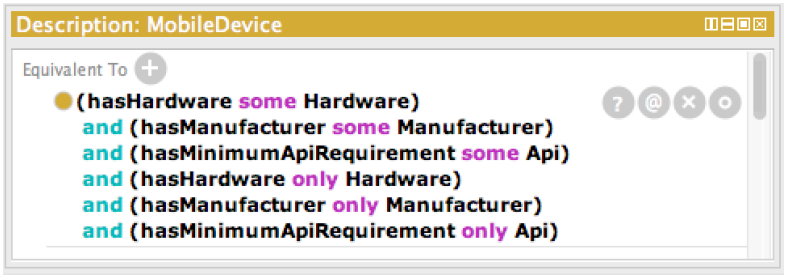


Figure 5.31 MobileDevice class: Necessary and sufficient conditions

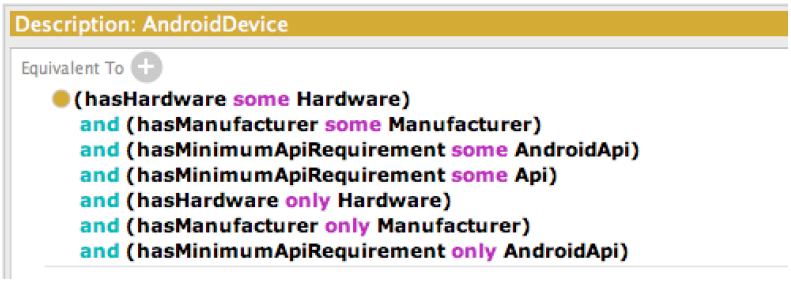


Figure 5.32 AndroidDevice class: Necessary and sufficient conditions

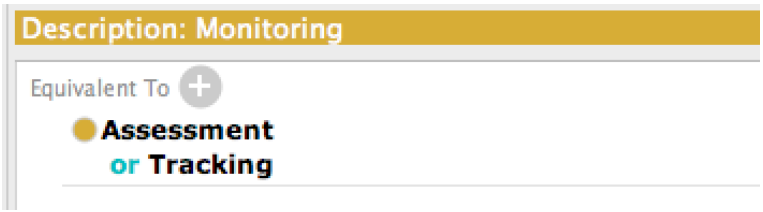
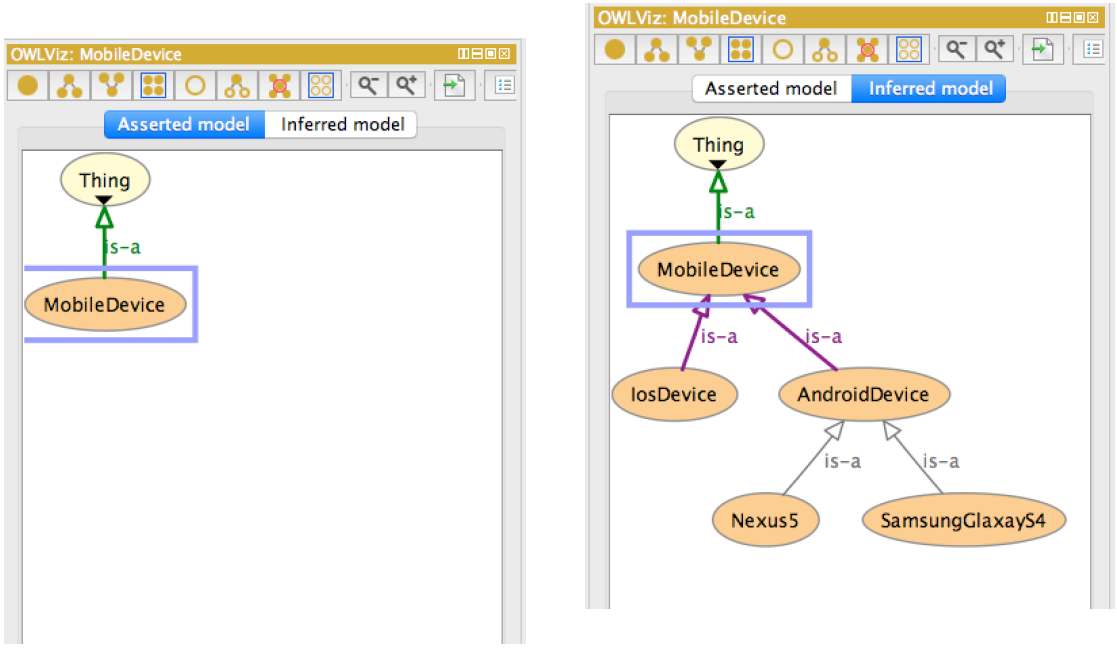


Figure 5.33 Monitoring class: Covering axiom

5.5.4 Running the Reasoner: Inferring Subsumption

The final activity in the coding process required running a reasoner. Although presented as the last phase of the coding process the reasoner was utilised throughout the coding process of the ontology. The role of the reasoner is discussed in further detail during the evaluation of the ontology in Chapter 7. For now, the reasoner has organised the classes belonging to the defined concept hierarchy, based upon their necessary and sufficient conditions. Figure 5.34 shows a comparison between the asserted class hierarchy (Figure 5.34a) and Inferred class hierarchy (Figure 5.34b) from the perspective of a `MobileDevice` class using the OWL Viz plug-in available in Protégé .



(a) Asserted Hierarchy

(b) Inferred Hierarchy

Figure 5.34 Comparison between the *Asserted Class* hierarchy & *Inferred Class* hierarchy in Protégé from the perspective of the 'MobileDevice' class

5.6 PMAD Ontology Overview

The objective of this section is to present the PMAD Ontology model as a completed artefact. Summarised in Table 5.19 are various metrics reported by Protégé that describe several aspects of the ontology. The PMAD Ontology created as part of this research consists of a total of 600 axioms, 340 logical axioms, 140 classes, 63 object properties and 203 annotation properties axioms. Protégé determines the Description Logic (DL) expressivity based upon the concept constructors, property (role) constructors and axioms used to encode knowledge [139, 166, 167]. The ontology engineered as part of this research has been characterised with a DL expressivity of $\mathcal{ALCHF}$, in short, the ontology utilises the base language constructs, which are extended via the use of complex concept negation, property hierarchies and functional properties as discussed throughout Section 5.4. For more information regarding the DL, expressivity see Appendix B.2. The remainder of this section presents an overview of the ontology, beginning with the value partition hierarchy.

Table 5.19 PMAD metrics

Ontology Metrics	
DL Expressivity	$\mathcal{ALCHF}$
Axioms Count	600
Logical Axiom Count	340
Classes Count	140
Object Properties Count	63
Disjoint Axiom Count	17
Annotation Properties Count	203

5.6.1 Value Partition Hierarchy

As discussed in Sections 5.2.3 and 5.4.3 the value partition represents self-standing concepts, the building blocks of used to define concepts. As shown in Figure 5.35, the upper level of the Value Partition hierarchy consists of 6 branches. Each branch is disjoint and has its own hierarchy, each of which is discussed throughout sections that follow.

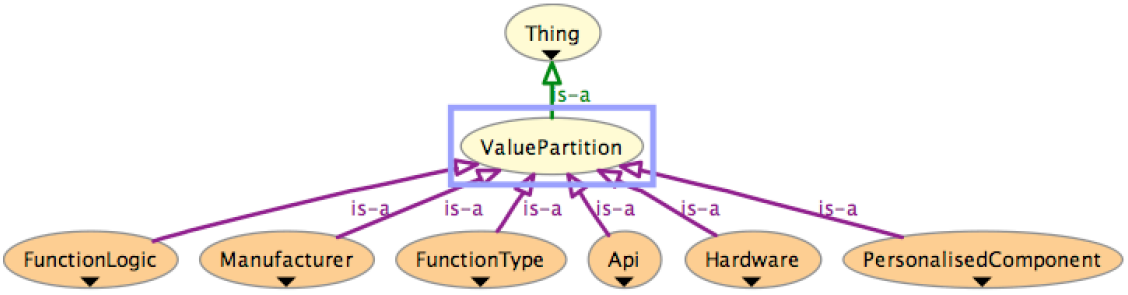


Figure 5.35 Upper level of the ValuePartition class hierarchy

5.6.1.1 Hardware

The hardware branch class represents self-standing concepts that depict the hardware features of a mobile device. The Android reference documentation provided an exhaustive list of hardware features of devices that operate within the Android platform. As shown in Figure 5.36 the `Hardware` class has 12 subclasses: `Audio`,`Bluetooth`, `Camera`, `Display`, `FingerprintReader`, `Infrared`, `Location`, `Nfc`, `Sensor`, `Telephony`, `Usb` and `Wifi`. Classes such as `Audio`, `Camera`, `Telephony` and `Sensor` also contained several subclasses. For example, the class `Sensor` for contains 13 distinct subclasses.

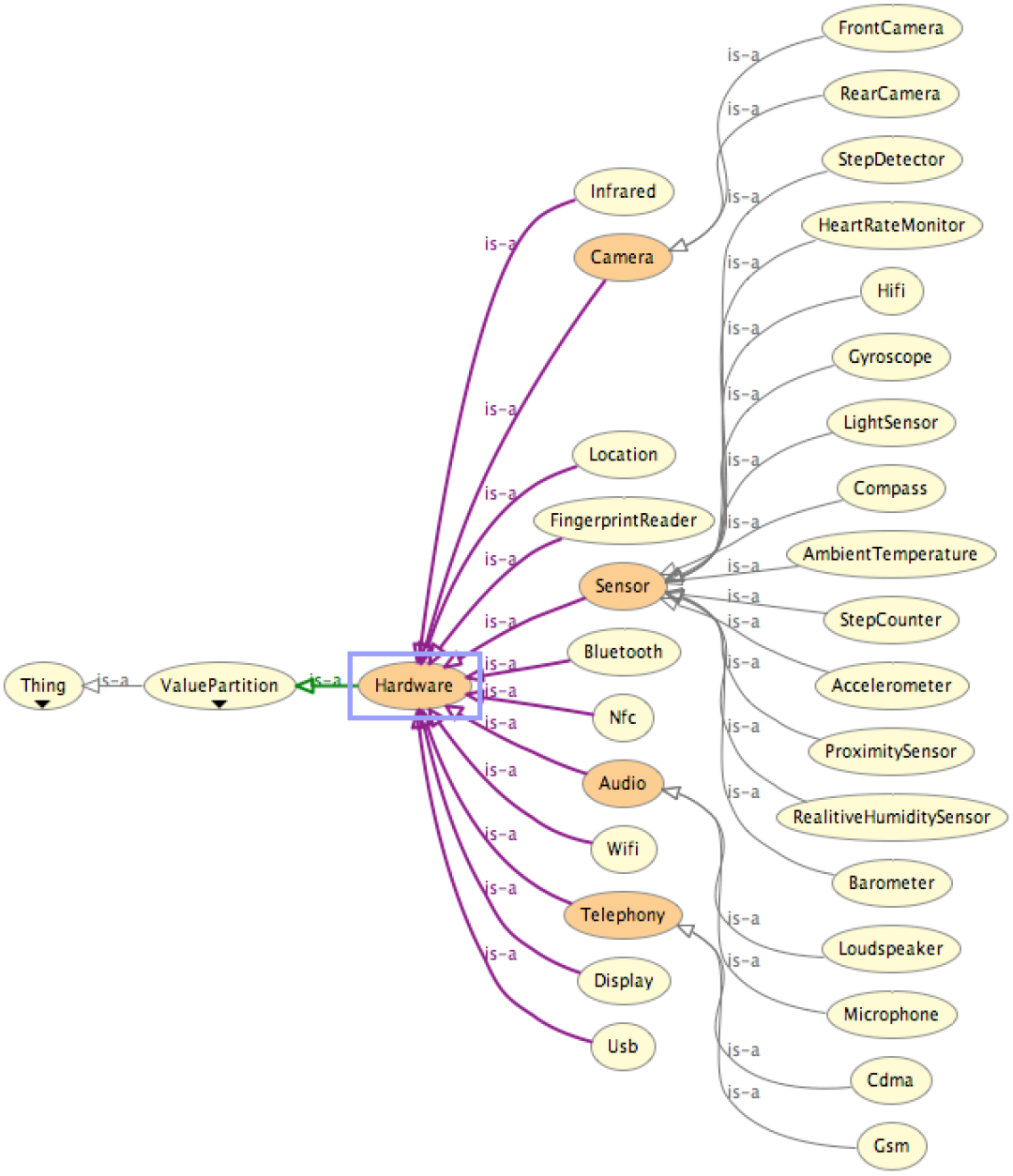


Figure 5.36 Hardware class hierarchy found within the ValuePartition hierarchy

5.6.1.2 Api

As described inSection 5.4.3, the `Api` class consists of the platform-specific API Levels. During the time of development, the Android Platform has 20 revisions of the Android API. an Excerpt from the API hierarchy is shown in Figure 5.37.

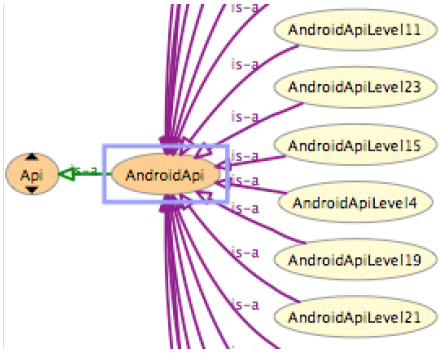


Figure 5.37 Excerpt of the API class hierarchy found within the ValuePartition hierarchy

5.6.1.3 FunctionType

As can be seen in Figure 5.38, `FunctionType` branch simply replicated the class hierarcy estanblished in the mHealth Application Function Taxonomy, refer to Section 4.4 for more information.

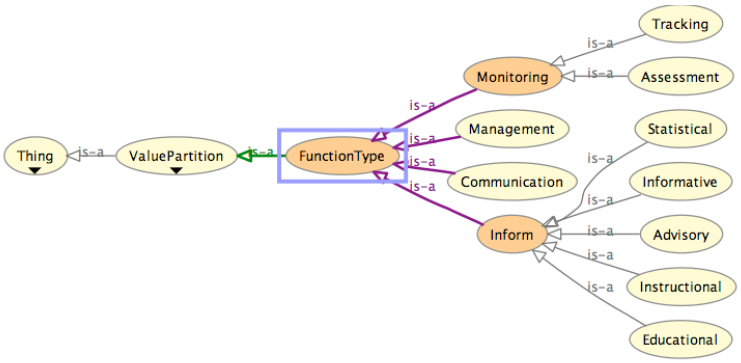


Figure 5.38 FunctionType class hierarchy found within the ValuePartition hierarchy

5.6.1.4 FunctionLogic

The service logic refers to the code that is necessary for a particular function to operate. Each of the classes that belong to the `FunctionLogic` is representative of the logic necessary for a particular function to operate. For each mHelath function required in the use case scenarios required a function logic class. All function logic concepts have the suffix 'Logic' appended to the end the class identifier for clarity. Figure 5.39 is a screenshot taken of `FunctionLogic` hierarchy.

5.6.1.5 Manufacturer

As shown in Figure 5.40 represents device manufacturers. There were four manufacturers within the use case scenarios: 'Apple', 'Asus', 'LG' 'Samsung'.

5.6.1.6 PersonalisedComponent

The `PersonalisedComponent` class represents the element of the function that can be tailored to the requirement of a healthcare consumer and had 14 subclasses.

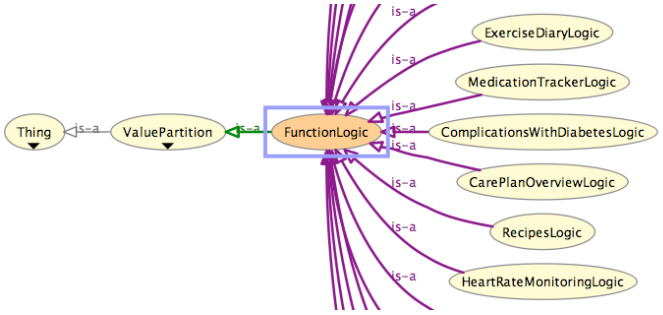


Figure 5.39 Excerpt of the FunctionLogic class hierarchy found within the ValuePartition hierarchy

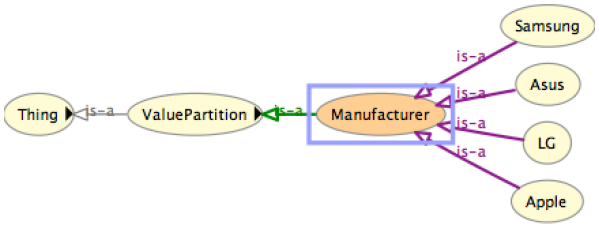
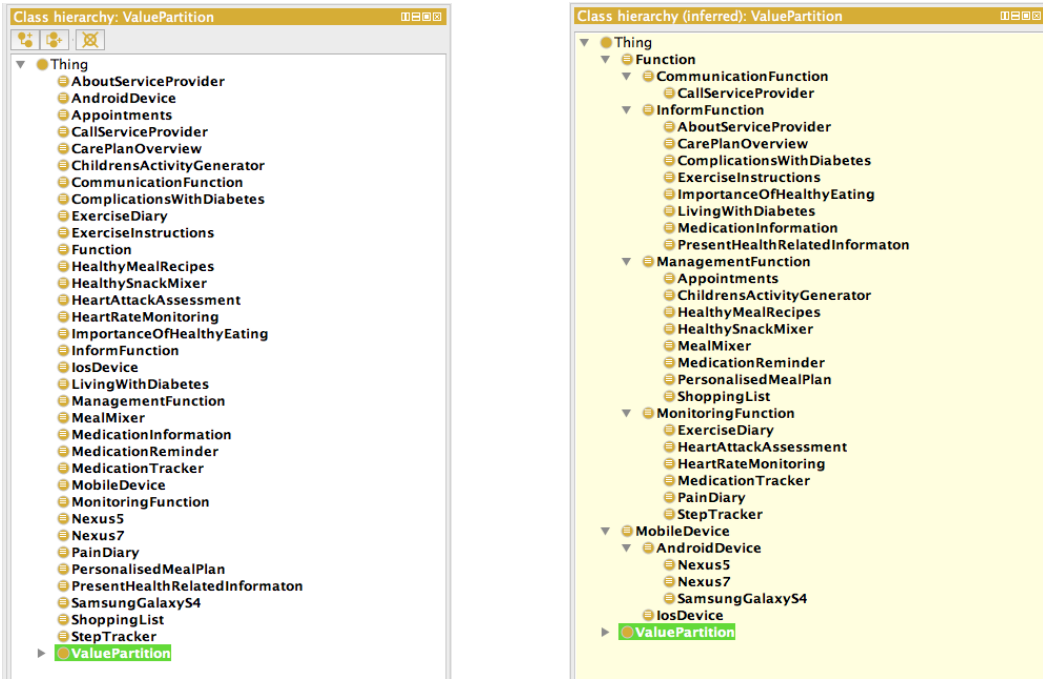


Figure 5.40 Manufacturer class hierarchy found within the ValuePartition hierarchy

Figure 5.41 PersonalisedComponent class hierarchy found within the ValuePartition hierarchy

5.6.2 Defined Concept Hierarchy

During the coding of the PMAD Ontology, defined concepts were not organised into any sort of a hierarchy, this task was left to the reasoner. This was for several reasons. The use of a reasoner, keeps the ontology in a logical, maintainable and modular state, promoting the reuse of the ontology in other ontologies and or applications, this is particularly useful as the ontology grows in size and complexity [149]. It also relieves the ontology engineer from a task whilst reducing human errors [87]. Figure 5.42a, shows the asserted 'defined concept hierarchy', notice that these classes are not arranged in any form of hierarchy. Based upon the necessary and sufficient conditions subsumption was inferred producing the hierarchy shown in Figure 5.42b. As can be seen the upper level of the 'defined concept hierarchy' consists of two disjoint classes `MobileDevice` and `Function`.



(a) Asserted class hierarchy **(b)** Inferred class hierarchy

Figure 5.42 Defined concept hierarchy comparison

5.6.2.1 MobileDevice

The `MobileDevice` class refers to a portable computing device, such as a smartphone or tablet computer. As can be seen in Figure 5.43, there are two subclasses `AndroidDevice`, `IosDevice`[16] which are disjoint since a mobile device can only operate on a single platform. The use case scenarios used three devices: 'Nexus5' & 'SamsungGalaxyS4' are cellular mobile devices, 'Nexus7' is tablet computer. Each of the devices are again disjoint and are defined using values from the Manufacturer, Hardware and ApiLevel hierarchies.

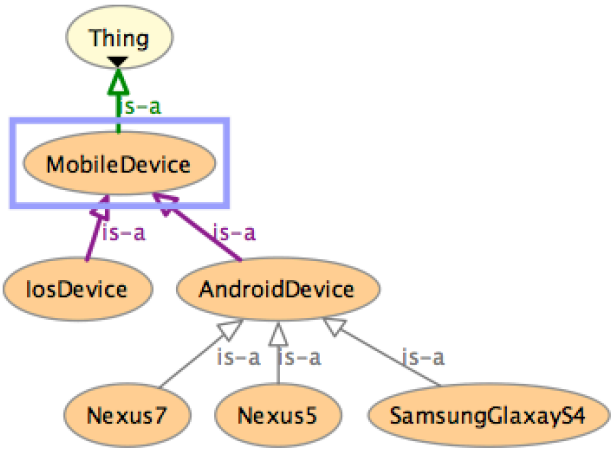


Figure 5.43 Defined concept hierarchy: MobileDevice Class hierarchy

[16] iOS was included during the development of the ontology for evaluation purposes

5.6.2.2 Function

A Function in the context of this domain is defined as a service that is provided to a health-care consumer. The use case profiles required a total of 22 functions categorised into one of 4 disjoint function categories shown in Figure 5.44. Functions were described using the values from the Hardware, FunctionType, FunctionLogic and PersonalisationComponent classes.

As discussed in Section 4.3.6.1, mHealth functions from a personalisation perspective existed in two forms: specialised and generalised. The majority of the functions required in the use case scenarios were specialised functions, meaning the logic and personalisation component was unique to that particular function. Whereas, generalised functions are described as using the same logic but was made unique because of the personalisation component. For example, `LivingWithDiabetes` and `AboutServiceProvider` were defined almost semantically identical from an app development perspective, it was the personalisation component 'personalised' the service it provided the user.

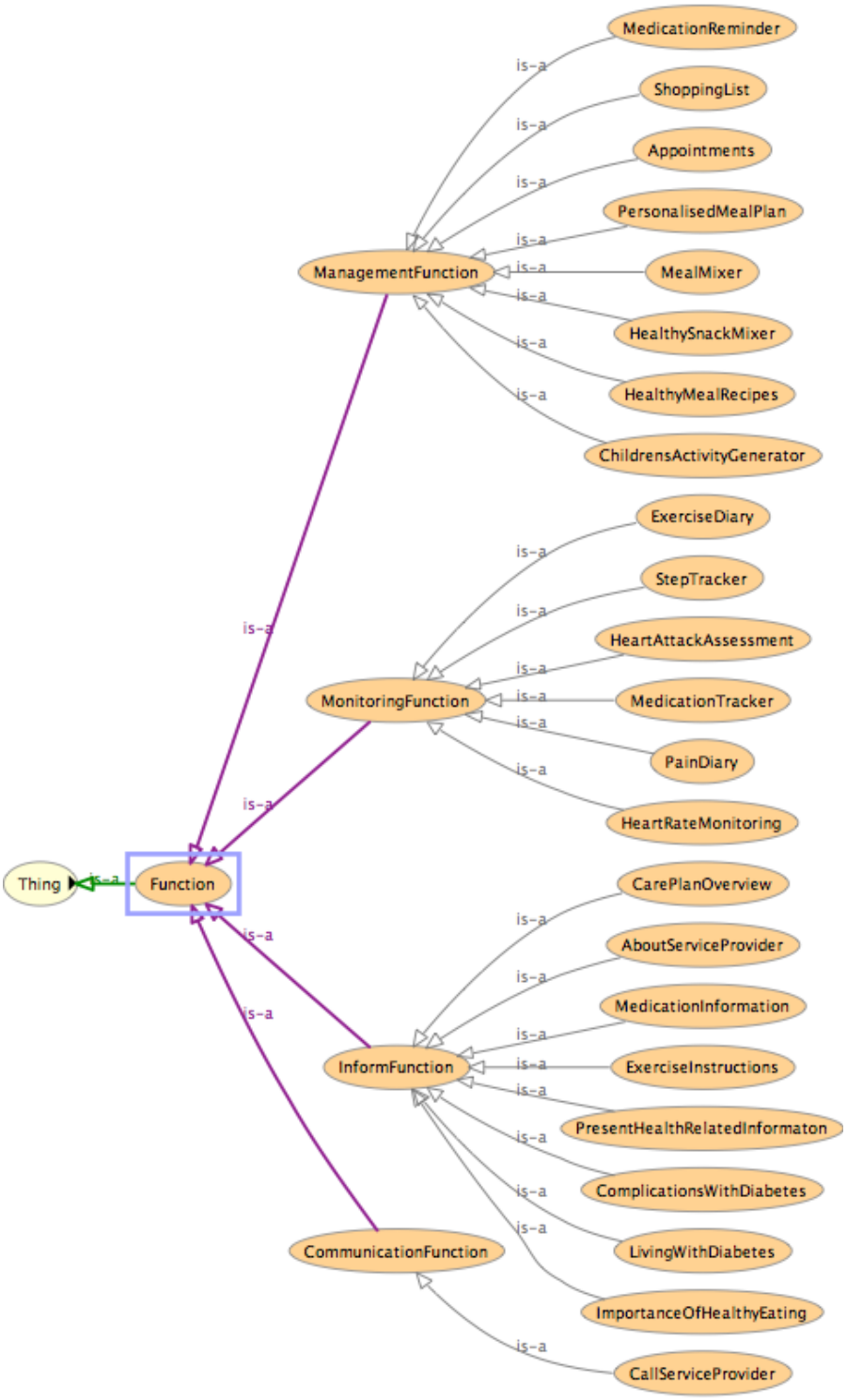


Figure 5.44 Defined concept hierarchy: Function class hierarchy

5.7 Summary

In summary, this chapter discusses the theoretical, design and development considerations of the PMAD Ontology. The purpose of the ontology developed throughout this chapter was to encapsulate knowledge associated with the development of personalised mHelath applications; so, it can be operationalised and play an integral role within a framework enabling healthcare professionals to create personalised applications to be used as part of a patients care plan. The ontology was developed following the Skeletal Methodology of Uschold and Grunninger and the design of the ontology was influenced by Gruber's ontology design criteria and Rector's modelling approach.

Chapter 6

PMAD Framework

The motivation behind this chapter is to present and discuss the design considerations of an ontology-driven framework. The PMAD Framework has been designed to enable healthcare professionals to create personalised mHealth applications for healthcare consumers. The chapter is structured in three parts, beginning with a discussion surrounding the purpose, scope and design criteria. The second aspect discusses the creation of a conceptual model and explores various factors that influenced the design of the PMAD Framework. The final part presents an overview of the PMAD Framework and discusses each component in detail.

6.1 Purpose, Scope and Design Influences

As discussed in Chapter 2, developing personalised mHealth applications is not feasible nor sustainable using traditional approaches to mobile application development. Existing end-user programming solutions that enable people who do not possess the necessary, skills, knowledge or familiarity of mobile application development to create, mobile applications have several limitations such as the small number of general functions, focus specifically on a particular aspect of healthcare or do not provide any mechanisms to expand the functions they support. The purpose of the PMAD Frameworkis to provide an on-demand service that enables healthcare professionals to create personalised mHealth applications for healthcare consumers as part of the healthcare plan. It is not the intention of this research to implement a fully operational platform, but rather a generic modular high-level framework which represents the main architectural components and services of such a system.

6.1.1 Design Criteria

As with other aspects of this research, it was important to establish a series of criteria to be used to evaluate choices throughout the design of the framework. The following design influences were inspirations taken from various worked cited and read during the development of this research. They represent considerations from both the user and systems perspectives.

- **Simplicity:** Healthcare professionals typically do not have the time or possess the necessary domain expertise to develop personalised mHealth applications. Therefore, the process of creating personalised mHealth applications should be intuitive and clear to the user.

- **Transparency:** Ties in with simplicity, the complexities of the inner mechanics should remain hidden from the end user.

- **Modularity:** components of the framework should be follow a modular design philosophy, dividing the system into smaller parts that can be independently created and used in different systems.

6.2 Building A Conceptual Design

Prior to the design of the framework, it is important to have a clear understanding of how the envisaged system will function [44, 168]. Budgen states that there are two possible strategies that can be deployed when seeking components that make up the functionality of a system [169]. *Element first* strategy, identifies the general needs of the problem and then searches for a set of components that collectivity match the functionality, bringing them together to form a system. *Framework first* strategy decomposes the problem into fairly well-defined sub problems and then seeks a set of components that will fit the needs of each sub problem. The development of the conceptual design combines both strategies opportunistically as the design matured.

6.2.1 Use Case Modelling

The construction of the conceptual design began by identifying at a high level the main functionality of the framework. In order to visualise the functionality of the framework from a users' perspective, a use case diagram was used. A use case diagram specifies a set of interactions between actors and use-cases in order to achieve a particular goal [112, 113]. Actors represent a person, organization, or external system that plays a role in one or more interactions with a system, in this work actors represent users. A use case

is presented as an oval and describes a sequence of actions that provide something of quantifiable value to an actor, ie functionality of a system. Lines represent associations. Finally, a system boundary defines the scope of the system, representing a collection of functionality. Figure 6.1, represents a high-level abstract interpretation of the core functionality of the framework. As can be seen there are three stakeholders; *Healthcare Professional, Healthcare Consumer* and *System Engineer*. Healthcare Professionals (HCP), are the primary users of the framework. A HCP will be involved in the creation of a personalised mHealth application as well as view the data produced by the application. Healthcare Consumers (HCC) are the secondary users. Although HCC won't interact directly with the system, they will interact with the applications created. System Engineers (SE) are the tertiary users of the system and have the responsibility to maintain update and further develop various components and services of the framework.

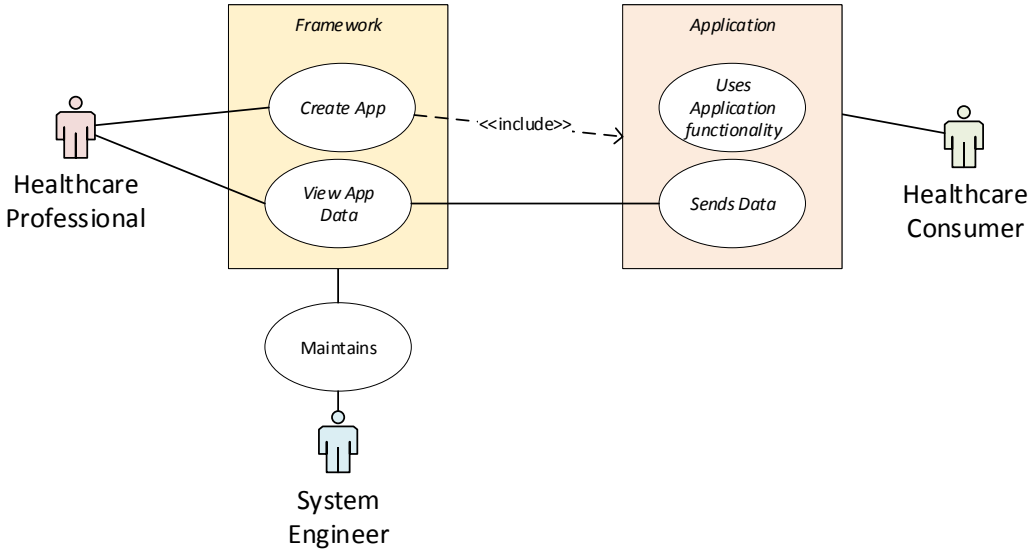


Figure 6.1 High level use case scenario

6.2.2 Development Process Considerations

The second phase analysed each of the end-user programming solutions discussed in Section 2.3. Sommerville describes, with respects to traditional software engineering processes[1], four fundamental software engineering activities, that in some form, are way present in all software engineering processes. Each activity is described briefly below within in the context of mobile application development:

1. **Specification:** definition of the mobile applications functionality and constraints on its operation.
2. **Development:** the design and coding of the mobile application defined in the specification.
3. **Validation:** checks performed on the mobile application to ensure that it is to the specification required.
4. **Evolution:** modifications to the mobile application.

One of the biggest challenges for this framework is ensuring that key activities associated with the mobile application development process are streamlined and the complexities remain transparent from the HCP. This is mainly for two reasons, the first is to dramatically reduce the knowledge needed to build mHealth applications in order to not to overwhelm the HCP. The second is to provide a sustainable method for developing personalised mHealth applications. Using these four fundamental activities as a baseline, the development process of each end-user programming solutions was modelled, analysed and discussed below. Figure 6.2 shows the development process for each of the existing solutions[2]. Each of the approaches shared similar characteristics. Each adopted a streamlined

[1] A software engineering process is a set of interrelated activities that result in the creation of a software product

[2] Details of each activity relating to the specific platforms can be found in Section 2.3

version of the engineering process; each relied on different implementations of graphical user interfaces to ease aspects of the engineering process.

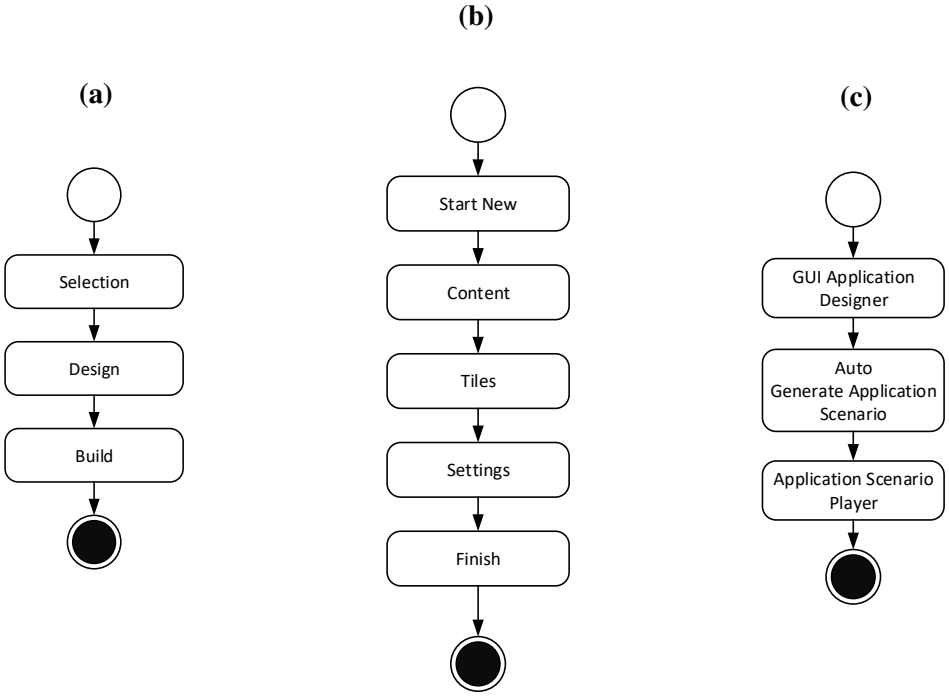


Figure 6.2 Existing solutions development process:**(a)** Appy Pie, **(b)** Microsoft App Studio and **(c)** Muss Platform

6.2.2.1 Validation

The purpose of the software validation activity is to determine if the application produced is compliant with the requirements specification and expectations of the end user [170]. Although in some software engineering processes validation remains a single exclusive entity, it is more common to see validation activities appear throughout the development lifecycle. For a framework such as this, it is vital that it includes validation mechanisms to review and inspect an application throughout key stages in the engineering process to minimise the required.

6.2.2.2 Specification

In a traditional setting, the specification aims to produce a mutually agreed upon document that details the requirements of a mobile application between stakeholders and application developers. It is described as the critical stage of the software process, throughout the literature as errors at this stage inevitably lead to problems later on in development. As emphasised throughout Chapter 2, HCP's understand the requirements of the HCC but lack the necessary skills, knowledge and familiarity associated with mobile application development, meaning that are not in a position to determine the feasibility of the application, with respect to its functionality and operating conditions. Existing solutions, attempt to neutralise this issue by restricting the scope of the application to a predefined set of rudimentary functions, that require 'primitive' (minimal) hardware for it to operate. However, as witnessed during the creation of the mHealth Application Function Taxonomy, mHealth functions exist in many forms, vary in complexity and even depend on the presence of specific hardware. Hence, one of the reasons for the creation PMAD Ontology. The knowledge contained within the ontology will be used to determine, based on the HCC's device; the feasibility of the functionality required based upon the operating constraints and recommend a suitable API. This not only simplifies the specification activity from the perspective of a HCP, but also helps minimise the likelihood of errors later on in development.

6.2.2.3 Design and Implementation

What is interesting is to see how each of the existing platforms tackle the 'design' and 'implementation' activities and how they differ from traditional interpretations. In traditional software engineering, the design activity can be summarised as a conceptual overview of the structures that form the application [44, 171, 172]. The form and the extent of the

design process can influence by the outputs from the specification activity and other factors such as scale of the application, design practises and methods of implementation [172]. The implementation activity follows naturally after the design activity and is the process of converting the conceptual interpretation of the application into an executable application.

With regards to the design of the application, each platform uses a controlled approach to the design process. All platforms utilise a graphical user interface to guide users through the design and implementation process, but tackle design from two different perspectives, aesthetics and logic. Both Appy Pie and Microsoft App Studio focused mainly on the aesthetics of the application allowing the end-user to customise various elements of the mobile applications user interface such as colour, layout and position of UI elements. Both these platforms used forms, wizards and validation when populating the necessary fields required by the selected function, neither require the users. Once completed the platform would then proceed to automatically build and compile the application. Whereas the MUSS platform required healthcare professionals to use a visual programming language to design the logic of a particular function. Based upon the results would generate XML that is then interpreted by the 'scenario player' installed on the user's device. Both approaches have their advantages under different use cases. However, the main focus of this framework is to enable HCP's to create personalised mHealth applications for HCC, without the need for intervention from application developers. HCP's should be able to utilise their knowledge of the HCC to personalise apps functions. Hence the second reason for the ontology. Based upon the functions required during the specification activity, the knowledge contained within the ontology will be used to generate forms consisting of fields that require the HCP to populate. These fields will represent the personalisation component of each function and will be validated prior to implementation. Leaving the system responsible for both the aesthetics and implementation of the application.

With regards to the mobile application architectural style, each of the end-user programming solutions utilises a different application architectural styles. mHealth applications developed using the MUSS platform utilise web applications. Mobile applications developed using Microsoft App Studio are native applications and Appy Pie utilises Hybrid applications. Deciding which type of application architecture to use for this framework required considering two aspects. The first considers the intended capabilities of the application. From a functional perspective, the development of the mHealth Application Function Taxonomy and PMAD Ontology shows there is a diverse range of health-related functions available, many even required access to specific hardware to operate. This ruled out the possibility for a complete web mobile application architecture since web applications have restricted performance and functionality, leaving to potential candidates Native and Hybrid. The next factor to consider was from the perspective of the framework, which application architecture would allow the most flexibility with regards to development. As discussed in Section 2.2.2, native applications provide the richest and most compelling experience for end-users since and excel in regards to raw performance and functionality they support. They are built using platform-specific tools. Meaning from a development perspective, for every platform the framework would have to handle both the UI and functions independently. Whereas Hybrid applications, as shown in Figure 6.3, allow for the amalgamation of both the native and web technologies allowing the reuse portions of the code, specifically the UI for different mobile platforms. From the perspective of the framework, this reduces development time and commitments. Therefore, the applications created using this framework should utilise a Hybrid application architecture.

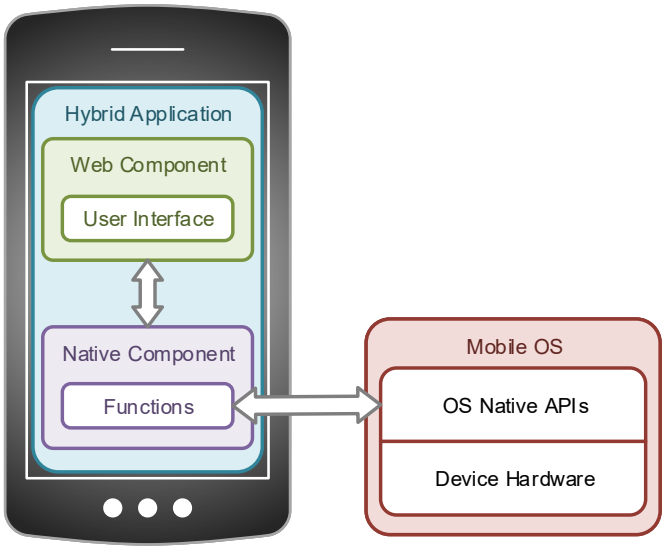


Figure 6.3 Overview of a hybrid mobile application architecture

6.2.2.4 Evolution

As to be expected, the evolution process occurs once the initial software product has been delivered. Once a mobile application is created, it is natural for the initial requirements to mature and evolve [44, 173]. The aim of the evolution process[3], is responsible for correcting faults, improving performance or adaptation of other attributes due to changes brought about by the operating environment or user [174]. The ISO/IEC/IEEE international standard categorises maintenance activities into four classes; *adaptive, perfective, corrective* and *preventive* [173]. Each is summarised below:

- **Adaptive maintenance:** provides enhancements necessary to accommodate changes in the environment in which a software product must operate.
- **Perfective maintenance:** provides enhancements for users brought about by new user requirements.

[3] Also referred to as the maintenance process, throughout the literature

- **Corrective:** the modification repairs the software product to satisfy requirements.
- **Preventive:** the modification of a software product after delivery to detect and correct latent faults in the software product.

Numerous studies surrounding the software evolution process [175–178] have revealed that user requirements are a core problem associated with software evolution and that 75% of all maintenance efforts are expended on adaptive and perfective maintenance activities. As discussed in Chapter 2 healthcare is diverse, and as with software, the healthcare requirements of a HCC can also change. Appy Pie and Microsoft App Studio platforms both have mechanisms to maintain and evolve the applications created by the platform. To maintain consistency with the user the platforms utilise the same design and implementation activities to introduce new functions into the application. Therefore it is critical that from a HCP perspective that the PMAD Framework also is capable of handling changes to HCC's personalised application.

6.2.3 System Related Considerations

As well as the considerations associated with the development of applications, there is also another system related issues highlighted in Chapter 2 that were also considered, these mainly focused around data and maintenance aspects.

6.2.3.1 Data

One of the fundamental problems with modern mHealth technologies is related to data [60]. At the consumer's fingertips, a mobile application enables unique access to a diverse range of tools, resources and utilities. Provisioned correctly they can provide for an efficient mechanism for promoting long-term well-being and independence. A big part of this is

due to the data that a mobile application can generate. However, data with respects to how data is used and protected requires considering: security & privacy, interoperability and how apps interface with the system.

Security and Privacy

Security, privacy and safeguarding of personal data are prominent concerns for the healthcare industry. Although mobile platforms may offer varying degrees of support for safeguarding data, it is the responsibility of the developer for protecting data. In the case of the framework, it is paramount that security, privacy and data management mechanisms are present in both the framework and the applications it produces.

System Interoperability

As discussed in Section 2.2.2.1, one of the challenges with a framework such as this is that healthcare services are heavily invested in their own IT systems. Data produced by the mobile applications provide a valuable insight into the health status of the healthcare consumer. However, this data to be effective in a clinical setting it has to be accessible. Therefore It is critical that the framework is capable of interfacing with existing IT system.

6.2.3.2 Maintenance

As described earlier, maintenance and evolution is a major factor in the engineering process and the framework is no exception. The field of mobile application development is a rapidly developing area and for a framework such as this to be effective, there is a need to maintain components such as the underpinning data sources, ontology and framework components.

6.2.4 Conceptual Design

As a consequence of the observations made and the topics discussed earlier in this section, a conceptual design of the framework was created. The conceptual design shown in Figure 6.4 was based up similar principles of a UML activity diagram and captures the activities and dynamic behaviour of the framework. As can be seen, the conceptual design provides a comprehensive overview of the framework and identifies several characteristics, including the sequence of activities, relationships between components & services and required data sources. For illustration purposes components shown in Figure 6.4 have been colour coded to represent different things: green - *user interface*, purple - *business logic*, range - *data/ information source interface*, red - *data/ Information source* and Grey - *cross cutting service*.

From a HCP's perspective the personalised mHealth application creation process, shown in green, follows a linear development process. Excluding the 'Login' activity which is considered a generic activity, each of the remaining activities are envisaged to facilitate a HCP throughout a specific phase of the personalised mHealth application process. These activities will be represented as graphical user interfaces that will be designed to aid guide and mask the underlying complexities of the system, reducing the need for mobile application development domain expertise.

The objective of the first stage is to determine if the target device (HCC device) can support the chosen functionality. From a HCP perspective the 'Specification Builder', begins by selecting a target device followed by the required functions of the personalised application. This information represents an *app specification*. As discussed in Chapter 2, during this stage in the personalised mobile application development process a HCP would not possess the necessary domain expertise to determine the feasibility of the mobile application. To compensate for the missing domain expertise the framework will

utilise the knowledge contained within the PMAD ontology developed in Chapter 5 to determine if the functionality required can be successfully implemented based upon the healthcare consumers mobile device. The 'specification validation' service is responsible for translating the app specification into a series of description logic queries that will be used to interrogate the PMAD ontology and based upon the knowledge returned determine the feasibility of the personalised mHealth application. If the system concludes that the requirements cannot be achieved under the current conditions the HCP professional is prompted with a message detailing potential solutions to resolve the issue. Otherwise, the HCP can proceed to the second phase, 'App Personalisation'. The interaction between the components mentioned in this paragraph form a vital feature of the framework and a critical step towards achieving the overall aim of this research.

Since each application has the potential to be unique, the framework will be required to dynamically generate forms that a HCP will have to complete in order to personalise the application. This will be the responsibility of the 'Personalised Form Generator'.

The generator will then push these forms to the 'App Personalisation' components that will present the generated forms to the HCP. A wizard will the guide the HCP throughout the personalisation process for each function. Once complete and the input from the HCP has been validated, the complete *app specification* is then sent to the 'App Factory' to be compiled. Leading to the final stage of the process app deployment, which will be the responsibility of the 'Deployment Dashboard' to guide the HCP through the process of deploying the personalised mHealth

Evidently, the conceptual design provided the schematic for a layered architecture of the PMAD Framework which is discussed throughout Section 6.3.

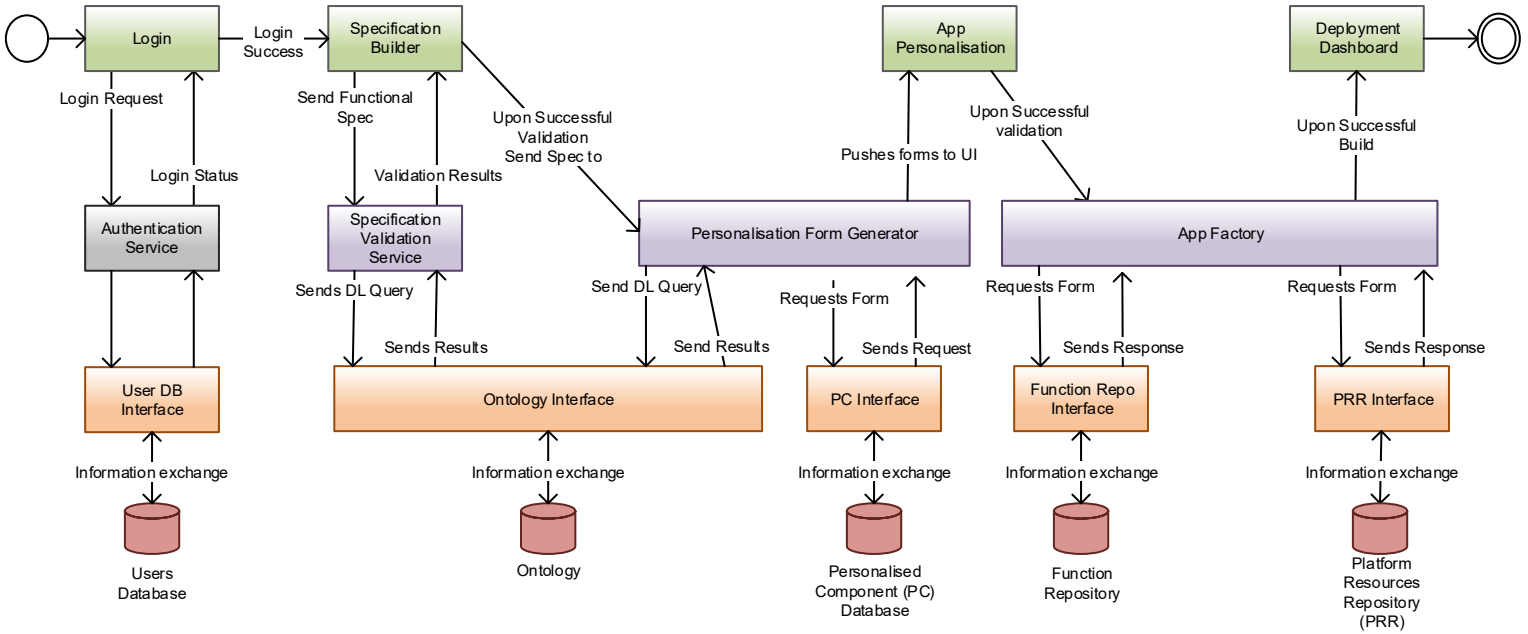


Figure 6.4 Conceptual design of the PMAD framework

6.3 PMAD Framework Architecture

As shown in Figure 6.5, the PMAD Framework is a multi-layer architecture that follows Microsoft's architectural design philosophy[4] [179]. Each of the components and services identified in the conceptual model was organised into one of four layers, *Presentation, Business and Data Access* and *Data Source*. Each layer of the architecture represents components and services that collectively share similar characteristics with regards to their and functionality with the framework. Although the PMAD framework defines many generic components such as those that belong in cross cutting services, data access and data sources. As discussed in Section 6.2.4, the components that are of most significance in achieving the overall of this research are those that provide the framework with its unique capabilities and/or are related specifically to the development of personalised mhealth applications. This includes components that are contained within the 'Personalised mHealth Application Builder'. The 'Specification Validation Service', 'Personalised Form Generator', 'App Factory' and 'App Data Process Service' from the business layer. And finally the 'Mobile App Interface' from the app service layer. The remainder of this section discusses in detail each of the components that reside in each of the layers presented in Figure 6.5.

[4]Note: architecture adopts names and definitions of layers and not the Microsoft specific technologies

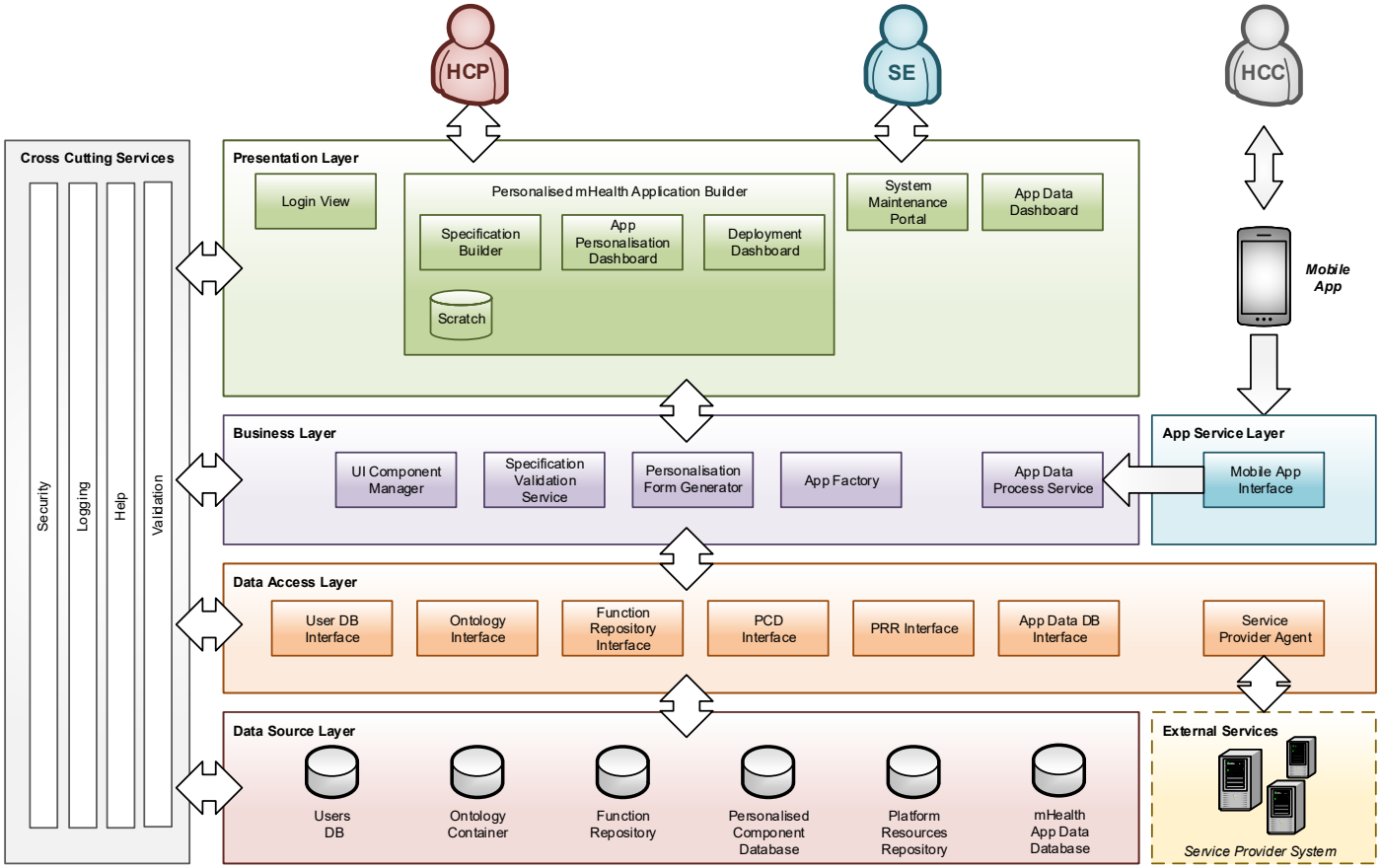


Figure 6.5 PMAD Framework architecture

Presentation Layer:

The presentation layer contains the components and services that are responsible for implementing and displaying the user's interface and also managing user interactions. As discussed in Section 6.2.4, the main objective of this layer is to house user interfaces that are associated with phases of the personalised mHealth application development process whilst keeping the internal mechanics of the framework transparent from the healthcare professional. Therefore, it is vital that the user interfaces are designed to be clean and simple to use.

- **Login View** - Logging into a system is an integral part of security procedures within the PMAD Framework, prior to any interaction with the system. Both HCPs and SEs will be required to authenticate their identity, with respects to the system prior to any further interactions.
- **App Builder** - Is a collection of presentation layer components that collectively enable a HCP to create a personalised mHealth application for a HCC. The app builder consists of four components:
  - **Specification Builder** - The specification builder is responsible for generating the applications requirements specification. The user interface elements should be populated/driven by entities contained within the ontology. The specification builder should also implement sufficient client-side validation rules to ensure that the data inputted is consistent and sufficient, prior to the system validating the app specification.
  - **App Personalisation Dashboard** - The App Personalisation Dashboard (APD) is responsible for guiding the HCP through the personalisation aspect of the application. The APD user interface should be dynamically generated based upon the required functions of the application. Once again client-side validation

rules should also be implemented, to reduce the likelihood of errors occurring. The product of this stage is the design specification of the app, which is then sent to the 'App Factory' to be compiled.

  - **App Deployment** - Once the application has been created by the system, the App Deployment Dashboard will guide the HCP through the installation process.
  - **Scratch Storage** - Temporary storage area used to store transient data throughout the app building process.

- **App Data Dashboard**

  The App Data Dashboard (ADD), enables a Healthcare Professional to view and retrieve data gathered by applications created the framework. The ADD should allow the healthcare professional to search for a particular HCC and view the respective data. The user interface should be dynamically generated based upon the type of data generated by the application.

- **System Maintenance Portal**

  Enables system engineers to maintain the components system.

Business Layer

Services belonging to the business layer represent the core functionality of the system and are concerned with tasks such as retrieval, processing, transformation and management of application or business data [179]. Here houses the frameworks logic that ultimately removes the need for a healthcare professional to possess the domain expertise of a mobile application developer. The main objective of components in this layer is to utilise data in putted from the healthcare professional and received from the underlying data and information sources to provide services that are consumed by components in the presentation layer.

- **User Interface Component Manager** - Responsible for populating various user interface elements with entities from the ontology.

- **App Specification Validation Service** - The App Specification Validation Services (ASVS), is responsible for determining the feasibility of the app specification. This requires managing, interpreting and processing interactions between the components'Specification Builder' and the 'Ontology Interface'.

- - **Personalisation Form Generator** - Responsible for dynamically generating the necessary forms required by the App Personalisation Dashboard.

- **App Factory** - The App factory is responsible for interpreting the design specification and compiling the application.

- **App Data Service** - Responsible for retrieving, managing and processing data from mobile applications created by the framework.

App Service Layer

The app service layer consists of a single outward facing interface, that enables mobile applications created by the framework to send data collected by functions to the system.

Data Access Layer

Components that reside in the data access Layer are interfaces designed to handle interactions between the Business Layer services and the underlying *Data Sources* or *External Services*. Components on this layer can be categorised in one of two categories.

- **Data Access** - provide functionality for accessing the data hosted within the system boundaries.
  - **User Database Interface** - handles interactions between the 'User Database' and business layer services
  - **Ontology Interface** - handles interactions between the 'Ontology Container' and business layer services.
  - **Database (PCD) Interface** - handles interactions between the 'Personalised Component Database' and 'Personalisation Form Generator'.
  - **Function Repository Interface** - handles interactions between the 'User Database' and 'App Factory'.
  - **Platform Resources Repository(PRR) Interface** - handles interactions between the 'Platform Resources Repository' and and 'App Factory'.
  - **App Data Database Interface** - handles interactions between the 'Platform Resources Repository' and 'App Data Process Service'.
- **Service Agents** - Are interfaces that handle and isolate the idiosyncrasies of external systems [179].
  - **Service Provider Agent** - The Service Provider Agent (SPA) is a configurable bespoke interface to manage the communication and mapping of data between the PMAD Framework and the existing system or systems in use by the service provider.

Data Layer and External Services

The data layer represents data sources that are hosted within the boundaries of the framework. Whereas external Services represent systems/services already in use by the healthcare service provider. Each data source is described below in Table 6.1.

Table 6.1 Description of data sources

Data Source	Description
User Database	Database consisting of users of the framework.
Ontology Container	Contains the PMAD Ontology described in Section 5.6.
Personalised Component Database	Database that describes (from a data perspective) each personalisation components within the ontology.
Function Repository	Centralised repository of functions supported by the framework.
Platform Resources Repository	Centralised repository of mobile platform development resources.
App Data Database	Centralised database of HCC data collected from applications created by the framework

Cross Cutting Services

Cross cutting services represent common centralised functionality that have an effect on the entire framework. These component support operations such as security, logging, validation and help.

6.4 Summary

In summary, this chapter describes the final contribution of this thesis. The framework presented in this chapter describes the fundamental components an ontology-driven approach to personalised mHealth application development. The PMAD Framework is designed to utilise knowledge contained within the ontology to enable healthcare professionals to create personalised mHealth applications on-demand for healthcare consumers. The framework has also been extended incorporating additional components that address limitations of existing end-user programming solutions, mainly extendibility and interoperability.

Chapter 7

Evaluation

The goal of the penultimate chapter of this thesis is to evaluate both the PMAD Ontology and PMAD Framework and assess if they satisfy the requirements of their respective objectives. The chapter begins with an overview of the evaluation process. It is worth reiterating that the evaluation process does not include the consultation of healthcare professionals, since the purpose of this research was to design a theoretical framework that defined the main architectural components thus the evaluation is designed to evaluate the potential pitfalls and consistency of the PMAD ontology model as well as competence from an implementation perspective of both the PMAD ontology and PMAD framework. Therefore, the evaluation of consists of three phases. The first two focuses specifically on evaluating the PMAD Ontology from two perspectives. The first stage evaluates the model using the OntOlogy Pitfall Scanner. The second evaluates the 'consistency' of the ontology. The final phase evaluates the competence of both the PMAD Ontology and PMAD Framework from an implementation perspective. This chapter concludes with a summary of the key and influential evaluation components discussed throughout this chapter.

7.1 Evaluating the Model

OntOlogy Pitfall Scanner (OOPS!) is a web-based ontology evaluation tool created by Villalon *et-al* for detecting common pitfalls in ontologies [144]. The OOPS! is a recent tool developed based on a comprehensive assessment of ontology evaluation literature. Villalon *et-al* identified a total of 41 common pitfalls, 38 of which can be detected automatically by the OOPS! tool. Figure 7.1 shows an overview of the pitfall catalogue, full details of each pitfall are documented here [145].

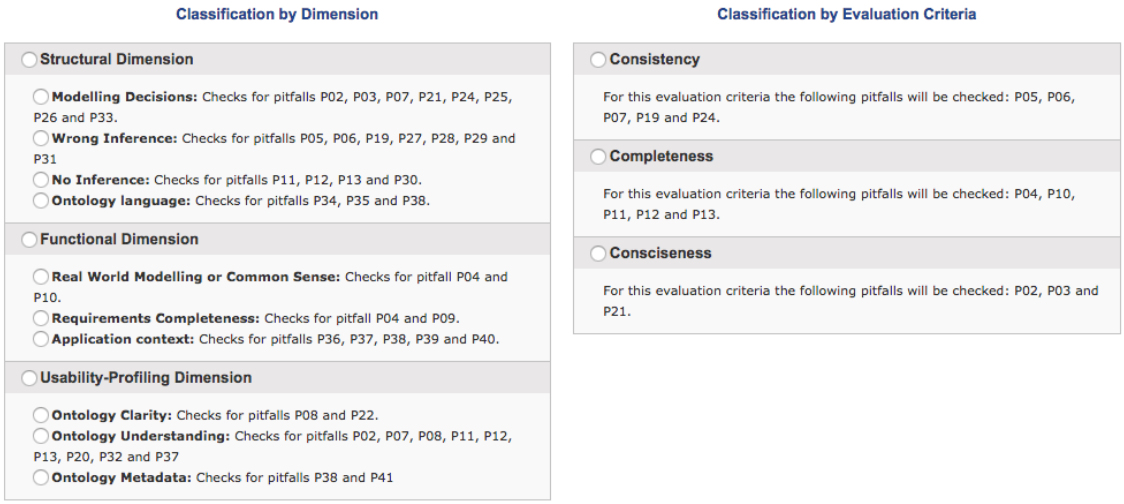


Figure 7.1 Overview of the OOPS! catalogue of pitfalls

Each pitfall within the catalogue has the following information: identifier, title, description, elements affected and importance level. The first three are self-explanatory. The 'elements affected' returns a list of affected elements associated with the particular pitfall. This may include specific elements of the ontology i.e classes, object properties etc. or the ontology its self. The 'importance level' categorises each potential pitfall into one of the three categories below based upon the impact it may have on the ontology.

- **Critical:** It is crucial to correct the pitfall. Otherwise, it could affect the ontology consistency, reasoning and applicability, among others.
- **Important:** Although not critical for ontology's function, it is important to correct this type of pitfall.
- **Minor:** It does not represent a problem. However, correcting it makes the ontology better organized and user friendly.

It is worth noting at this stage that not all pitfalls are applicable, as some of them depend on the domain being modelled or the specific requirements or use case of the ontology [180, 145]. For those pitfalls that can be detected automatically, their applicability as criteria to evaluate the ontology is determined accordingly. The applicability of those pitfalls that require manual intervention is discussed in the sections that follow.

This space has been intentionally left blank for presentation purposes

7.1.1 Manual Intervention Pitfalls

Prior to uploading the PMAD Ontology to the OOPS! tool, It was worthwhile assessing each of the manual pitfalls to determine if they are applicable or not. Table 7.1 summaries the results associated with each pitfall. Further discussion and justification of the results presented in the table below are discussed throughout the remainder of this section.

Table 7.1 Summary of the results for each for each pitfall that required manual intervention.

Pitfall ID	Applicable	Present
P01	Yes	No
P09	Yes	No
P14	Yes	No
P15	No	-
P16	Yes	No
P18	Yes	No
P23	Yes	No

P01 Creating Ploysemous Elements

The first manual pitfall identifies ontology entities whose identifier has different interpretations within the ontology but is used to denote more than once conceptual idea or property [145]. This pitfall was acknowledged and was discussed in detail Section 5.1.3. To prevent this from occurring involved careful considerations surrounding the entities within the domain. Section 5.1.2 demonstrated how semantic models can be used as mechanisms to address various challenges. The creation of the glossary of terms as discussed in Section 5.4.2 aimed to catalogue, control and remove the potential for ambiguity within the vocabulary used. The glossary was a vital step in the development of the PMAD Ontologyto prevent this pitfall being present. For further clarity and readability of the

ontology, annotation properties were used to appended descriptions to entities. Therefore this pitfall is not considered to be present in the PMAD Ontology.

P09 Missing domain information

Part of the information needed for modelling the intended domain is not included in the ontology. This pitfall relates to the 'completeness' of the ontology and requires considering 2 factors [145]. The first factor requires determining if the ontology meets its intended purpose. As stated in Section 1.2, the objective of the ontology was to establish a suitable model that encapsulates the necessary and sufficient knowledge. During the initial conception of the ontology, the purpose and scope of the ontology were defined. As part of this process, a series of competency questions were established. As explained earlier the competency questions represent a series of questions that the ontology must be capable of answering to be considered competent at tackling the problem it has set out to solve. To assess if the ontology is competent at answering these questions a small-scale implementation of the framework was developed. The outcome of the first influences the second factor. If the ontology is not capable of answering the competency questions, what knowledge can be / should be added to the ontology to make it more 'complete'. Therefore this pitfall is considered to be applicable to this work and is discussed in further detail in Section 7.3.

P14 Misusing of the universal quantifier restriction:

This pitfall consists in using the universal restriction ($\forall$) as the default qualifier instead of the existential restriction ($\exists$). A description of both universal and existential restrictions and demonstration of their correct uses is discussed in detail in Section 5.4.5. Therefore, the PMAD Ontology is considered not be associated with this particular pitfall.

P15 Using "some not" in place of "not some"

The pitfall consists in using a "some not" structure when a "not some" is required. This is due to the miss-placement of the existential quantifier ($\exists$) and the complement operator (

¬) [145]. This issue pitfall is not applicable to this particular ontology as the complement operator is not used to describe or define classes within the PMAD Ontology.

P16 Using a primitive class in place of a defined one

This pitfall implies creating a primitive class rather than a defined one in case automatic classification of individuals is intended [145]. Again, this issue is not applicable to this ontology. As discussed in Section 5.4.5.2 and Section 5.5.4, all classes that are not a subclass of the `ValuePartition` are defined classes and are defined utilise necessary and sufficient conditions, meaning that inferred hierarchy occurs naturally and the automatic classification of individuals is feasible. Therefore, this pitfall is not present within the PMAD Ontology.

P18 Over-specialising the domain or range

Villalon *et al.* describe this pitfall as defining a domain or range that is not general enough for a property. This pitfall is applicable to the PMAD Ontology. However, a modelling decision was made not to restrict the domain of the object property to a particular class, as this can result in incorrect inferences made by the reasoner. Instead, a single restriction was applied to the range of each object property. This enabled the hierarchy to be inferred based on the necessary and sufficient conditions, as discussed in Section 5.4.4.3. Therefore, the pitfall linked to the over-specialising of the domain or range of properties is not considered to be present within the PMAD Ontology.

P23 Duplicating a datatype already provided by the implementation language

This pitfall exists if a class and its corresponding individuals are created to represent datatypes that already exist in the implementation language. Villalon *et al.* provide the following example 'An example of this type of pitfall is to create the relationship "isEcological" between an instance of "Car" and the instance "Yes" or "No", instead of creating the attribute "isEcological" whose range is Boolean' [180]. Again, this particular pitfall was acknowledged during the knowledge extraction phase, see Section 5.4.1. Therefore, this pitfall is applicable but is considered to be not present in the PMAD Ontology.

7.1.2 Automatic Pitfalls

The PMAD Ontologywas uploaded to the OOPS! pitfall scanner which returned an evaluation report. A complete copy of the OOPS! evaluation report can be found at Appendix C.3. Figure 7.2 provides a summary of the potential pitfalls that were identified by the OOPS! tool. As can be seen, there are 3 minor and 3 important pitfalls each is discussed in detail below.

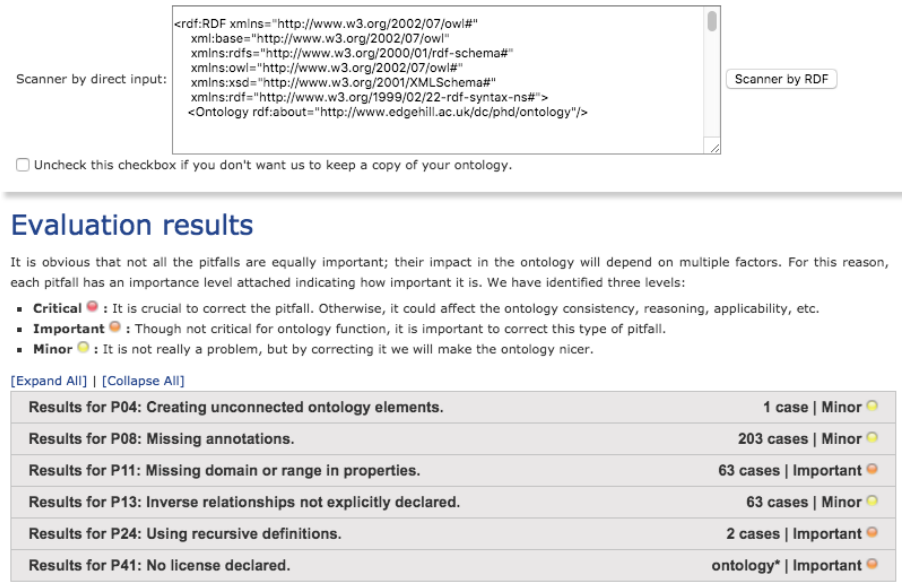


Figure 7.2 OOPS! evaluation summary of results

7.1.2.1 Minor Pitfalls

P04: Creating unconnected ontology elements

This pitfall identifies entities within the ontology that are isolated and have no relation to the rest of the ontology. Of the 203 entities within the PMAD Ontology, one class, `ValuePartition` was identified to have this pitfall. However, as stated during the ontology's development, the `ValuePartition` serves a unique role within the ontology. The `ValuePartition` class was added to the ontology as part of Rector's modelling approach and its sole purpose is used to partition primitive concepts from defined concepts. From a modelling perspective helps increase the clarity of the ontology, maintains modularity and allows concepts to be defined in a significant amount of detail. Overall the `ValuePartition` class is supposed to be isolated from other entities within the ontology and therefore this pitfall can be regarded as a false positive.

P08: Missing Annotations

The missing annotation refers to an entity that fails to provide a human-readable annotation attached to it. Although upon initial inspection the results seemed alarming, further inspection revealed that each entity within the PMAD Ontology required both a 'label' and 'comment' annotation properties in order to not be associated with this particular pitfall. Hence the surprisingly large number of entities associated with this pitfall. To put this into perspective the results of this particular metric were processed and presented in Figure 7.3.

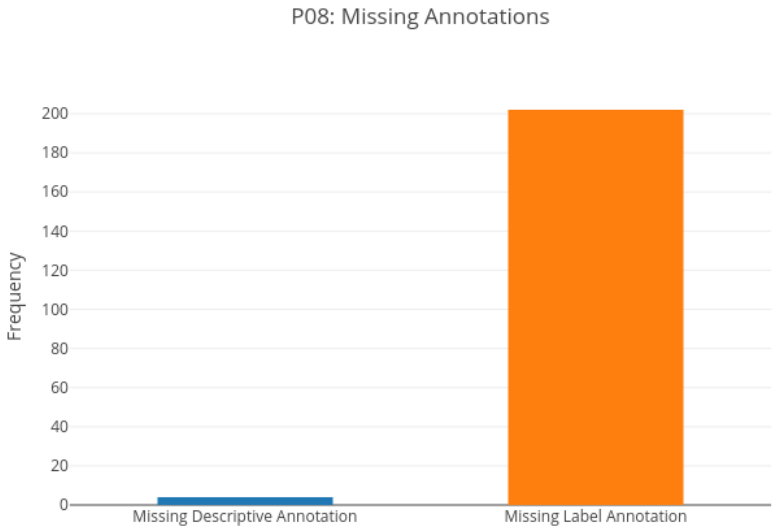


Figure 7.3 P08: Missing Annotations: Graph to show a comparison of entities missing description annotations vs those missing label annotations

Immediate attention was drawn to the 4 entities (Accelerometer, Communication, Inform and requiresMicrophone) that are missing a 'descriptive' annotation property. As discussed in Section 5.5.2.1, annotation properties were used to provide both ontology engineers and users of the PMAD Framework with a concise and unambiguous description of entities within the ontology and thus increasing the overall readability in the process. Therefore, these oversights made during the implementation activity required addressing to maintain consistency with the rest of the entities within the ontology. The hotfix was applied to the ontology and the test for this pitfall was run again Figure 7.4. As can be seen in Figure 7.4, all entities within the ontology now have a descriptive annotation property

With regards to the use of annotation properties for 'labels' of entities within the PMAD Ontology. The readability of the ontology was considered very early on in its design and development. The first activity in the development process identified and defined several guidelines that included design criteria and naming conventions for entities within

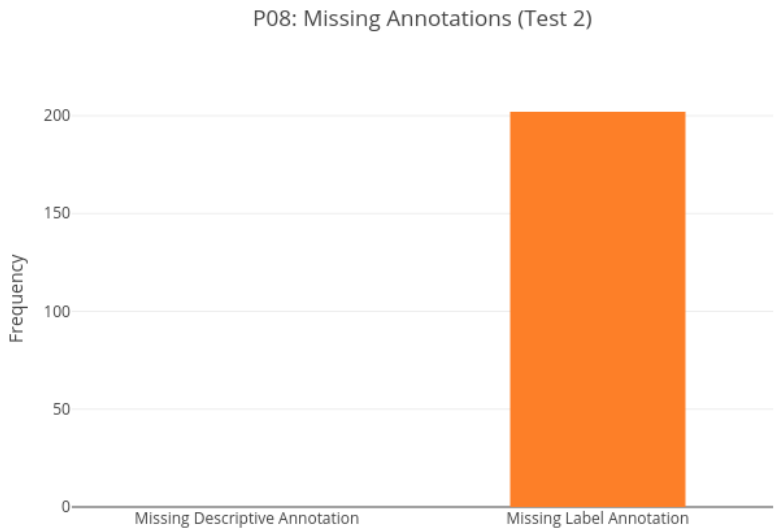


Figure 7.4 P08: Missing Annotations (test 2): Graph to show a comparison of entities missing description annotations vs those missing label annotations

ontology. As shown during the first stage of the evaluation, all the identifiers for each of the entities within the PMAD Ontology were named according to the rules specified in the naming convention guidelines. In addition, the creation of the glossary of terms identified a name for each entity, but also highlighted any acronyms and synonyms that were also associated to it. The identifier of entities within ontology are considered to be axiomatic as each entity was carefully reviewed. Furthermore, all entities are represented using natural language. As a result, the entities missing 'labels' has been disregarded. Overall this pitfall highlighted four oversights within the PMAD Ontologythat were immediately corrected, therefore this particular pitfall is considered no longer present within the PMAD Ontology.

P13: Inverse relationships not explicitly declared

This pitfall is identified by OOPS! if there is any relationship [1] that do not have an inverse relationship explicitly defined within the ontology. With regards to the PMAD Ontology, this issue was briefly discussed in Section 5.4.4.2, the decision not to include an inverse relationship was because there are no individuals modelled within the domain and including

[1] except those that are defined as symmetric

inverse object properties for class definitions would not provide any further or substantial knowledge to the ontology. Therefore this pitfall is considered not impact the PMAD Ontology.

7.1.2.2 Important Pitfalls

P11: Missing Domain or Range in properties

Although highlighted as a pitfall, this is an expected consequence of a modelling decision. As discussed earlier on in this chapter and in detail in Section 5.4.4.3, the decision not to restrict the domain of an object property was to prevent incorrect inference being made as a consequence of domain restriction, but rather rely solely on necessary and sufficient conditions so that a reasoner could infer subsumption. As discussed earlier on in this chapter, the inferred hierarchy produced by the reasoner matched its intended design. Furthermore, each object property within the PMAD Ontology has a restriction placed on the range as shown in Figure 7.5. Therefore, this pitfall can be considered as a false positive.

P24: Using recursive definitions

This pitfall was present within the PMAD Ontology, as there were two classes (`ShoppingList` and `MedicationTracker`) that used its self in its own definition. In both cases, the class was used as a property filler for the `hasFunctionLogic` object property. This was not intentional. Again, this was a minor mistake made during implementation. This modelling error was immediately corrected, to remove this pitfall from the ontology. As can be seen in Figure 7.6, running an additional test for this particular pitfall shows that the pitfall is no longer present within the PMAD Ontology.

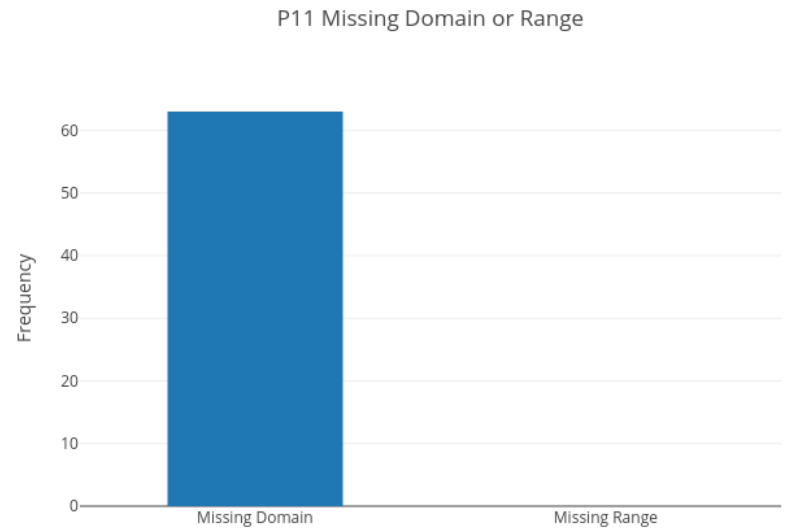


Figure 7.5 P11: Missing 'Domain' or 'Range' in properties

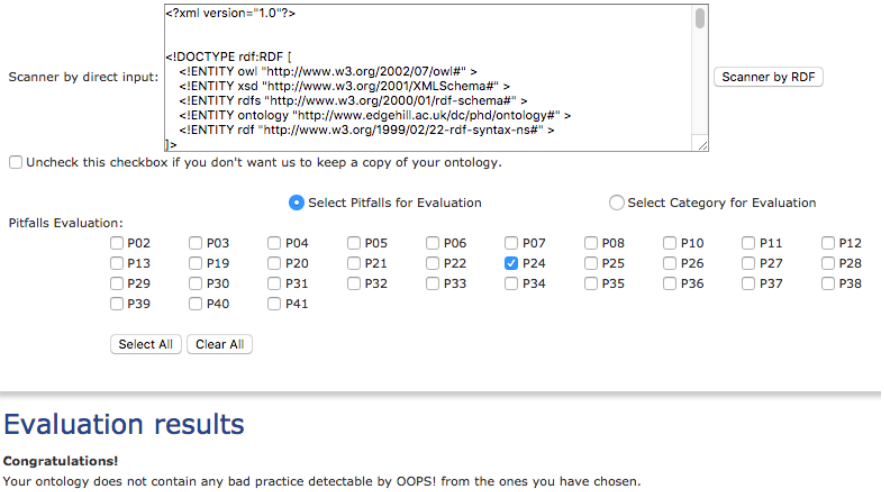


Figure 7.6 P24: second test result

P41: No Licence declared

Although identified as important, this pitfall does not directly impact the model but rather governs the use of it. Therefore, this pitfall was disregarded from the evaluation discussion.

7.1.3 Naming Convention Compliance

Section 5.2.3.1, naming convention rules were established at during the beginning of the ontology's development. This was for two reasons: to promote consistency across entities within the ontology since OWL does not specify conventions and to improve overall clarity & readability of the ontology. The following test procedure was conducted for each type of the entity within PMAD Ontology.

Naming convention testing procedure:

1. Extract all <*entity type*> from ontology using the OWL API.
2. For each entity assess its compliance against the rules specified in the naming convention guidelines.
3. Present and discuss results.
4. If necessary, apply hotfixes to ontology.

7.1.3.1 Results and Discussion: Naming Convention Compliance

The results from the naming convention compliance tests are discussed and presented below.

- **Classes:** Protégé ontology metrics reported that there is a total of 138 classes within the PMAD Ontology. The naming convention guidelines state that all classes within the ontology must follow 'CamelCase' notation. As can be seen in Figure 7.7 all classes within the ontology are compliant with the rule defined in Section 5.2.3.1.

- **Object Properties:** Protégé ontology metrics reported that there are a total of 63 object properties within the PMAD Ontology. The naming convention guidelines state that all classes within the ontology must follow 'camelCase' notation. As can be seen in Figure 7.7 all object properties within the ontology are compliant with the rule defined in Section 5.2.3.1.

- **Annotation Properties:** excluding the built-in annotation properties, there was only one further annotation property (`description`) created during the development of the ontology. As can be seen in Figure 7.7 all annotation properties within the ontology are compliant with the rule defined in Section 5.2.3.1.

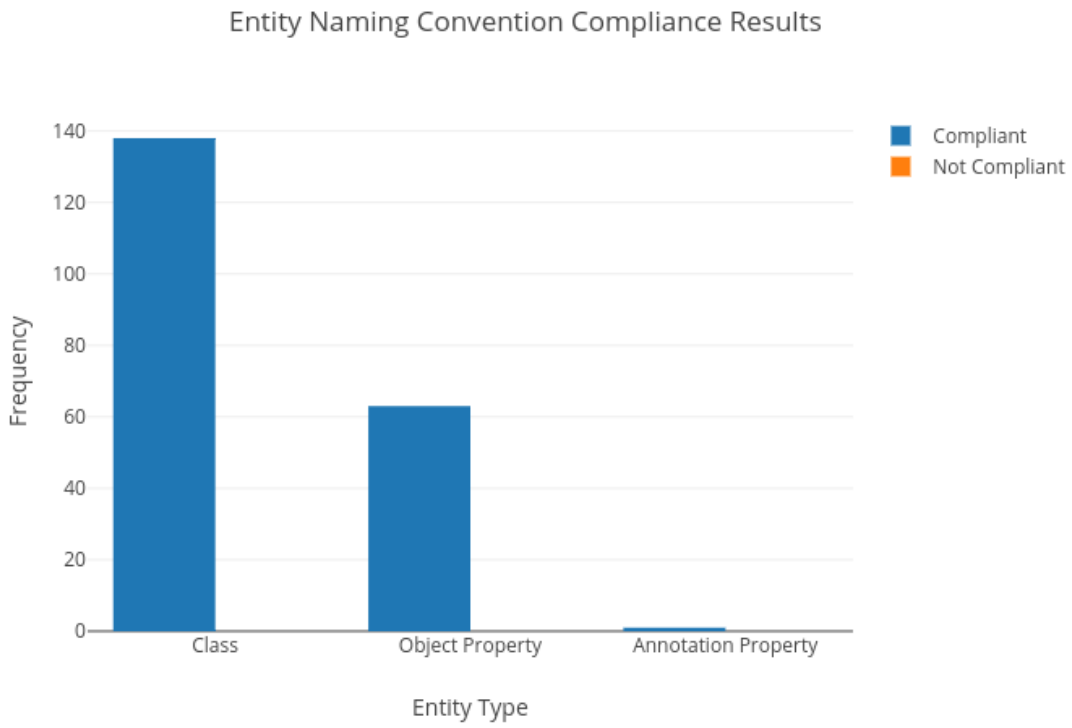


Figure 7.7 Naming convention compliance results: classes, object-properties and annotation-properties

7.1.4 Model Synopsis

The PMAD Ontology was evaluated using a combination of manual and automated techniques to evaluate the presence of potential common pitfalls within the model. To summarise five[2] out of the seven pitfalls that required manual intervention was considered applicable but not present within the ontology. Furthermore, the OOPS! identified the presence of six potential pitfalls. As discussed PO4, P11 and P13 were to be expected, the presence of these pitfalls was detected due to modelling choices described in Sections 5.2.3, 5.4.4.2 and 5.4.4.3 respectively. Therefore, are they are not considered as pitfalls, but rather as consequences of modelling choices. However, PO8 and P24 identified a total of 6 minor errors. These errors were the result of oversights during implementation and were immediately rectified. P41 was disregarded as a pitfall as it does not have an impact the model but governs the use of it. The final test revealed that all entities within PMAD Ontology were compliant with the naming convention guideline established in Section 5.2.3.1. Overall, for the reasons discussed above, the PMAD Ontology model is considered to be free from all pitfalls (manually and automatic) that can be identified by the OntOlogy Pitfall Scanner, except those highlighted as consequence of modelling decisions.

[2]There were six pitfalls. However, as discussed pitfall P09 required further testing, see Section 7.4

7.2 Evaluating Consistency

Throughout the design aspect of the ontology, several terms have been used to describe specific inferences that a reasoner can make as a consequence of the knowledge described within the ontology. Realistically the correctness of the ontology was continuously assessed throughout the implementation activity, as helped isolate smaller errors as they occurred rather than trying to unravel them at the end. The reasoner used to during the development and evaluation of the ontology was HermiT, more information about the HermiT can be found in Section 5.1.5.2. With respect to the TBox ($\mathcal{T}$), there are four main reasoning tasks associated with a reasoner. These are satisfiability, subsumption, equivalence and disjoint [139, 181, 182]. Each task is described below. Note, in OWL the class `Nothing` represents an empty set.

- **Satisfiability:** C is deemed satisfiable with respect to $\mathcal{T}$ if there exists a interpretation of $\mathcal{T}$ such that the interpretation of C is an non-empty set; otherwise C is unsatisfiable. In other words, satisfiability checks definitions to determine if they make sense or whether they are contradictory of the axioms contained within the ontology.

- **Subsumption:** C is subsumed by D if the interpretation of C is a subset of the interpretation of D with respect to $\mathcal{T}$. To put another way subsumption determines whether a class subsumes another in accordance to their generality to form a taxonomy.

- **Equivalence:** C and D are considered equivalent with respect of $\mathcal{T}$ if the interpretation of C and the interpretation of D are equal.

- **Disjointness:** C and D are disjoint with respect to $\mathcal{T}$ if the intersection of their interpretation results in an empty set.

As shown below, each of these reasoning tasks can be reduced to concept subsumption [139]. Let C and D represent concepts (classes).

$$
\begin{aligned}
C \text{ is } \textit{unsatisfiable} &\leftrightarrow C \text{ is subsumed by } \bot \\
C \text{ and } D \text{ are } \textit{equivalent} &\leftrightarrow C \text{ is subsumed by } D \text{ and } D \text{ is subsumed by } C \\
C \text{ and } D \text{ are } \textit{disjoint} &\leftrightarrow C \text{ and } D \text{ is subsumed by } \bot
\end{aligned}
$$

It is from this perspective, that the correctness of the knowledge encoded within the PMAD Ontology will be assessed. The ontology will be considered correct, if and only if the ontology passes each of the tests that follow. Note, equivalence has been excluded from these tests as there are no classes within the ontology that are semantically equivalent to one another.

7.2.1 Satisfiability

As mentioned earlier, testing the satisfiability of the PMAD Ontologywas a continuous activity throughout implementation. Regarding the design of the ontology If a class is deemed unsatisfiable, the class is subsumed by the class`Nothing` and the Protégé ontology editor highlights the entity in red as demonstrated in Figure 7.8.

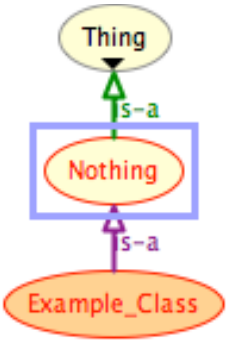


Figure 7.8 Demonstration of an inconsistent class in Protégé

There were some instances throughout the development of the PMAD Ontologythat a class would be deemed unsatisfiable. These were not due to the design of the ontology,

but rather an impact of user error. These instances often occurred during the definition of larger classes and were caused by relying on the auto-complete feature built into Protégé 's class expression editor. These minor setbacks were quickly addressed using Protégé 's built-in inference explanation tool.

With regards to the initial version of the ontology, the HermiT reasoner determined no classes were contradictive of the axioms contained within the PMAD Ontology. To verify this claim, the description logic query '*Nothing*' was executed. As presented in Figure 7.9, the result shows that there are no classes equivalent to the class `Nothing`.

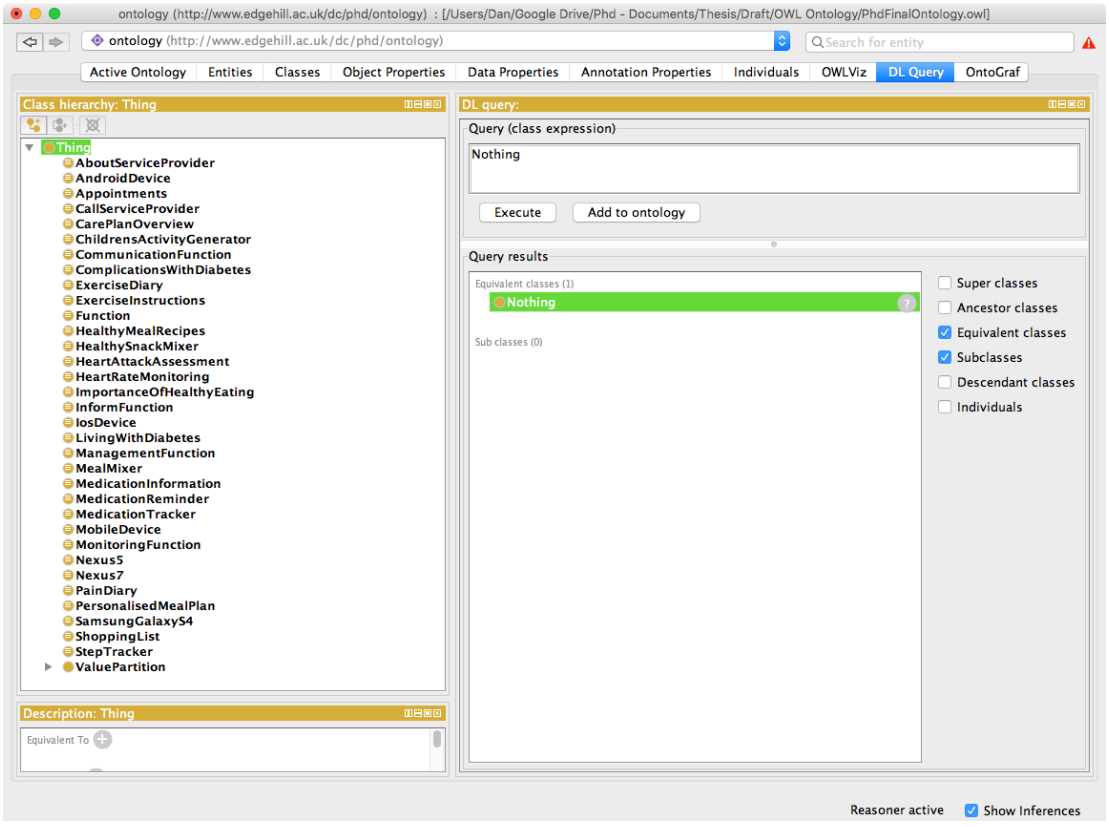


Figure 7.9 Satisfiability verification using a description logic query

7.2.2 Subsumption

As discussed in Section 5.5.4, based upon the necessary and sufficient conditions the HermiT reasoner created the inferred hierarchy. Although the knowledge encoded within the ontology is satisfiable. It was also important to check whether the inferred class hierarchy was also correct[3]. Protégé conveniently presents the classification results in a simple list. This list contains a total of 32 inferred axioms as a consequence of the necessary and sufficient conditions of the defined classes. This list represents the inferred hierarchical structure of the ontology and can be processed against the model created during the capturing phase.

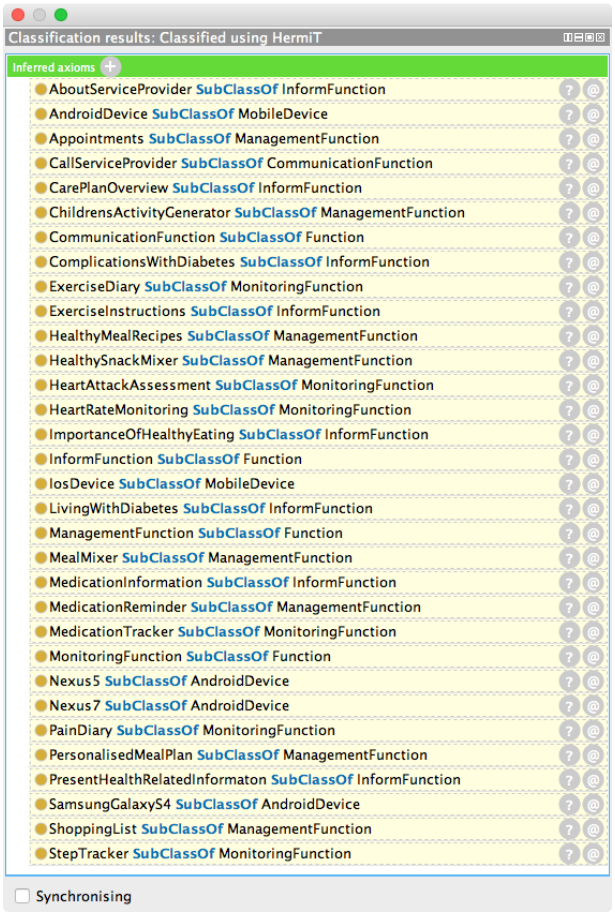


Figure 7.10 Protégé classification results using the HermiT reasoner

[3]The value partition hierarchy was auto-generated and its structure is not impacted by the reasoner. The structure of the value partition was tested and verified to be correct during implementations as discussed in Section 5.5.1

Elements within both lists were compared using a simple python script that determined two potential outcomes: True or False. True indicates that the axiom is present in both lists, whereas False indicates there is an error present in the inferred hierarchy. The ontology with respects to subsumption will be considered correct, if all results are returned by the script are 'True'. As this indicates that the class hierarchy has been inferred correctly.

For presentation purposes, the complete listing of the results has been omitted from this section but can be found at Appendix C.1. Figure 7.11 is a summary of the results. As can be seen, all the classification axioms present within the inferred hierarchy, therefore the PMAD Ontology is considered to be complete from a subsumption perspective.

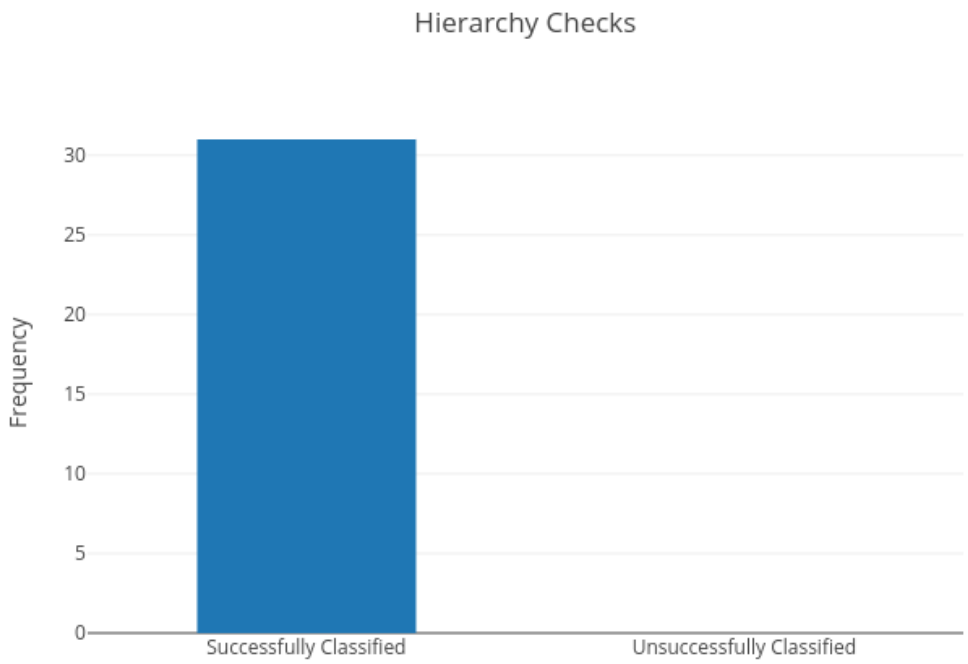


Figure 7.11 Bar chart to show the results from the class hierarchy checks

7.2.3 Disjointness

Currently, testing of the correctness of the PMAD Ontology has shown that the knowledge encoded is satisfiable and that subsumption has been inferred correctly. The final tests aim to demonstrate the presence of the disjoint axioms. During the design of the ontology, several classes utilised the 'DisjointWith' axiom. To demonstrate the presence of the disjoint axioms, description logic queries were used. As stated at the beginning of this section, the disjointness between two or more classes can be reduced by concept subsumption, by checking if the intersection between them results in them begin subsumed by the `Nothing`. Essentially the description logic query creates an anonymous class that consists of the class under review and the intersection[4] between the classes that it was designed to be disjoint with. An example of the description logic query used to test the presence of the disjointwith axiom between `Function` and `MobileDevice` classes is shown in Figure 7.12. Each test has two potential outcomes as shown in Table 7.4. The ontology will be considered correct if all the tests yield a pass result.

Again, for presentation purposes, the complete listing of the results has been omitted from this section, but details for each of the 96 tests and results can be found at Appendix C.2. Figure 7.13 provides a summary of the results. As can be seen, each of the queries yields successful results (pass), therefore the PMAD Ontology is considered to be complete from a disjoint axiom perspective.

Table 7.4 Disjoint: description logic query outcomes

Outcome	Description	DL Result
Pass	The use of the disjoint axiom is present.	Anonymous class is subsumed by `Nothing`
Fail	The use of the disjoint axiom is not present.	Anonymous class is not subsumed by `Nothing`

[4] The intersection between classes uses the *and* operator

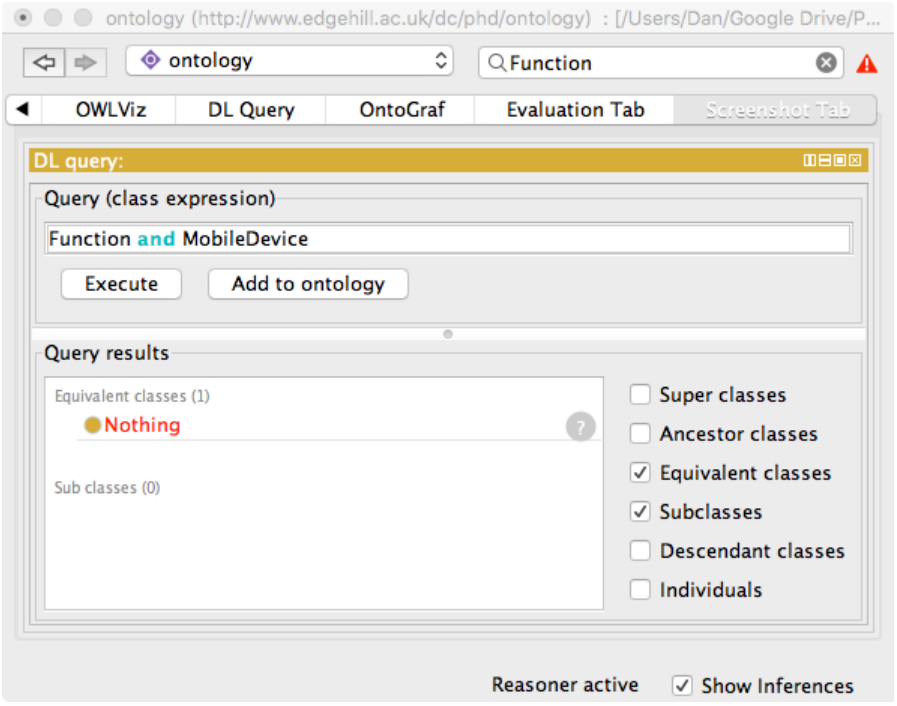


Figure 7.12 Testing disjointness using description logic queries

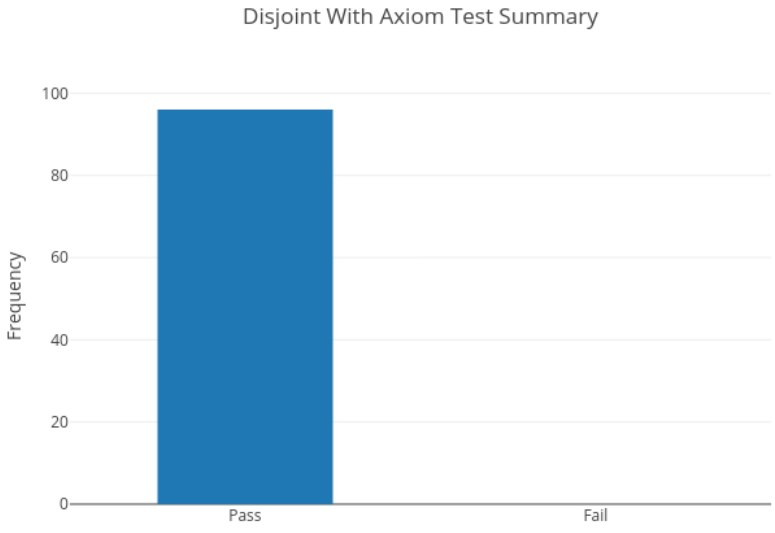


Figure 7.13 Disjoint test results

7.2.4 Consistency Synopsis

The correctness of the ontology has been evaluated from three perspectives. Testing the satisfiability of the ontology deemed that all concepts defined within it were satisfiable based upon the axioms contained within the ontology. The second test subsumption determined that the inferred class hierarchy created by the HermiT reasoner matched the model of the class hierarchy created during the design of the ontology. The final test provided evidence of the presence of the disjoint axioms within the ontology. Overall, the outcome of each test described in this section collectively provided sufficient evidence to suggest that the knowledge contained within the ontology has been modelled correctly.

7.3 Evaluating Competence

As the name of this section suggests the final phase in the evaluation aims to determine if both the PMAD Ontology and PMAD Framework are competent in solving the problems they are designed to address. As stated in Chapter 1, the overall aim of this research is to produce an ontology-driven framework that enables healthcare professionals to develop personalised mHealth applications, without the need for intervention from mobile application developers. From the ontology's perspective, a series of competency questions were established during the initial stages of the ontology's development. As explained in Section 5.3.1 they represent questions that the ontology must be capable of answering to in order to be considered competent. It is these questions (see, Table 5.11) that will serve as the competency criteria for the ontology. Where as the framework provides the facility to personalise mHealth applications. As discussed in Sections 2.3.2 and 4.3.6, personalisation of a mHealth applications in the context of this work consists of two elements: the first element focuses on the selection of functionality required by the healthcare consumer and the second on the tailoring of the chosen functions to the requirements of the healthcare consumer. Although the framework presented in this thesis is conceptual, core components can be implemented that integrate the ontology and simulating the key functionality of the framework. Enabling the competence of both the PMAD Ontology and PMAD Framework. For clarity, the application is referred to from this point forward as the *PMADs* (Personalised Mobile Application Development simulation). The remainder of this section provides an overview of PMAD's, evaluation procedure followed by a discussion of the results.

7.3.1 PMADs Overview

PMADs was implemented using the Java programming language and utilised the OWL API to interface with the PMAD Ontology. PMAD's aim is to simulate some of the functionality from the specification builder and the app personalisation activities discussed in Sections 6.2.4 and 6.3. As shown in Figure 7.14, PMAD's consists of a simple user interface that is driven by entities contained within the ontology. The user is required to select a `Mobile Device` and chose the required `mHealth Functions` from the list provided. PMAD's will then utilise the knowledge contained within the ontology to determine the feasibility of the mHealth application based upon the selected `Mobile Device`. Note, the PMAD's will not create a fully operational mobile application but rather produce a design specification report (DSR) for the intended mHealth application. An example of a DSR is shown in Figure 7.14. The information returned by the DSR coincidences with overall aim of this research and the competency questions presented in Section 5.3.1.

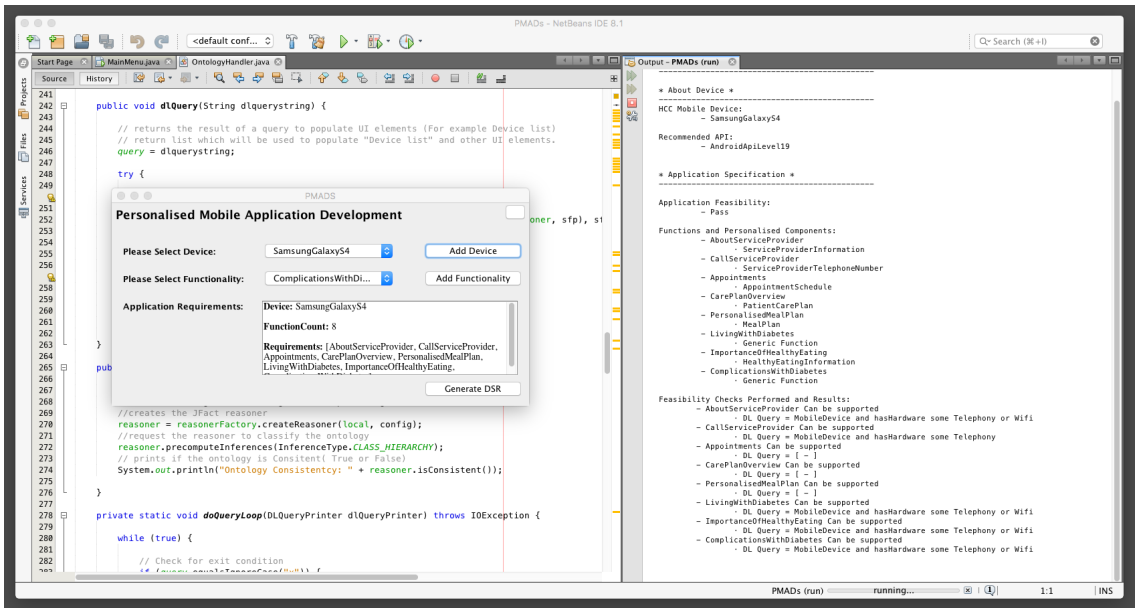


Figure 7.14 Screenshot showing PMADs and DSR example

The information contained in the DSR is summarised below:

- **About Device:** Details of the Healthcare consumers device which includes:
  - **Mobile Device:** The selected mobile device of the HCC from the use case.
  - **Recommended API Level:** The minimum API Level of the mobile device.
- **Application Specification:** Details of the personalised mHealth application, that includes:
  - **Application Feasibility:** The overall feasibility of the personalised mHealth application:
    - ***Pass:*** The mobile device selected is capable of supporting the all of the functions.
    - ***Fail:*** The mobile device selected is capable of supporting some or none of the functions.
  - **mHealth Functions and Personalised Components:** A list of required functions and their personalised components (if available).
  - **Feasibility Checks and Results:** Provides details of the feasibility checks performed and their results.

7.3.2 Competence: Evaluation Procedure

In order to evaluate the competence of the PMAD ontology and framework, required using the use case scenarios created during the development of the taxonomy. Each use case scenario is subjected to the same evaluation procedure described below:

1. Manually evaluate the feasibility of the personalised mHealth application (this will serve as the expected result).
2. Using PMADs...
   (a) Select the HCC <mobile device> from use case scenario.
   (b) Select <mHealth functions> required by the HCC from the scenario.
   (c) Press ‘Generate DSR’ button.
   (d) Archive DSR.
3. Evaluate results.

7.3.3 Results

Using the data collected from the previous section, this section presents a discussion of the competence of both the ontology and framework.

Technical Feasibility

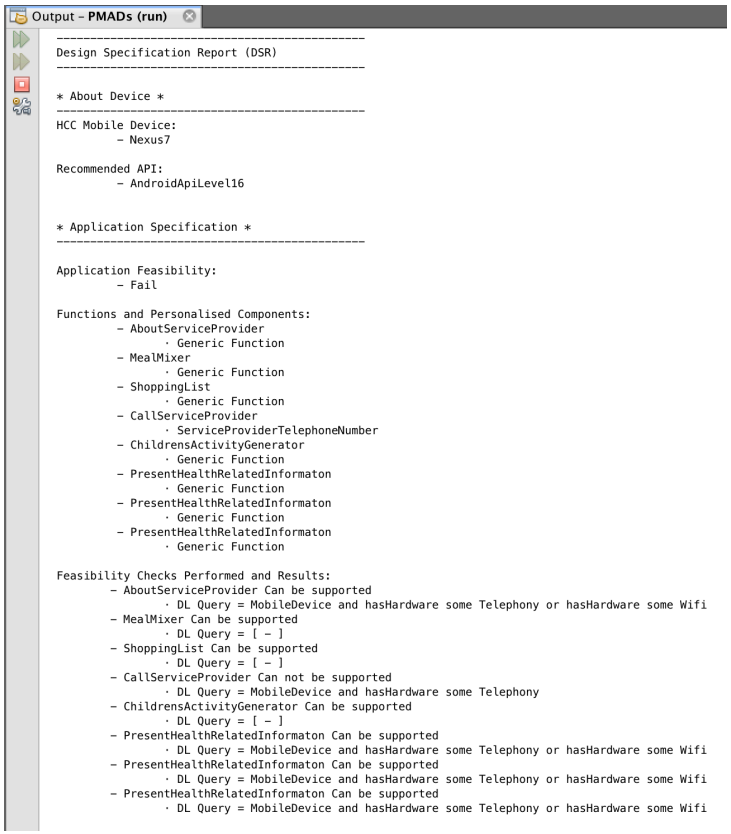
There are two competency questions that are related to technical feasibility of the PMAD framework. The first competency question (CQ1) focuses on determining the individual feasibility of each function required by the personalised mHealth application documented in each of the use case scenarios. From the perspective of a healthcare professional, the 'Specification Builder' in PMADs requires the selection of the healthcare consumers mobile device and the required mHealth functions. Whilst remaining transparent to the HCP, PMADs determines the feasibility of each mHealth function by extracting and parsing the functions class definition identifying if the particular function has specific hardware requirements. If hardware requirements are identified, a description logic query is automatically generated based on how they are represented in the class definition. For example Figure 7.15a shows two different examples of differing description logic queries. The function `CallServiceProvider` requires the presence of 'Telephony' hardware in the healthcare consumers mobile device. Therefore, the query generated reflects the single hardware requirements. Whereas in the case of the function `AboutServiceProvider` which requires a mobile device to either have 'Telephony' or 'Wifi' (or both) hardware in order for it to be feasible, the description logic query generated reflects this. Fundamentally the description logic query asks the ontology if there is a mobile device present within the ontology that has possessed the hardware required by the function. This returns a list of mobile devices classes presents within the ontology that satisfy the query, which is checked to see if the healthcare consumers mobile device is present in the list. If the mobile device appears in this list, we can assume that the mobile device is capable of supporting that

particular function. If it is not in the list, then the assumption is made that the healthcare consumer's mobile device is not capable of supporting that particular function. The result is then stored and the process repeated for the remaining functions.

The second competency question (CQ 2) requires utilising the outcome from the first (CQ 1) to assess the overall feasibility of the personalised mHealth application. The results stored in the list is then checked by the PMADs. If all the functions are capable of being supported by the healthcare consumers device, the personalised mHealth application specification passes the application feasibility, as shown in Figure 7.15a. However, if a function is not capable of being supported by the healthcare consumers mobile device, a flag is triggered which changes the feasibility of the personalised mHealth application specification to *fail* as shown in Figure 7.15b.

With regards to the use case scenarios, the manual feasibility result was compared to that provided by PMADs. The comparison shows that the knowledge contained is capable of being operationalised to determine both the feasibility of all the functions required and the overall application feasibility. Thus satisfying the requirements of competency questions CQ 1 and CQ 2. From a personalisation perspective, these two competency questions also demonstrate the first element of personalisation (selection of specific functions) in the context of this work.

(a) Paul: Use case scenario (Pass)



(b) Emma: Use case scenario (Fail)

Figure 7.15 DSR examples

Application Development

Competency questions CQ 3 and CQ 4 focus specifically on development choices relating to the personalisation process. In order to answer this CQ 3 requires identifying the 'personalised components' of the selected functions. This is achieved in PMADs by again extracting the class definition and parsing it to identify the personalised components of each function selected and exporting them in the DSR as shown in Figure 7.15. As shown in the Figure 6.4, this process from occurs from a healthcare professionals perspective during the transition between the 'Specification Builder' and 'App Personalisation'. In the context of the framework, the personalised components extracted from the class definitions are utilised by the 'Personalisation Form Generator' to generate the series of forms that the healthcare professional will complete to in order to personalise the mHealth functions for the healthcare consumer. Therefore demonstrating the second element of personalisation, the tailoring of mHealth functions.

The final competency question (CQ 4) asks if the PMAD Ontology is capable of providing a suitable API based upon the healthcare consumers device. In PMADs this process remains transparent to the healthcare professional. Again, the class definition for the mobile device is extracted, parsed and interpreted to identify the mobile devices minimum API requirement.

The class identified, along with the completed forms are then utilised by the 'App Factory' component within the business layer of the framework to compile the personalised mHealth application.

7.3.4 Competency Synopsis

To summarise the objective of this phase of the evaluation was to determine if both the ontology and the framework are competent in addressing the tasks/problems they are designed to address. Evaluation of the competence both the ontology and framework utilised the 5 use case scenarios and the Personalised Mobile Application Development simulation (PMADs). Although the entire framework is not fully operational in PMADs, the evaluation concluded that the knowledge contained within the ontology is capable of providing answers to the competency questions defined in Section 5.3.1 and that the framework is capable of utilising the knowledge to in theory create mHealth applications that contain mHealth functions specifically chosen and tailored for an individual healthcare consumer. Overall, the PMAD Ontology and PMAD Frameworkare considered competent in the tasks they are designed to address within the scope of this research.

7.4 Evaluation Summary

In summary, this chapter presents the evaluation of both the PMAD Ontology and PMAD Framework. The ontology was evaluated from three perspectives, model, consistency and competence. For the reasons discussed in Sections 7.1.4, 7.2.4 and 7.3.4, the:

- **PMAD Ontology**
  - Is considered to be: free from all pitfalls (manually and automatic) that can be identified by the OntOlogy Pitfall Scanner, expect those highlighted as consequence of modelling decisions.
  - The knowledge encoded within the ontology is consistent.
  - When utilised in PMADs is capable of addressing the competency questions, defined in Section 5.3.1.
- **PMAD Framework**
  - Although conceptual, PMADs demonstrated that the framework has the necessary capabilities to achieve the personalisation of mHealth applications as intended by this research.

Overall, the evaluation of both the PMAD Ontology and PMAD Framework have provided sufficient evidence that satisfies the requirements of objectives (c) and (d) of this research.

Chapter 8

Conclusion

The final chapter in this thesis seeks to discuss this works contributions, achievements, limitations, recommendations for future work and is followed by a brief critical evaluation of the research undertaken.

8.1 Contributions and Achievements

As stated in Section 1.2, the major contribution of this research set out to develop an extensible ontology-driven framework that enables healthcare professionals to create personalised mHealth applications for healthcare consumers, without the need for intervention from mobile application developers.

This was achieved by exploring the challenges and issues surrounding the development of personalised mHealth applications, existing approaches and related works. The outcome being the literature review presented in Chapter 2. The literature review identified a series of signposts of the work required to achieve the aim and of this research. These signposts were formed on the basis of ideas, challenges and issues that are present and considered

significant within the literature. Each, influenced the design and development of the three main contributions of this work. Fulfilment of the aim was achieved via the combination of contributions described below:

1. Completion of the second objective produced the mHealth Application Function Taxonomy. As discussed in Section 4.4, the taxonomy serves as a classification tool capable of categorising health-related functions in mobile health care applications designed for healthcare consumers. The taxonomy was developed as part of this research was the product of a systematic exploration and analysis of health-related functions and created to gain an understanding of the type of health functions available and the distinct attributes that make them unique. As a result of observations made during the development led to the design of several use case scenarios and the generic function model that is used throughout the design and development of the PMAD Ontology and PMAD Framework. Outside of the boundaries of this research, the taxonomy also can be utilised from a practical standpoint during the development of a mHealth application. The taxonomy along with the classification tool can also utilise by mobile application developers to identify the types of healthcare-related functions required for a specific application during the design phase since the taxonomy is based on the principles of a functional hierarchy diagram.

2. As shown in Chapter 7, the PMAD Ontology is a critical aspect of this work and is the driving force within the PMAD Framework, that enables healthcare professionals to create personalised mHealth applications for healthcare consumers without the domain expertise of mobile application developers. As demonstrated in the evaluation, knowledge encapsulated within the ontology has been successfully operationalised, enabling key development decisions to be answered automatically such as: determining the feasibility of a personalised mHealth application, selection of a suitable API and identification of the personalised components of a mHealth

function. Thus, removing the complexities of personalised mobile application development from the healthcare professional. Following the procedures described throughout Chapter 5 the ontology can be extended to include, new mobile device, functions, hardware, mobile platforms and thus satisfy the requirements of objective (c).

3. The final contribution made by this work is the PMAD Framework. The framework presented in this thesis defines a multilayer high-level architecture that follows Microsoft's architectural design philosophy. It describes the fundamental components, services and data sources that collectively provide the functionality that enables healthcare professionals to create a personalised mHealth application for healthcare consumers. The framework is designed to tackle key issues and challenges that were discussed throughout this thesis. Although conceptual, the core components of the framework were implemented simulating key functionality.

8.2 Limitations & Recommendations for Future Work

Throughout the development of this research, there have been several occasions were interesting questions, thoughts and ideas came to mind that lay outside of the scope and focus of this particular research. Moreover, the limitations of this research also provide opportunities for future work. Highlighted below are some of these areas:

- **mHealth Application Function Taxonomy**
  - ***Evolution:*** The Taxonomy has been designed with evolution in mind, ensuring it is still applicable and represents current themes within the domain, its vital that the new information is introduced periodically to advance, refine and introduce new classes in the taxonomy.

- **PMAD Ontology**
  - ***Inclusion of more elaborate knowledge to the ontology:*** The PMAD Ontology in its current form contains sufficient knowledge to answer the competency questions, defined in Section 5.3.1. Expanding the knowledge contained within the ontology could further improve the reasoning capabilities of the ontology and add new capabilities to the framework.
  - ***Inclusion of medical devices and sensors:*** With the increasing availability of medical sensors and rise in popularity of wearable computing devices knowledge surrounding these devices could be added in future revisions of the ontology to broaden the scope and possibilities of potential functions and applications.
- **PMAD Framework**
  - ***Machine Learning:*** Machine Learning was briefly discussed in Chapter 2 and has become a trending topic of research within recent literature. Although out of the scope of the work described in this thesis, the questions of 'can/how machine learning be utilised to further assist healthcare professionals throughout the personalised mobile healthcare application development process?'
  - ***Work towards implementation:*** One of the limitations of this research is that the many of the components described within the framework are conceptual. Although implementations of core components that related to the interfacing and utilisation of the knowledge encapsulated within the ontology were demonstrated in the development of PMADs, there is still further work required to take the concept to an operational service.

- **Apply approach to personalisation of mobile applications in other domains**

  Reflecting upon the outcome of this work, the approach to personalisation of mobile applications described in this thesis, in theory, could be applied to other domains such as education. Following the same process of, developing a taxonomy of functions relating to the domain, addition of the functions relating to the domain to the ontology and finally modification of components and services within the framework to support the creation of mobile applications relating to the new domain.

8.3 Critical Evaluation

With regards to the aim of this research, the three main contributions that were produced were sufficient at addressing the challenges and issues they intended to solve. Each of the contributions were a response to challenges extracted from literature and personal experiences of working on projects relating to software and mobile application development within the healthcare industry. Each contribution was reinforced and governed by the signposts defined in Table 2.5. The signposts provided a suitable mechanism to help direct and prevent divergence, ensuring decisions made during the development of each contribution remained in line with the overall aim and scope of this research.

What was interesting however, was reflecting on how this research was originally envisaged and how its direction and overall outcome was influenced by the works reviewed and analysed. The entire research was a continuous learning curve from start to finish not only from a theoretical and practical standpoint but also from an interpersonal standpoint too. Towards the beginning expectations, tangents and fuzzy scope made it difficult to really focus on fulfilling the aim. But as new knowledge was acquired, and various skills developed, the picture became clearer. This experience at the beginning of the research process was a vital learning experience. New knowledge, discussions and experiences no

doubt influenced the final products of this research, but some provided a different outlook on aspects of this work. As discussed in Section 8.2, some highlighted areas of work that could either have; been improved if time was not a factor or identified potential areas of future work to progress this work further. What is key to take away from this is that even if the limitations discussed in the previous section were addressed, the same conclusions would have been reached. However, they would have improved the quality or applicability of the contributions made in this work.

References

[1] Mckinsey & Company; GSMA, "mHealth: A new vision for healthcare," *Mckinsey & Company*, pp. 1–20, 2010.

[2] K. Arning and M. Ziefle, "Different perspectives on technology acceptance: The role of technology type and age," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 5889 LNCS, pp. 20–41, 2009.

[3] C. Free, G. Phillips, L. Felix, L. Galli, V. Patel, and P. Edwards, "The effectiveness of M-health technologies for improving health and health services: a systematic review protocol," *BMC Research Notes*, vol. 3, no. 1, p. 250, 2010.

[4] J. Cruickshank, G. Beer, and E. Winpenny, "Healthcare without Walls: A Framework for Delivering Telehealth at Scale," 2020Health, Tech. Rep. November, 2010.

[5] R. S. Istepanian, S. Laxminarayan, and C. Pattichis, *M-Health: Emerging Mobile Health Systems*, 1st ed., R. S. Istepanian, S. Laxminarayan, and C. Pattichis, Eds. Springer, 2010. [Online]. Available: http://dx.doi.org/10.1007/b137697

[6] R. Snyderman and C. D. Drake, "Personalized Health Care: Unlocking the potential of genmoic and precision medicine," *The Journal of Precision Medicine*, 2016.

[7] EGAN, "Personalised Healthcare Frequently Asked Questions." [Online]. Available: http://www.geneticalliance.org.uk/docs/egan{_}personalisedhealthcare.pdf

[8] Society for Public Health Education, *Health Promotion Programs: from theory to practice*, D. Fertman, Carl; Allensworth, Ed. Toronto: John Wiley and Sons, 2010.

[9] S. C. Wangberg, "Personalized Technology for Supporting Health Behaviors," *2013 Ieee 4Th International Conference on Cognitive Infocommunications (Coginfocom)*, pp. 339–344, 2013.

[10] M. E. Cupples, A. McKnight, C. O'Neill, and C. Normand, "The effect of personal health education on the quality of life of patients with angina in general practice," *Health Education Journal*, vol. 55, no. 1, pp. 75–83, mar 1996. [Online]. Available: http://hej.sagepub.com/content/55/1/75.abstract

[11] Y. Y. Shieh, F. Y. Tsai, A. Anavim, M. D. Wang, and C.-M. C. Lin, "Mobile Healthcare: Opportunities and Challenges," *International Conference on the Management of Mobile Business (ICMB 2007)*, vol. i, no. Icmb, pp. 50–50, 2007. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4278593

[12] R.-l. P. H. Care, G. S. Ginsburg, J. Staples, and A. P. Abernethy, "Academic Medical Centers : Ripe for Raid-Learning Personalized Health Care," *Science Translational Medicine*, vol. 3, no. 101, pp. 1–4, 2011.

[13] M. Lee, B. Kang, D. Han, S. Jung, and C. Cho, "A platform for personalized mobile u-health application design and development," *2008 10th IEEE Intl. Conf. on e-Health Networking, Applications and Service, HEALTHCOM 2008*, pp. 221–226, 2008.

[14] D. Martín, D. López-de Ipiña, A. Alzua-Sorzabal, C. L. Lamsfus, and E. Torres-Manzanera, "A methodology and a web platform for the collaborative development of context-aware systems," *Sensors (Switzerland)*, vol. 13, no. 5, pp. 6032–6053, 2013.

[15] M. Alloghani, A. Hussain, D. Ai-jumeily, P. Fergus, O. Abuelma, and H. Hamden, "A Mobile Health Monitoring Application for Obesity Management and Control Using the," *2016 Sixth International Conference on Digital Information Processing and Communications (ICDIPC)*, pp. 19–24, 2016.

[16] S. Akter and P. Ray, "mHealth - an Ultimate Platform to Serve the Unserved." *Yearbook of medical informatics*, no. November, pp. 94–100, 2010.

[17] AppyPie, "About Us | Appy Pie," 2015. [Online]. Available: http://www.appypie.com/about-us

[18] MIT, "MIT App Inventor | Explore MIT App Inventor," 2015. [Online]. Available: http://appinventor.mit.edu/explore/

[19] Microsoft, "Windows App Studio–Free Tool to create apps in Windows Stores|Microsoft," 2015. [Online]. Available: http://appstudio.windows.com/en-us

[20] S. Myneni, M. Amith, Y. Geng, and C. Tao, "Towards an Ontology-driven Framework to Enable Development of Personalized mHealth Solutions for Cancer Survivors ' Engagement in Healthy Living," 2015.

[21] K. L. Skillen, L. Chen, C. D. Nugent, M. P. Donnelly, and I. Solheim, "A user profile ontology based approach for assisting people with dementia in mobile environments." *Conference proceedings : ... Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Conference*, vol. 2012, pp. 6390–3, 2012. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/23367391

[22] K. L. Skillen, L. Chen, C. D. Nugent, M. P. Donnelly, W. Burns, and I. Solheim, "Ontological user modelling and semantic rule-based reasoning for personalisation of Help-On-Demand services in pervasive environments," *Future Generation Computer Systems*, vol. 34, pp. 97–109, 2014. [Online]. Available: http://dx.doi.org/10.1016/j.future.2013.10.027

[23] Vital Wave Consulting, "mHealth for Development: The Opportunity of Mobile Technology for Healthcare in the Developing World," United Nations Foundation and Vodafone Foundation Technology Partnership, Tech. Rep. 1, 2009. [Online]. Available: http://www.globalproblems-globalsolutions-files.org/

[24] S. Laxminarayan and R. S. Istepanian, "UNWIRED E-MED: the next generation of wireless and internet telemedicine systems." *IEEE transactions on information technology in biomedicine : a publication of the IEEE Engineering in Medicine and Biology Society*, vol. 4, no. 3, pp. 189–193, 2000.

[25] R. S. H. Istepanaian and Y.-T. Zhang, "Guest editorial. Introduction to the special section: 4G Health–the long-term evolution of m-Health." *IEEE transactions on information technology in biomedicine : a publication of the IEEE Engineering in Medicine and Biology Society*, vol. 16, no. 1, pp. 1–5, 2012.

[26] M. N. Kamel Boulos, S. Wheeler, C. Tavares, and R. Jones, "How smartphones are changing the face of mobile and participatory healthcare: an overview, with example from eCAALYX." *Biomedical engineering online*, vol. 10, no. 1, p. 24, apr 2011.

[27] M. Ebling and J. Kannry, "Healthcare," *Pervasive Computing*, pp. 14–17, 2012.

[28] P. Yu, M. X. Wu, H. Yu, and G. Q. Xiao, "The challenges for the adoption of m-health," *2006 IEEE International Conference on Service Operations and Logistics, and Informatics, SOLI 2006*, pp. 181–186, jun 2006. [Online]. Available: http://ieeexplore.ieee.org/articleDetails.jsp?arnumber=4125574

[29] Oxford Dictionaries, "Personalize - definition of personalize in English from the Oxford Dictionaries," 2014. [Online]. Available: https://en.oxforddictionaries.com/definition/personalize

[30] D. Grigg-saito, S. Och, S. Liang, R. Toof, and L. Silka, "Health Promotion Practice," in *Images of Heatlh*, 2nd ed., O. et Al., Ed. Toronto: Canadian Scholars Pres, 2008, pp. 3–30.

[31] A. S. M. Mosa, I. Yoo, and L. Sheets, "A Systematic Review of Healthcare Applications for Smartphones," *BMC Medical Informatics and Decision Making*, vol. 12, no. 1, p. 67, 2012. [Online]. Available: BMCMedicalInformaticsandDecisionMaking

[32] P. Nicole, H. Castañeda, M. Nichter, S. Wind, L. Carruth, and M. Muramoto, "Lay Health Influencers: How They Tailor Brief Tobacco Cessation Interventions," *Health Education and Behavior*, vol. 29, no. 6, pp. 997–1003, 2012.

[33] R. Snyderman, "Personalized health care: From theory to practice," *Biotechnology Journal*, vol. 7, no. 8, pp. 973–979, 2012.

[34] A. Coulter, S. Roberts, and A. Dixon, "Delivering better services for people with long-term conditions- The King's Fund," no. October, pp. 1–28, 2013.

[35] S. Y. Ho and S. B. Bull, "User's adoption of mobile services: Preference and location personalization," *Proceeding - 5th International Conference on Computer Sciences and Convergence Information Technology, ICCIT 2010*, pp. 314–319, 2010.

[36] J. Blom, "Personalization - A Taxonomy," *Chi 2000*, no. April, pp. 1–2, 2000.

[37] K. Henricksen and J. Indulska, "Personalising context-aware applications," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 3762 LNCS, pp. 122–131, 2005.

[38] I. Jorstad and T. Van Do, "Service personalisation in mobile heterogeneous environments," *Proceedings of the Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications and Services, AICT/ICIW'06*, vol. 2006, p. 70, 2006.

[39] Android Open Source Project, "Android Overview | Open Handset Alliance," 2013. [Online]. Available: http://www.openhandsetalliance.com/android{_}overview.html

[40] Deloitte, "The UK Cut: Game of Phones," *Mobile Consumer 2015*, pp. 1–69, 2015. [Online]. Available: https://www.deloitte.co.uk/mobileuk/assets/pdf/Deloitte-Mobile-Consumer-2015.pdf

[41] Ofcom, "The Communications Market Report," Ofcom, London, Tech. Rep., 2015. [Online]. Available: http://stakeholders.ofcom.org.uk/binaries/research/cmr/cmr15/CMR{_}UK{_}2015.pdf

[42] I. Sommerville, *Software Engineering*, 5th ed., 1995.

[43] IEEE, "INTERNATIONAL STANDARD ISO / IEC / IEEE Systems and software engineering — engineering," pp. 1–94, 2011. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp={&}arnumber=6146379{&}isnumber=6146378

[44] I. Sommerville, *Software Engineering*, 9th ed. Addison-Wesley, 2011.

[45] D. Campbell, E. G. Pereira, and G. McDowell, "Ontology driven framework for personal mhealth application development," in *Proceedings - 2014 8th International Conference on Next Generation Mobile Applications, Services and Technologies, NGMAST 2014*, 2014, pp. 320–325.

[46] M. Olff, "Mobile mental health: A challenging research agenda," *European Journal of Psychotraumatology*, vol. 6, pp. 1–8, 2015.

[47] H. K. Flora, X. Wang, and S. V. Chande, "An Investigation on the Characteristics of Mobile Applications: A Survey Study," *I.J. Information Technology and Computer Science Information Technology and Computer Science*, vol. 11, no. 11, pp. 21–27, 2014.

[48] G. Forman and J. Zahorjan, "The challenges of mobile computing," *Computer*, vol. 27, no. 4, pp. 38–47, apr 1994. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=274999

[49] A. K. Gupta, "Challenges of Mobile Computing," in *2nd National Conference on Challenges & Opportunities in Information Technology (COIT-2008) RIMT-IET*, Roorkee, 2008, pp. 86–90.

[50] F. Liu, P. Shu, H. Jin, L. Ding, J. Yu, D. Niu, and B. Li, "Gearing resource-poor mobile devices with powerful clouds: Architectures, challenges, and applications," *IEEE Wireless Communications*, vol. 20, no. 3, pp. 14–21, 2013.

[51] M. E. Joorabchi, A. Mesbah, and P. Kruchten, "Real Challenges in Mobile App Development," *2013 ACM / IEEE International Symposium on Empirical Software Engineering and Measurement*, pp. 15–24, 2013.

[52] K. R. S. Patch, J. W. Spellman, and K. I. A. Wahlbin, "Mobile Accessibility: How WCAG 2.0 and Other W3C / WAI Guidelines Apply to Mobile," *Working Draft*, no. February, pp. 1–14, 2015. [Online]. Available: https://www.w3.org/TR/mobile-accessibility-mapping/

[53] Android, "Android 5.1 APIs | Android Developers," 2015. [Online]. Available: http://developer.android.com/about/versions/android-5.1.html

[54] Kantar, "Market share held by the leading smartphone operating systems in Great Britain from January 2013 to March 2016," 2016. [Online]. Available: http://www.statista.com/statistics/274119/market-share-held-by-smartphone-os-in-great-britain/

[55] Salesforce, "Native, HTML5, or Hybrid: Understanding Your Mobile Application Development Options - developer.force.com," 2016. [Online]. Available: https://developer.salesforce.com/page/Native,{_}HTML5,{_}or{_}Hybrid:{_}Understanding{_}Your{_}Mobile{_}Application{_}Development{_}Options

[56] P. Smutný, "Mobile development tools and cross-platform solutions," *Proceedings of the 2012 13th International Carpathian Control Conference, ICCC 2012*, pp. 653–656, 2012.

[57] R. Solutions, "How to Choose the Right Architecture For Your Mobile Application Rapid-Value Mobile Applications can Sell Products & Services , Raise Productivity , and Increase Awareness of Your Brand," Tech. Rep., 2012.

[58] B. Raluca, "Mobile Web Apps vs . Mobile Native Apps : How to Make the Right Choice," p. 13, 2013.

[59] P. Gokhale and S. Singh, "Multi-platform strategies, approaches and challenges for developing mobile applications," *2014 International Conference on Circuits, Systems, Communication and Information Technology Applications (CSCITA)*, pp. 289–293, 2014. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6839274

[60] F. Vergari, S. Bartolini, F. Spadini, A. . D'Elia, G. Zamagni, L. Roffia, and T. Cinotti, "A Smart Space application to dynamically relate medical and environmental information," *Design, Automation & Test in Europe Conference & Exhibition (DATE),2010*, vol. 86, no. 10, pp. 1542–1547, 2010. [Online]. Available: http://ieeexplore.ieee.org/search/srchabstract.jsp?tp={&}arnumber=5457056{&}openedRefinements=*{&}filter=AND(NOT(4283010803)){&}searchField=Search+All{&}queryText=A+Smart+Space+Application+to+Dynamically+Relate+Medical+and+Environmental+Information

[61] Infosys, "M-Health: Challenges, benefits, and keys to successful implementation," Tech. Rep. 0, 2008.

[62] M. Plachkinova and S. Andrés, "A Taxonomy of mHealth Apps – Security and Privacy Concerns," *2015 48th Hawaii International Conference on System Sciences*, vol. 48, no. June 2013, pp. 3187–3196, 2015.

[63] Accenture, "Mobile Application Development : Challenges and Best Practices An increasing number of both mobile devices and potential applications are forcing," Tech. Rep., 2012.

[64] M. A. Ben Yahmed, M. A. Bounenni, Z. Chelly, and A. Jlassi, "A new mobile health application for an ubiquitous information system," *Proceedings of 2013 6th Joint IFIP Wireless and Mobile Networking Conference, WMNC 2013*, 2013.

[65] A. Dzhagaryan, A. Milenković, and M. Burtscher, "Energy Efficiency of Lossless Data Compression on a Mobile Device: An Experimental Evaluation," Tech. Rep., 2013.

[66] G. Deepak and B. Pradeep, "Challenging Issues and Limitations of Mobile Computing," *Int. J. Computer Technology & ...*, vol. 3, no. 1, pp. 177–181, 2012. [Online]. Available: http://core.kmi.open.ac.uk/download/pdf/1133485.pdf

[67] I. Jorstad, D. V. Thanh, and S. Dustdar, "Personalisation of Next Generation Mobile Services," pp. 927–941, 2006. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.141.1246

[68] I. Jørstad, D. V. Thanh, and S. Dustdar, "Towards Service Continuity for Generic Mobile Services," *IFIP International Conference on Intelligence in Communication Systems (INTELLCOMM 04)*, 2004.

[69] B. Schmidt-Belz, A. Nick, S. Poslad, and A. Zipf, "Personalized and location-based mobile tourism services," *Workshop on "Mobile Tourism Support Systems" in conjunction with Mobile HCI*, p. 14, 2002. [Online]. Available: http://195.130.87.21:8080/dspace/handle/123456789/622

[70] C. Fredrich, H. Kuijs, and C. Reich, "An Ontology for User Profile Modeling in the Field of Ambient Assisted Living," *SERVICE COMPUTATION 2014, The Sixth International Conferences on Advanced Service Computing*, vol. 5, pp. 24–31, 2014.

[71] B. Holtkamp, R. Gartmann, and Y. Han, "FLAME2008 - Personalized Web services for the olympic games in 2008 in Beijing," in *Proceedings of eChallenges 2003*, Bologna, 2003.

[72] N. Weibenberg and F. Isst, "Using ontologies in personalized mobile applications," *Proceedings of the 12th annual ACM international workshop on Geographic information systems GIS 04*, no. 01, pp. 2–11, 2004. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1032222.1032225

[73] D. Brickley and L. Miller, "The FOAF Project," 2015. [Online]. Available: http://www.foaf-project.org/

[74] H. information and management systems society, "HIMSS Patient engagement framework," p. 4, 2014. [Online]. Available: http://www.himss.org/himss-patient-engagement-frameworkhttp://himss.files.cms-plus.com/HIMSSorg/NEHCLibrary/HIMSS{_}Foundation{_}Patient{_}Engagement{_}Framework.pdf

[75] D. C. Mohr, S. M. Schueller, E. Montague, M. N. Burns, and P. Rashidi, "The behavioral intervention technology model: An integrated conceptual and technological framework for ehealth and mhealth interventions," *Journal of Medical Internet Research*, vol. 16, no. 6, 2014.

[76] M. Sutterer, O. Droegehorn, and K. David, "UPOS: User profile ontology with situation-dependent preferences support," *Proceedings of the 1st International Conference on Advances in Computer-Human Interaction, ACHI 2008*, pp. 230–235, 2008.

[77] D. Heckmann, T. Schwartz, B. Brandherm, M. Schmitz, and M. von Wilamowitz-Moellendorff, "Gumo - The General User Model Ontology," *Proceedings of {UM} 2005: 10th {I}nternational {C}onference on {U}ser modeling*, vol. 3538, pp. 428–432, 2005.

[78] E. Toch, Y. Wang, and L. F. Cranor, "Personalization and privacy: A survey of privacy risks and remedies in personalization-based systems," *User Modeling and User-Adapted Interaction*, vol. 22, no. 1-2, pp. 203–220, 2012.

[79] J. Kay, B. Kummerfeld, and P. Lauder, "Personis: a server for user models," *Adaptive Hypermedia and Adaptive Web-Based Systems*, vol. 2347, pp. 203–212, 2002. [Online]. Available: http://www.springerlink.com/index/2L54YRGC0P8N2D5G.pdf

[80] M. Palmieri, I. Singh, and A. Cicchetti, "Comparison of cross-platform mobile development tools," *2012 16th International Conference on Intelligence in Next Generation Networks, ICIN 2012*, pp. 179–186, 2012.

[81] Gartner, "Gartner Says Demand for Enterprise Mobile Apps Will Outstrip Available Development Capacity Five to One," 2015. [Online]. Available: http://www.gartner.com/newsroom/id/3076817

[82] C. LeRouge and J. Ma, "User Profiles and Personas in Consumer Health Technologies," *2010 43rd Hawaii International Conference on System Sciences*, no. June, pp. 1–10, 2010. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs{_}all.jsp?arnumber=5428383{%}5Cnhttp://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5428383

[83] R. Nickerson, J. Muntermann, U. Varshney, H. Isaac, and H. I. Taxonomy Devel-, "Taxonomy Development In Information Systems: Developing A Taxonomy Of Mobile Applications," *HAL ID: halshs-00375103*, vol. HAL ID:, 2009. [Online]. Available: https://halshs.archives-ouvertes.fr/halshs-00375103

[84] P. Olla and C. Shimskey, "mHealth taxonomy: a literature survey of mobile health applications," *Health and Technology*, vol. 4, no. 4, pp. 299–308, 2015.

[85] K. D. Bailey, *Typologies and taxonomies: an introduction to classification techniques*, 1994, no. 07.

[86] Y. Kwang Hooi, M. F. Hassan, A. I. B. Z. Abidin, N. I. B. Arshad, and A. M. Shariff, "A Generic Ontology Methodology and Factors to Address Design Consistency," *IT Convergence and Security (ICITCS), 2015 5th International Conference on*, pp. 1–5, 2015. [Online]. Available: http://ieeexplore.ieee.org/ielx7/7287494/7292885/07292919.pdf?tp={&}arnumber=7292919{&}isnumber=7292885

[87] R. Hoekstra, "Ontology Representation, Design patterns and ontologies that make sense," *Frontiers in Artificial Intelligence and Applications*, vol. 197, no. 1, pp. 1–236, 2009.

[88] M. Uschold, M. Gruninger, M. Uschold, and M. Gruninger, "Ontologies : Principles , Methods and Applications," *Knowledge Engineering Review*, vol. 11, no. 2, pp. 93–136, 1996.

[89] M. Uschold, "Building Ontologies : Towards a Uni ed Methodology," no. September, 1996.

[90] F. M. Gruninger M., "Methodology for the design and evaluation of ontologies, in: Workshop on Basic Ontological Issues in Knowledge Sharing, Montreal." 1995.

[91] M. F. López, A. Gómez-Pérez, J. P. Sierra, and A. P. Sierra, "Building a chemical ontology using methontology and the ontology design environment," *IEEE Intelligent Systems and Their Applications*, vol. 14, no. 1, pp. 37–46, 1999.

[92] M. Fernández-López and A. Gómez-Pérez, "Overview and analysis of methodologies for building ontologies," *The Knowledge Engineering Review*, vol. 17, no. 2, pp. 129–156, 2002.

[93] N. Dahlem, J. G. J. Guo, a. Hahn, and M. Reinel, "Towards an User-Friendly Ontology Design Methodology," *2009 International Conference on Interoperability for Enterprise Software and Applications China*, pp. 180–186, 2009.

[94] A. F. Sawsaa, *Ontological Engineering approach of developing ontology of information science*, 1st ed. Huddersfield: Anchor Academic Publishing, 2015.

[95] D. Vrandecic, "Ontology Evaluation," Ph.D. dissertation, Karlsruher Instituts fur Technologie, 2010.

[96] R. Shearer, B. Motik, and I. Horrocks, "HermiT: a highly-efficient OWL reasoner," p. 10, 2008.

[97] ISO, IEC, and IEEE, "Systems and software engineering - System life cycle processes," ISO; IEC; IEEE, Tech. Rep., 2015.

[98] U. Flick, *An Introduction to Qualitatative*, 4th ed. London: Sage, 2009.

[99] D. Platter, "Welcome UML web site," 2008. [Online]. Available: http://www.uml.org/

[100] Microsoft, "UML Activity Diagrams: Reference," 2013. [Online]. Available: https://msdn.microsoft.com/en-us/library/dd409360.aspx

[101] J. Garland and R. Anthony, "UML Quick Tour," in *Large-scale Software Architecture A Practical Guide Using UML*, 2003, ch. 5 UML Quic, pp. 69–86.

[102] Developers Android, "Android, the world's most popular mobile platform," pp. 3–5, 2014. [Online]. Available: https://developer.android.com/about/android.html

[103] T. Gruber, "Toward principles for the design of ontologies used for knowledge sharing," *International Journal of Human-Computer Studies*, vol. 43, no. 5-6, pp. 907–928, 1993. [Online]. Available: http://dx.doi.org/10.1006/ijhc.1995.1081

[104] W. Hesse, "Ontologies in the Software Engineering process," *2nd GI-/GMDS-Workshop on Enterprise Application Integration (EAI'05)*, vol. 141, pp. 3–15, 2005.

[105] T. S. Dillon, E. Chang, and P. Wongthongthain, "Ontology-based software engineering-software engineering 2.0," *Aswec 2008: 19Th Australian Software Engineering Conference, Proceedings*, pp. 13–23, 2008.

[106] Y. He, J. Zhang, L. Q. Yue, Z. M. Li, and L. J. Tang, "Based on ontology methodology to model and evaluate System of Systems (SoS)," *Proceedings of the 9th International Conference on System of Systems Engineering: The Socio-Technical Perspective, SoSE 2014*, no. 3, pp. 101–106, 2014.

[107] G. K. Saha, "Web ontology language (OWL) and semantic web," *Ubiquity*, vol. 2007, no. September, pp. 1–1, 2007. [Online]. Available: https://www.w3.org/TR/owl-features/http://portal.acm.org/citation.cfm?doid=1295289.1295290

[108] T. Saito and J. Sadoshima, "The Protégé Project: A Look Back and a Look Forward," vol. 116, no. 8, pp. 1477–1490, 2016.

[109] I. Palmisano, "OWL API," Oxford, 2014. [Online]. Available: http://sourceforge.net/projects/owlapi/files/OWLAPI(forOWL2.0)/4.0.0/

[110] O. Corcho, M. Fernández-López, A. Gómez-Pérez, and A. López-Cima, "Building legal ontologies with METHONTOLOGY and WebODE," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 3369 LNAI, pp. 142–157, 2005.

[111] J. D. Novak and a. J. Cañas, "The Theory Underlying Concept Maps and How to Construct and Use Them," *IHMC CmapTools*, pp. 1–36, 2008. [Online]. Available: http://www.ode.state.or.us/teachlearn/subjects/science/resources/msef2010-theory{_}underlying{_}concept{_}maps.pdf{%}5Cnpapers://dee23da0-e34b-4588-b624-f878b46d7b3d/Paper/p348

[112] C. Britton and J. Doake, *Object-Oriented Systems Development: A gentle introduction*, D. Hatter, Ed. London: McGraw Hill Publishing, 2000.

[113] M. Prestley, *Practical Object-Oriented Design With UML*, 2nd ed., A. Waller, Ed. London: McGraw Hill Publishing, 2000.

[114] ISO, "Information technology — Software product quality," *Iso/Iec Fdis 9126-1*, vol. 2000, pp. 1–26, 2000. [Online]. Available: http://www.cse.unsw.edu.au/{~}cs3710/PMmaterials/Resources/9126-1Standard.pdf

[115] J. O.Grady, *System Requirements Analysis*, 2nd ed., ser. Elsevier insights. Elsevier Science, 2013. [Online]. Available: https://books.google.co.uk/books?id=o8kIR7lvrRQC

[116] M. Pincher, "A guide to developing taxonomies for effective data management," 2010. [Online]. Available: http://www.computerweekly.com/feature/A-guide-to-developing-taxonomies-for-effective-data-management

[117] PyData Development Team, "Python Data Analysis Library — Pandas," 2012. [Online]. Available: http://pandas.pydata.org/index.htmlhttp://pandas.pydata.org/

[118] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank Citation Ranking: Bringing Order to the Web," *World Wide Web Internet And Web Information Systems*, vol. 54, no. 1999-66, pp. 1–17, 1998.

[119] N. Effingham, *An Introduction to Ontology*. Birmingham: Polity Press, 2013.

[120] F. Ruiz and J. Hilera, *Ontologies for Software Engineering and Software Technology*, 2006. [Online]. Available: http://link.springer.com/chapter/10.1007/3-540-34518-3{_}2{%}5Cnhttp://link.springer.com/10.1007/3-540-34518-3

[121] A. Madche, H.-P. Schnurr, S. Staab, and R. Studer, "Representation Language-Neutral Modeling of Ontologies," *Modelle und Modellierungssprachen in Informatik und Wirtschaftsinformatik : Beitrge des Workshops*, pp. 129–142, 2000.

[122] A. Gómez-Pérez, M. Fernández-López, O. Corcho, and a. Gomez-Perez, *Ontological Engeenering*, 2004. [Online]. Available: http://delicias.dia.fi.upm.es/wiki/images/a/aa/1.{_}Intro-Sweb.pdf{%}5Cnhttp://books.google.com/books?id=UjS0N1W7GSEC

[123] T. R. Gruber, "A translation approach to portable ontology specifications," *Knowledge Acquisition*, vol. 5, no. 2, pp. 199–220, 1993. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S1042814383710083

[124] X. Ma, C. Wu, E. J. M. Carranza, E. M. Schetselaar, F. D. van der Meer, G. Liu, X. Wang, and X. Zhang, "Development of a controlled vocabulary for semantic interoperability of mineral exploration geodata for mining projects," *Computers and Geosciences*, vol. 36, no. 12, pp. 1512–1522, 2010. [Online]. Available: http://dx.doi.org/10.1016/j.cageo.2010.05.014

[125] D. L. McGuinness, O. Come, I. Dieter, J. Hendler, and H. Lieberman, "Ontologies come of age: The web ' s growing needs," *Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential*, pp. 1–13, 2005.

[126] M. Daconta, L. Obrst, and K. Smith, "Understanding Taxonomies," in *The Semantic Web*. Indianapolis: Wiley Technology Publishing, 2003, ch. 7, pp. 145–180.

[127] L. Obrst, H. Liu, R. Wray, and L. Wilson, "Ontologies for semantically interoperable electronic commerce," *IFIP Advances in Information and Communication Technology*, vol. 108, pp. 325–333, 2003.

[128] R. Poli, M. Healy, and A. Kameas, *Theory and Applications of Ontology: Computer Applications*, ser. Theory and applications of ontology : [2]. Springer Netherlands, 2010. [Online]. Available: https://books.google.co.uk/books?id=lQS6Abf9wzwC

[129] J. F. Sowa, "Ontology, metadata, and semiotics," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 1867, 2000, pp. 55–81. [Online]. Available: http://www.jfsowa.com/ontology/ontometa.htm

[130] C. K. Ogden and I. A. Richards, *The Meaning of Meaning: A Study of the Influence of Language Upon Thought and of the Science of Symbolism. Supplementary Essays by B. Malinowski and F.G. Crookshank*. Harcourt, 1923. [Online]. Available: https://books.google.co.uk/books?id=i3MIAQAAIAAJ

[131] N. Guarino and D. Oberle, "What Is an Ontology?" Tech. Rep., 2009. [Online]. Available: http://link.springer.com/10.1007/978-3-540-92673-3

[132] F. Probst, "Investigating the Applicability of DOLCE as Foundation for Service Descriptions," *W3C Workshop on Frameworks for Semantics in Web Services*, no. October, pp. 1–5, 2005. [Online]. Available: ftp://ftp-sop.inria.fr/acacia/W3CAtelierWS/papers/PositionPaperProbst.pdf

[133] M. Daconta, L. Obrst, and K. Smith, *The Semantic Web: a guide to the future of XML, Web services, and knowledge management*. Wiley, 2009. [Online]. Available: http://scholar.google.com/scholar?hl=en{&}btnG=Search{&}q=intitle:the+semantic+web+a+guide+to+the+future+of+xml,+web+services,+and+knowledge+management{#}0

[134] N. Guarino, "Formal Ontology and Information Systems," *Proceedings of the first international conference*, no. June, pp. 3–15, 1998. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.29.1776{&}rep=rep1{&}type=pdf

[135] J. Davies, R. Studer, and P. Warren, *Semantic Web Technologies trends and research in ontology based systems*. Chichester: Wiley, 2006.

[136] D. Kalibatiene, "Perspectives in Business Informatics Research," vol. 261, no. January 2011, 2011. [Online]. Available: http://link.springer.com/10.1007/978-3-319-45321-7

[137] V. Maniraj and D. Sivakumar, "Ontology Languages - A Review," *International Journal of Computer Theory and Engineering, 2010*, vol. 2, no. 6, pp. 1793–8201, 2010.

[138] OWL Working Group, "OWL - Semantic Web Standards," 2012. [Online]. Available: https://www.w3.org/OWL/https://www.w3.org/2001/sw/wiki/OWL

[139] C. D. Baader Franz and P. F. McGuinness, Deborah L, Nardi Daniele, Patel-Schneider, *The Description Logic Handbook*, 2nd ed., Cambridge, 2007.

[140] M. Horridge and S. Bechhofer, "The OWLAPI: A Java API for OWL ontologies," *Semantic Web*, vol. 2, no. 1, pp. 11–21, 2011.

[141] I. Horrocks, P. F. Patel-Schneider, and F. Van Harmelen, "From SHIQ and RDF to OWL: The making of a Web Ontology Language," *Web Semantics*, vol. 1, no. 1, pp. 7–26, 2003.

[142] F. v. H. Deborah L. McGuinness, "Owl web ontology language overview," *W3C recommendation 10.2004-03*, vol. 2004, no. February, pp. 1–12, 2004. [Online]. Available: https://www.w3.org/TR/owl-features/

[143] O. Dameron, D. L. Rubin, and M. a. Musen, "Challenges in converting frame-based ontology into OWL: the Foundational Model of Anatomy case-study." *AMIA ... Annual Symposium proceedings / AMIA Symposium. AMIA Symposium*, vol. 1, pp. 181–185, 2005.

[144] M. Poveda-Villalón, M. Carmen Suárez-Figueroa, M. Ángel García-Delgado, and A. Gómez-Pérez, "OOPS! (OntOlogy Pitfall Scanner!): supporting ontology evaluation on-line," *Undefined*, vol. 1, pp. 1–5, 2009.

[145] "OOPS! - OntOlogy Pitfall Scanner! - Pitfall Catalogue," 2015. [Online]. Available: http://oops.linkeddata.es/catalogue.jsp

[146] Oracle, "Oracle | Integrated Cloud Applications and Platform Services," 2014. [Online]. Available: https://www.oracle.com/index.htmlhttp://www.oracle.com/index.html

[147] Quality Open Software, "Simple Logging Facade for Java (SLF4J)." [Online]. Available: http://www.slf4j.org/

[148] I. Palmisano, "jFact DL Reasoner," 2015. [Online]. Available: http://jfact.sourceforge.net/

[149] S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein, "OWL Web Ontology Language Reference," 2009. [Online]. Available: https://www.w3.org/TR/owl-ref/

[150] B. Rodriguez-Castro and H. Glaser, "Whose "Fault" Is This? Untangling Domain Concepts in an Ontology of Resilient Computing ," *Fast Abstracts at the 7th European Dependable Computing Conference (EDCC-7)*, 2008. [Online]. Available: http://eprints.ecs.soton.ac.uk/15408/

[151] C. Welty and N. Guarino, "Supporting ontological analysis of taxonomic relationships," *Data and Knowledge Engineering*, vol. 39, no. 1, pp. 51–74, 2001.

[152] A. Rector, "Modularisation of domain ontologies implemented in description logics and related formalisms including OWL," *Proceedings of the international conference on Knowledge capture - K-CAP '03*, p. 121, 2003. [Online]. Available: http://dl.acm.org/citation.cfm?id=945645.945664

[153] ——, "Normalisation of ontology implementations: Towards modularity, re-use, and maintainability," *Multi-Agent Systems*, 2002.

[154] J. Malone and H. Parkinson, "Reference and Application Ontologies," \url{http://ontogenesis.knowledgeblog.org/295}, 2010. [Online]. Available: http://ontogenesis.knowledgeblog.org/295

[155] M. Gruninger and M. S. Fox, "The Role of Competency Questions in Enterprise Engineering," *IFIP WG5 - 7 Workshop on Benchmarking - Theory and Practice*, pp. 1–17, 1994. [Online]. Available: http://www.eil.utoronto.ca/enterprise-modelling/papers/benchIFIP94.pdf

[156] C. Bezerra, F. Freitas, and F. Santana, "Evaluating ontologies with Competency Questions," *Proceedings - 2013 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology - Workshops, WI-IATW 2013*, vol. 3, no. November, pp. 284–285, 2013.

[157] Android Developers, "<uses-feature>," 2015. [Online]. Available: https://developer.android.com/guide/topics/manifest/uses-feature-element.html

[158] Android Developer, "Android Debug Bridge." [Online]. Available: https://developer.android.com/studio/command-line/adb.html

[159] Android Developers, "Setting up CTS | Android Open Source Project." [Online]. Available: https://source.android.com/compatibility/cts/setup

[160] Android Developers;, "Codenames, Tags, and Build Numbers | Android Open Source Project," 2015. [Online]. Available: https://source.android.com/source/build-numbers

[161] Android Developers, "Manifest.permission_group | Android Developers." [Online]. Available: https://developer.android.com/guide/topics/manifest/uses-sdk-element.html

[162] ——, "Manifest.permission_group | Android Developers." [Online]. Available: https://developer.android.com/reference/android/Manifest.permission{_}group.html{#}PHONE

[163] W3C, "OWL 2 Web Ontology Language Primer," 2007. [Online]. Available: https://www.w3.org/2007/OWL/wiki/Primer

[164] M. Horridge, "A Practical Guide To Building OWL Ontologies Using Protégè 4 and CO-ODE Tools Edition 1.3," *Matrix*, pp. 0–107, 2011. [Online]. Available: http://owl.cs.manchester.ac.uk/tutorials/protegeowltutorial/resources/ProtegeOWLTutorialP4{_}v1{_}3.pdf

[165] M. Horridge and P. F. Patel-Schneider, "OWL 2 Web Ontology Language: Manchester Syntax," 2008.

[166] C. Paz-trillo, J. Riani, M. Ribeiro, L. N. D. Barros, and R. Wassermann, "Classifying Ontologies," pp. 1–12. [Online]. Available: http://www.cecm.usp.br/{~}marciomr/publicacoes/WontoGeral.pdf{%}5Cnhttps://www.researchgate.net/publication/221336546{_}Classifying{_}Ontologies

[167] M. Horridge, "Justification based explanation in ontologies," Ph.D. dissertation, University of Manchester, 2011.

[168] H. Fan, F. K. Hussain, M. Younas, and O. K. Hussain, "An integrated personalization framework for SaaS-based cloud services," *Future Generation Computer Systems*, vol. 53, pp. 157–173, 2015.

[169] D. Budgen, "Component-Based Design," in *Software Design*, 2nd ed. Tottenham: Perason, 2003, ch. Component, pp. 401–418.

[170] ISO/IEC/IEEE 29119-4, *Software and systems engineering — Software testing — Part 4: Test Techniques*, 2015.

[171] A. Eliëns, *Principles of Object Orientated Development*, 2nd ed. Addison-Wesley, 200.

[172] D. Budgen, *Software Design*, 2nd ed. Pearson, 2003.

[173] ISO;IEEE;IEC, "INTERNATIONAL STANDARD ISO / IEC Software Engineering — Software Life," ISO; IEEE; IEC, Tech. Rep., 2006.

[174] K. H. Bennett and V. T. Rajlich, "Software maintenance and evolution," *Proceedings of the conference on The future of Software engineering - ICSE '00*, pp. 73–87, 2000. [Online]. Available: http://portal.acm.org/citation.cfm?doid=336512.336534

[175] J. T. Nosek and P. Palvia, "Software Maintenance Management: Changes in the Last Decade," *Journal of Software Maintenance*, vol. 2, no. 3, pp. 157–174, sep 1990. [Online]. Available: http://dx.doi.org/10.1002/smr.4360020303

[176] B. P. Lientz and E. B. Swanson, "Problems in Application Software Maintenance," *Commun. ACM*, vol. 24, no. 11, pp. 763–769, nov 1981. [Online]. Available: http://doi.acm.org/10.1145/358790.358796

[177] J. Martin and C. L. McClure, *Software Maintenance: The Problems and Its Solutions*. Prentice Hall Professional Technical Reference, 1983.

[178] H. van Vliet, *Software Engineering: Principles and Practice*, 3rd ed. Wiley Publishing, 2008.

[179] Microsoft Corporation, *Microsoft Application Architecture Guide*, 2nd ed. Microsoft Corporation, 2009. [Online]. Available: http://www.microsoft.com/en-us/download/details.aspx?id=16236

[180] M. Poveda-villalón and M. C. Suárez-figueroa, "OOPS ! – OntOlogy Pitfalls Scanner !" *OOPS! – OntOlogy Pitfalls Scanner!. Monografía (Informe Técnico). Facultad de Informática (UPM), Madrid.*, 2012.

[181] B. Glimm, I. Horrocks, B. Motik, and G. Stoilos, "Optimising ontology classification," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6496 LNCS, no. PART 1, pp. 225–240, 2010.

[182] N. Matentzoglu, B. Parsia, and U. Sattler, "OWL Reasoning: Subsumption Test Hardness and Modularity," *Journal of Automated Reasoning*, pp. 1–35, 2017.

[183] M. Krötzsch, F. Simancik, and I. Horrocks, "A Description Logic Primer," no. June, pp. 1–17, 2012. [Online]. Available: http://arxiv.org/abs/1201.4089

Appendix A

Taxonomy appendices

A collection of appendices associated with the development of the taxonomy

A.1 Taxonomy analysis: Frequency analysis Python script

Python script to perform frequency analysis on article dataset.

```
import csv
from collections import Counter
import pandas as pd

files = ['papers','apps']

for csvfile in files:
    codesraw = []
    codefix=[]
    flist = []
    clist= []

    with open(csvfile+'.csv') as file:
        reader = csv.reader(file)
        next(reader)
        for row in reader:
            temp = str(row[1])
            temp = temp.strip()
            codesraw.append(temp)

        series = Counter(codesraw)
        series.keys()

        for key, freq in series.items():
            flist.append(int(freq))
            clist.append(str(key))

    data = {'code': clist, 'freq': flist}

    df = pd.DataFrame(data=data)
    df.sort_values(by='freq',ascending=1, inplace=True)
    df.to_csv('processed'+csvfile+'.csv')
    print(df.describe())
```

A.2 Taxonomy analysis: Concept maps

A series of concept maps representing set A - D respectively.

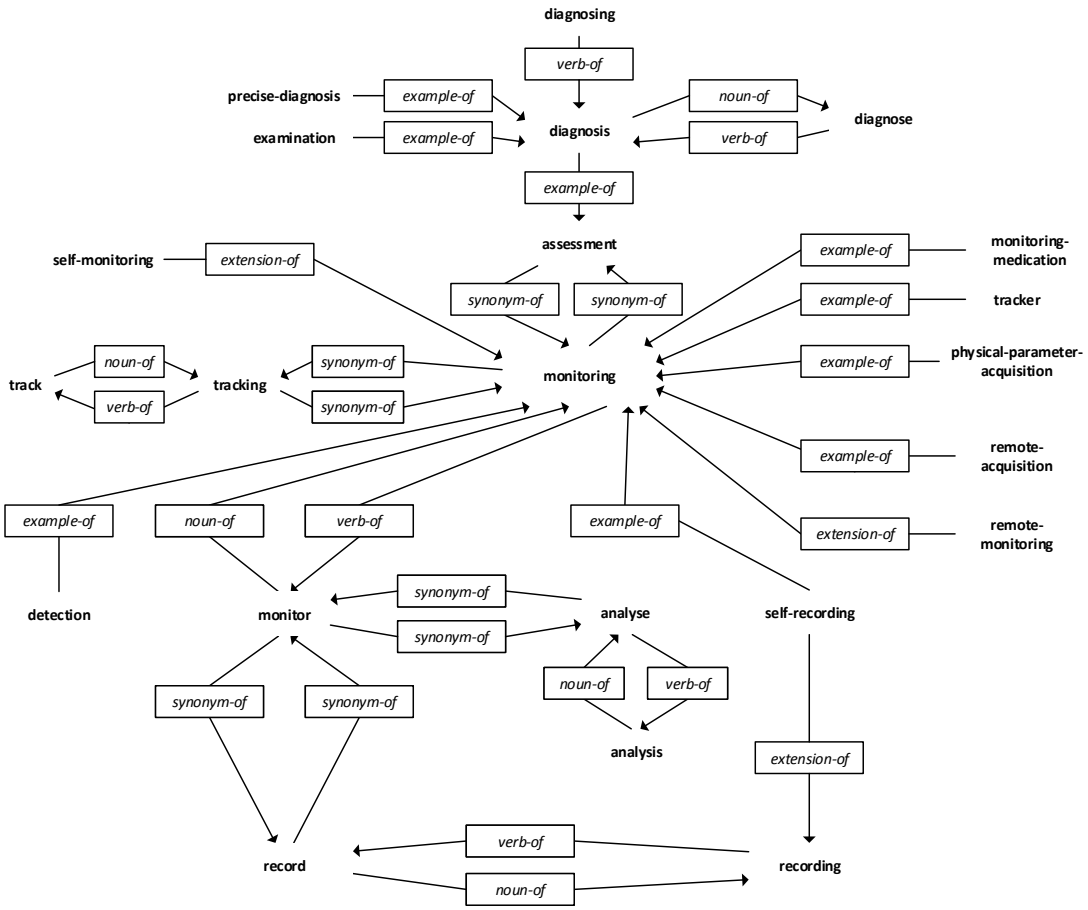


Figure A.1 Set A Concept Map

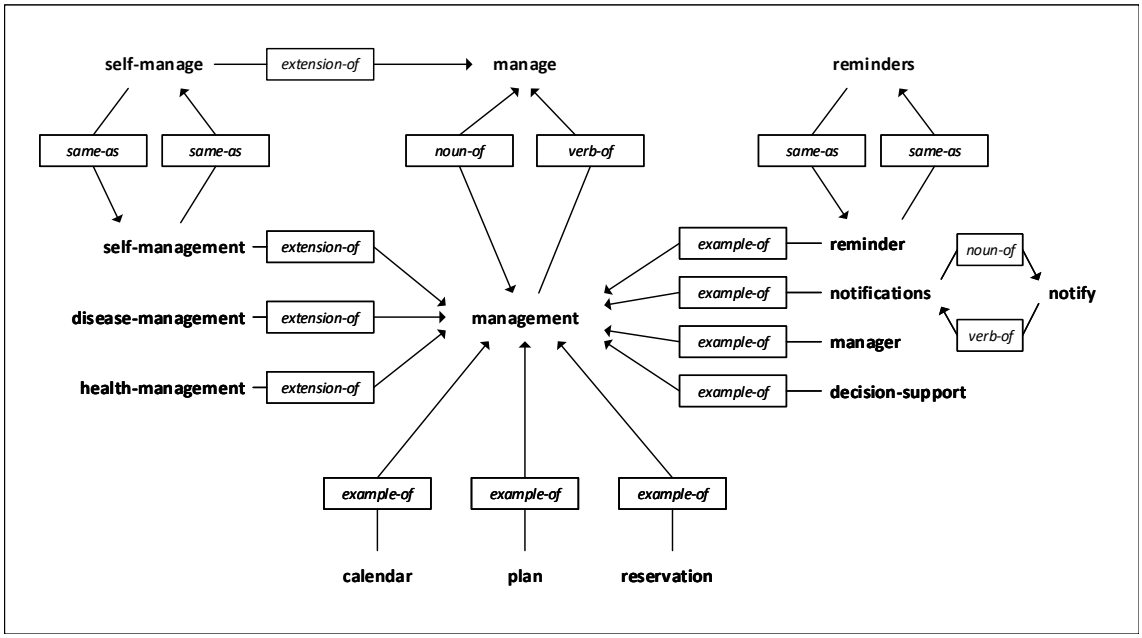


Figure A.2 Set B Concept Map

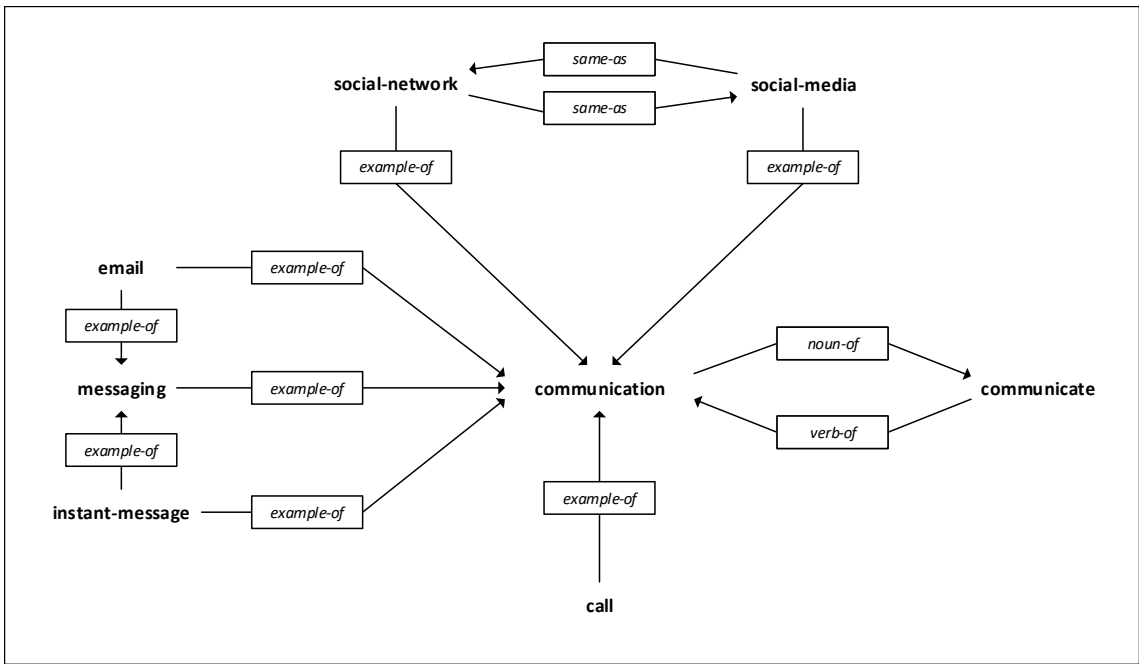


Figure A.3 Set C Concept Map

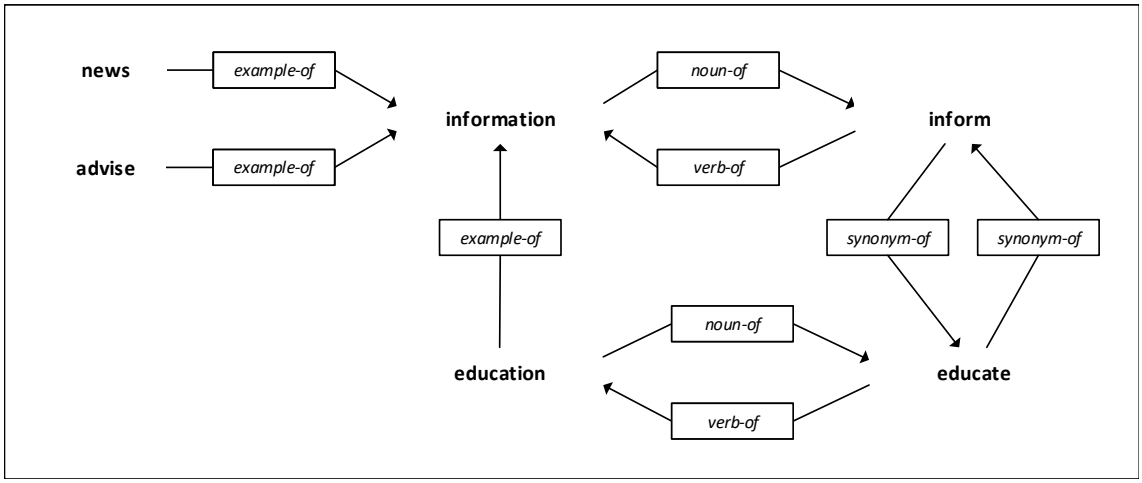


Figure A.4 Set D Concept Map

A.3 Taxonomy analysis: PageRank Python script

Python script developed to perform PageRank.

```
import numpy as np
import pandas as pd
import plotly.plotly as py
import plotly.graph_objs as go
import math

name = 'Set A'
csv = 'setA.csv'
dfcolumns = []
pagerankvalues = []
dfdata = []
counter = 0
countls = []
epsilon = 0.01
dampeningfactor = 0.8

def __euclideanNorm(series):
    return math.sqrt(series.dot(series))

def buildMatrixFromCSV(CSVfile):
    mdf = pd.read_csv(CSVfile, header=None)
    matrixdf = mdf.as_matrix()
    return matrixdf

# Matrix M
M = buildMatrixFromCSV(CSVfile=csv)

# Number of nodes in map
nodecount = len(M)

# intial node values
r0 = np.array([1 / nodecount] * nodecount)

# beta * M
M = dampeningfactor * M

# Auto populates a matrix [N] with the (1-beta)(1/nodecount)
N = [[(1 / nodecount) * (1 - dampeningfactor)
    for y in range(nodecount)] for x in range(nodecount)]

# A (Matrix R + Matrix N)
A = M + N

#Appends PR(v_1) to the PageRank List
pagerankvalues.append(r0)

while (True):
    countls.append(counter)
```

```
    tempPageRank = A.dot(pagerankvalues[counter])
    delta = tempPageRank - pagerankvalues[counter]
    pagerankvalues.append(tempPageRank)
    counter += 1
    if __euclideanNorm(delta) < epsilon:
        countls.append(counter)
        break

countdf = pd.pandas.DataFrame(data=countls, columns=['Index'])
vectordf = pd.pandas.DataFrame(data=pagerankvalues, columns=dfcolumns)
dataframe = pd.concat([countdf, vectordf], axis=1)


dataframe.to_csv('PageRank '+name+'.csv')
print(dataframe)

# Automation of Scatter plot
axisX = dict(
    showgrid=True,
    zeroline=False,
    showline=True,
    tick0=0,
    dtick=1,
    showticklabels=True,
    title='Iteration',
    range=[0, counter]
)
axisY = dict(
    showgrid=True,
    zeroline=False,
    tick0=0,
    dtick=0.1,
    showline=True,
    title='PageRank',
    range=[0, 1]
)

layout = go.Layout(
    title=name,
    xaxis=axisX,
    yaxis=axisY
)

graphdata = [] #List containing the graphs data

for col in dataframe:
    if 'Index' not in col:
        trace = go.Scatter(
            name=col,
            mode='lines+markers',
            x=dataframe['Index'],
            y=dataframe[col],
        )
```

```
102         graphdata.append(trace)
103
104 # create figure and export
105 fig = go.Figure(layout=layout, data=graphdata)
106 py.image.save_as(fig, filename=name + '.png')
```

A.4 Taxonomy analysis: Function categorisation

Table containing the a list of functions, description and assigned category.

SOURCE ID	FUNCTION NAME	DESCRIPTION	CATEGORY	NOTES
A2	Measuere Blood Glucose	The user has the ability to see previous glucose measurements and enter new related data. The user is required to select time, units of measurements, reception place and annotation as well as if the measurement is taken before or after a meal.	Monitoring	
	View Blood Glucose Recordings	The user has the ability to see previous glucose measurements and enter new related data.	Inform	
	Injected doses	The user can see/provide information regarding type, dose, type of insulin, reception place and time.	Monitoring	
	Insulin Pump	The user can see/provide information regarding the used insulin pump, and the basal insulin.	Monitoring	
	Meal – Drink	The user can see/provide information regarding the meal and included grams of carbohydrates. Furthermore, the user can select if it was before or after insulin intake and or physical activity.	Monitoring	
	Exercise	The user can see/provide information regarding their physical activity (Type, starting time, duration and impact)	Monitoring	
	Daily Treatment	Total daily insulin and glucose information is provided to the user after statistical analysis. Results are provided in tables and graphs.	Inform	
	Calendar	The user can see/provide notes on special events during their everyday life and set reminders. Additionally, the user can send the data generated to the PMU (server) for the access by the health care provider.	Management	
A3	Symptom Checker	Not available	Unknown	Lack of context
	Drug/ Treatment Information	Not available	Unknown	Lack of context
	First Aid essentials	Not available	Unknown	Lack of context
	Conditions	Not available	Unknown	Lack of context
	Local Health listings	Not available	Unknown	Lack of context
	Diet tracker	Provides the user with calorie breakdowns per meal	Monitoring	
	Weight tracker	Generates a 2d weight chart	Monitoring	
	Work out tracker	Displays the number of calories burned during a workout	Monitoring	
	Medication tracker	Not available	Unknown	Lack of context
	Blood sugar level monitoring	Not available	Unknown	Lack of context
	Glucose tracker	Not available	Unknown	Lack of context
	Activity tracker	Not available	Unknown	Lack of context
	Water consumption tracker	Not available	Unknown	Lack of context
	Weight Tracker	Not available	Unknown	Lack of context

SOURCE ID	FUNCTION NAME	DESCRIPTION	CATEGORY	NOTES
A4	Sensing	A PPG sensor attached to ta mobile phone via Bluetooth gathers the user's heartbeat information.	Monitoring	
	Questionnaire	Function to gather and record information regarding the user's symptoms.	Monitoring	
	Data Processing	Analyses the user's bio metric information, results from questionnaire and calculates a stress index measure.	Monitoring	
	Disease Treatment	Based upon the results from data processing this function recommends a suitable stress solving programme to the user.	Inform	
A6	Heart monitoring	Measures the users heart rate, identifies the resting heart rate (Max – Min and Standard deviation).	Monitoring	
	Physical Activity	Identifies if the user is Walking or Running.	Monitoring	
	Posture	Identifies if the user is standing or lying down	Monitoring	
	Alerts	If an alert is triggered the function identifies the type and explanation of the abnormal situation and reports it to the clinician.	Monitoring	
A7	Health monitoring	Not Available	Unknown	Lack of context
	Appointment Reservation	Not Available	Unknown	Lack of context
	Historical Graphs	Not Available	Unknown	Lack of context
	Suggested diet plans	Not Available	Unknown	Lack of context
	Directions	Not Available	Unknown	Lack of context
	Emergency Alarm	Not Available	Unknown	Lack of context
	Tag scanning	Not Available	Unknown	Lack of context
A9	Data acquisition	Retrieves data from the patients monitoring device, for example blood pressure or glucose level.	Monitoring	
A10	Alert	The application identifies abnormal heart behaviour and issues an alert.	Monitoring	
	Location	Uses GPS to find the users current location	Monitoring	
	Call	Calls the patients primary doctor or emergency service.	Communication	
A11	Search	Locate nearby trained LE therapists	Inform	
	Volume change tracking	Allows user to track limb volume changes over long periods of time, results are shown in the format of a line graph	Monitoring	
	Symptom Report	Function that allows the user to self-record symptoms. The user can choose from 19 options.	Monitoring	
	Alerts	User can set reminders for upcoming appointments and or goals such as weight.	Management	
	Resources	Provides the user with basic information regarding LE, it aims to educate patients with fundamental information, statistics, websites and articles to increase awareness.	Inform	

SOURCE ID	FUNCTION NAME	DESCRIPTION	CATEGORY	NOTES
A13	Peak detection	Detects the R-peak for rhythm assessment	Monitoring	
	Quality assessment		Monitoring	
	Rhythm assessment		Monitoring	
	User feedback	Returns the signal quality index (SQI) and heart rate to the user	Inform	
A14	Patient diary	This function allows the user to record their own test results	Monitoring	
A15	Clock	Daily medication clock that represents 24 hours of the day showing the patients medication schedule.	Management	
	Chart	Presents the user with details regarding medication that was taken correctly or not taken at all and estimated blood plasma concentration	Management	
	Simulation	Provides a simulation of the impact of medication adherence/no adherence to the user.	Inform	
A16	Alert Messaging	A message is sent using SMS once an alert is detected. The Alert message is states the location, time and pattern of the wandering episode will be sent to the caregivers or attending physicians.	Monitoring	
	Location tracking	Scans and obtains the RSSI values from 4 access points alongside the acceleration and orientation signals from the mobile devices on-board sensors	Monitoring	
	Wandering Detection	The algorithm analyses and the location tracking data to detect and classify wandering patterns in real time	Monitoring	
A17	Diabetes Diary (type 1 and 2)	Type 1: Includes everything from Type 2 but also includes easy recording of insulin injections, insulin calculator, nutrition and options for commenting. Type 2: Is composed of a tailored step counter, eating habit registration, educational system with practical tips.	Monitoring	
	Blood Glucose (type 1 and 2)	Transferring reading automatically from a BG monitor	Monitoring	
	SMS Education (type 1 and 2)	User receives educational information relating to their specific type of diabetes	Inform	
	Food Picture Diary	Provides the user with the facility to take a picture of their current meal and append data such as Blood glucose measurements and insulin correction dosage.	Monitoring	
	Physical activity monitoring	Records number of steps a user takes	Monitoring	
	Nutrition Information	Provides the user with food specific information including alternatives suggestions, nutritional information and preparation.	Inform	

SOURCE ID	FUNCTION NAME	DESCRIPTION	CATEGORY	NOTES
A18	AQHI Forecast	Reports the daily Air Quality Health Index information to the user.	Inform	
	Breathing Test	Used to enter the Peak Flow data	Monitoring	
	Today's Symptoms	Enables the user to record daily symptoms	Monitoring	
	Your Current Zone	Calculated based on the uses symptoms and peak flow inputs	Monitoring	
	Your Asthma Plan	Users can view of their complete action plan	Management	
	Breathing Test Graphs	Provides the user with a summary of their peak expiratory flow values	Monitoring	
A19	Electronic medication Blister	To track objectively the dosage and timing of the medication intake using electronic blisters via NFC. Recording the position, timestamp and dosage of the medication.	Monitoring	
A20	Exercise tracking	Allows the user to record length of exercise, Heart rate, Calories burnt, Speed, distance altitude and distance	Monitoring	
	Weight monitoring	Allows the user to track their weight, BMI and if the reading is before or after a meal.	Monitoring	
	Food Intake	Food diary	Monitoring	
	Blood pressure monitoring	Records the users systolic max, diastolic max, systolic and diastolic values.	Monitoring	
	Blood Glucose monitoring	Record the mmol/L classification (Low, normal, high) and if the insulin correction dose was take before or after meal	Monitoring	
	Cholesterol monitoring	Monitors user cholesterol	Monitoring	
	Temperature monitoring	Records the users temperature	Monitoring	
	Reparation monitoring	Monitors the users breathing	Monitoring	
	Bowl Movement monitoring	Allows the user to track their bowl movements	Monitoring	
	Heart rate monitoring	Tracks the users heart rate	Monitoring	
A21	BMI	Calculate the users BMI	Management	
	Eye Test	Series of changing eye charts randomly appear for testing the users visual acuity. User is required to follow an onscreen target until the user makes a mistake. The eye sight index is calculated using the Chinese national standard GB11533-2011	Monitoring	
	Colour Discrimination	Provides the user with 24 pictures each with 5 options. The functionality can detect, fake, red, green and total colour blindness based on the user's selections.	Monitoring	
	Hearing Assessment	The functionality plays a series of sound frequencies. The user is then required to input if the sound could be heard or not.	Monitoring	
	ECG Examination	Records the users Systolic and diastolic values and display it in a graphical format	Monitoring	
	Height and weight	Records the users height and weight	Monitoring	
	Blood pressure	Records the users heart rate and display it in a graphical format	Monitoring	
	Blood Glucose Test	Records the users Systolic and diastolic values and display it in a graphical format	Monitoring	

SOURCE ID	FUNCTION NAME	DESCRIPTION	CATEGORY	NOTES
	Voice Check	Records the users voice to be assessed by a physician	Monitoring	
	Heart sound	Records the users heart beat to be assessed by a physician	Monitoring	
	Tongue Check	Allows the user to take a picture of their tongue using the camera to be assessed by a physician	Monitoring	
	Face record	Allows the user to take a picture of their face using the camera to be assessed by a physician	Monitoring	
	Examination Report	All examination results are added to a report which is stored to be assessed by a physician.	Communication	
B1	Creates user profile	Creates user profile: requests the user to input D.O.B, height, weight, goals.	Management	
	Personal Calorie Calculator	Personal Calorie Calculator: calculates and records daily calorie intake based on user's height and weight	Monitoring	
	Social Interaction	Social Interaction: Allows users to communicate with other users of the application for support.	Communication	
	Weight Loss Graph	Weight Loss Graph: Presents the users weight loss in a line graph.	Inform	
	Badges	Badges: monitors user progress an displays user milestones and achievements	Management	
B2	Q&A	Asks the user a series of simple questions, based on the response provides advice.	Inform	
	Provides Information	Provides the user with advice, tips, guidance, facts, telephone numbers and links to websites	Inform	
	I.C.E contacts	Allows the user to input a series of in case of emergency contacts	Communication	
	Jargon Buster	Describes keywords used throughout the application	Inform	
	Healthcare Service Locater	Using GPS locates nearby healthcare services	Inform	
	Pill Manager	Allows the user to input the pill name, frequency and notes. There is also an option to add alert to pill tracker.	Management	
	Pill Tracker	Reminds user to take pill at a specified time	Management	
B3	Medication Manager	Medication Manager: Allows the user to input their medication (dosage and frequency, times)	Management	
	Medication Reminder	Triggers medication reminders notifying the user to take specific medication and correct dosage at predetermined times throughout the day.	Management	
	Medication History	Medication History: Allows the user to record the medication they taken	Inform	
	Prescription Reminder	Notifies the user when to order/collect their prescription.	Management	
	Contact List	User can input a useful telephone number such as local Pharmacy	Management	

SOURCE ID	FUNCTION NAME	DESCRIPTION	CATEGORY	NOTES
B4	Diabetes Information	Provides the user with various information regarding diabetes.	Inform	
	Annual Review Tracker	Annual Review Tracker: Allows the user to keep track of annual reviews or important test results	Management	
B5	Recipes	Provides the user with a series of recipes for Breakfast, Lunch, Evening meal and puddings.	Inform	
	Meal Mixer	Randomly generates recommendations for healthy balanced meals for breakfast, lunch and evening meaning allowing user to create a shopping list	Inform	
	Shopping List	Provides a detailed list of all the necessary ingredients required to make the provided recipes.	Management	
	Favourite Recipes	Favourite Recipes: Allows user to save their favourite recipes	Management	
	Share with Friends	Allows the user to share recipes with friends via social media	Communication	
	Healthy Eating Information	Provides Information: provides the user with cooking tips, healthy food options	Inform	
B6	Timer	Displays how long the user has left during the exercise session	Management	
	Personal Coach	Pre-recorded exercise sessions to act as a personal coach and records progress	Monitoring	
	Weekly Progress	Shows the user progress through a 9-week exercise programme	Inform	
	Feedback Diary	Allows the user rate how they feel after each exercise session.	Monitoring	
	Exercise Information	Provides educational information such as tips and advice for new runners	Inform	
B7	Record Blood Pressure	Records Blood Pressure*: Allows the user to record their Blood pressure and notes such as activity	Monitoring	
	Reminders	Reminders: Reminds the user to take their blood pressure	Management	
	View Results	View results: User can view their blood pressure recordings in 30, 60, 90 days' intervals or all.	Inform	
	Create User Profile	Create user profile: Requests the user to input, DOB, gender and blood pressure goals (Systolic and Diastolic)	Management	
B8	Create Personal Profile	Requests the number of cigarettes the user smokes per day, time of first cigarette and notification preference	Management	
	Progress Notifications	Presents the user with achievements, progress towards goals or current savings	Inform	
	Savings Calculator	Calculates the user's estimated savings to date, per month and annually.	Management	
	Badges	Badges are used to track users progress and key milestones	Monitoring	
	Personal Motivations	Creates personal motivations that are used to motivate the user	Management	
	Tips for success	Provides information on how to handle cravings, distraction techniques	Inform	

SOURCE ID	FUNCTION NAME	DESCRIPTION	CATEGORY	NOTES
	Additional resources	Provides links to additional resources and helpful information	Inform	
B9	Period Tracker	Records the date the mensuration cycle starts	Monitoring	
	Add Notes	Enables the user to record: intimacy, symptoms, moods, weight and temperature	Monitoring	
	Calendar	Displays a prediction of the month including, mensuration, ovulation and fertile window	Management	
	Logs and Charts	Provides the user logs and graphs relating to the user's menstruation cycle	Inform	
B10	Step counter	Tracks number of steps taken	Monitoring	
	Monitor blood glucose*	Monitor blood glucose	Monitoring	
	Monitor blood pressure*	User can record the systolic and diastolic values, pulse rate (BPM) [requires compatible hardware/ or manually entered]	Monitoring	
	Track hours slept*	Tracks the number of hours slept	Monitoring	
	Track water intake	Tracks the number of glasses (manually)against a pre-determined target	Monitoring	
	Track caffeine intake	Tracks the number of cups against a pre-determined target	Monitoring	
	Monitor Sp02*	Monitor Oxygen saturation	Monitoring	
	Monitor weight*	Monitor weight	Monitoring	
	Calculate BMI	Calculates the users current BMI	Management	
	Create User profile	Includes name, gender, height, weight, activity level, personal bests, weekly summary and program history	Unknown	
	Exercise regimes	Various exercise regimes (10+) utilises various sensors including the accelerometer to track users steps, pace and distance traversed during the exercise programme	Monitoring	
	Track food	User can manually record their food intake for meals throughout the day to calculate their calorie intake based on a target	Monitoring	
	Challenge friends**	Set challenges for friends and view leader boards	Unknown	Locked Premium Feature
	Discover	Provides the user with various health related news articles from various sources	Inform	
B11	Sleep analysis*	Monitors the users sleep using a 3rd party sensor or built in microphone and accelerometer	Monitoring	
	View Trends**	Locked Premium Feature	Unknown	Locked Premium Feature
	View Sleep Phases	Shows an analysis of the users sleep cycle and transitions from the different phases	Inform	
	Alarm clock	Allows the user to set a time to be woken from sleep, the function determines the optimal time based on the user's current sleep phase.	Management	
	Set-Goals**	Allows the user to set seep goals	Unknown	Locked Premium Feature

SOURCE ID	FUNCTION NAME	DESCRIPTION	CATEGORY	NOTES
B12	Activity Generator	Requests the user to input various data such as Play location (Indoor / Outdoor) and how many are taking part. The function will provide a user with a list of recommendations. Once selected the user is presented with information regarding the activity such as how to play, where to play, tips for adults and what you will need.	Management	
	Share	shares recommended activities through social media	Communication	
	Favourites	Stores a list of favourite activities	Management	
B13	About Meningitis	Provides information about meningitis	Inform	
	Sign and Symptom	Signs and Symptoms: information reading the signs and symptoms of meningitis	Inform	
	Call NHS Direct	Allows the user to call... NHS direct (England/Wales and Scotland), ambulance	Communication	
	Call Ambulance	Allows the user to call an ambulance	Communication	
	Search for GP	Opens a link to the NHS choices 'Find services page'	Inform	
	Find nearest A&E	Opens a link to the NHS choices 'Find services page'	Inform	
	Quiz	Assess the user's knowledge of meningitis and provides key information relating to the question	Inform	
	Email	Sends an email to contacts to promote the application	Communication	
	Social Media	Directs the user to the Meningitis now support groups on social media	Communication	
B14	Symptom Assessment	User responds to questions regarding the children's symptoms and provides recommendations of actions	Inform	
	Locate NHS Trust	Enables the user to locate a NHS trust or emergency service	Inform	
	Contact NHS Trust	User can contact a service provider	Communication	
B15	Rate Your Mood	Select current mood by selecting 1 of six the on screen emoticons and add a comments	Monitoring	
	Mood Diary	User selects from an influence from a predefined list. The user can also append comments (details) regarding their current mood	Monitoring	
	Tips	Tips on how to maintain your mood	Inform	
	24/7 Help and Support	Provides the users with contact information to websites and 24/7 hotlines.	Inform	
	Personalised Triggers	Based upon the data entered in the mood diary, the application prompts the user with tips on how to improve or maintain their current mood.	Management	

SOURCE ID	FUNCTION NAME	DESCRIPTION	CATEGORY	NOTES
B16	My Wellbeing	User tracks; their current mood by choosing from a predefined selection of icons, Things they are looking forward too, things they have achieved, things they are grateful for. Events can be added to a calendar for motivation or to improve the user's current mood.	Monitoring	
	Help with Stress	Provides interactive information regarding stress, how it can affect the body and how to manage it and contact information.	Inform	
	Help with Anxiety	Provides interactive information regarding anxiety how it can affect the body and how to manage it and contact information.	Inform	
	Help with depression	Provides interactive information regarding depression how it can affect the body and how to manage it and contact information	Inform	
	Help in a mental crisis	Provides contact information for mental health services.	Inform	
	Relax	User can choose from 3 audio relaxation (breathing) exercises	Management	
	Play Game	The game snake is used as a distraction technique to improve the user's mood. when feeling anxious	Management	
	Share	User can share achievements, things they are grateful for with friends on social media	Communication	
B17	Step Tracker	Records the steps the user takes, including GPS to track distance	Monitoring	
	Weight Tracker	Allows the user to monitor weight	Monitoring	
	Set Goals	Unlock achievements and milestones as a reward to motivate the user to maintain a healthy lifestyle	Management	
	View Records	View previous recorded historical data	Inform	
	View Trends	View current average steps and distances	Inform	
	Explore	Join support groups to collaboratively work together to achieve a collaborative goal / achievement	Communication	
	Socialise	Share progress goals and thoughts with other users of the pacer app	Communication	
B18	Symptom Checker	User selects the part of the body that is affected. Then proceeds to answer questions regarding their symptoms to diagnose a potential condition/s	Monitoring	
	Conditions	Learn from medically reviewed content about the conditions that are effecting you and their potential cause	Inform	
	Drugs and Treatment	Search an extensive data base of drugs and treatment (overview, dosages, uses, side effects, warnings)	Inform	
	First Aid Essentials	Handy guide for administering first aid for minor and major medical emergencies.	Inform	
	Get Directions	Locate nearest healthcare service provider	Inform	
	Email Me	Email content to your personal email for review later	Communication	

SOURCE ID	FUNCTION NAME	DESCRIPTION	CATEGORY	NOTES
	Email Friends	Email content to a friend/s email for review later	Communication	
B19	Pregnancy Information	Provides the user with daily blogs and pregnancy information.	Inform	
	Personal Diary	Allows the user to record Thoughts, feeling and movements	Monitoring	
	Weight Log	Keep track of weight gain over the course of the pregnancy	Monitoring	
	Doctor visit Log	Tool to keep a record appointment	Monitoring	
	Diet, Exercise and Labour information	Provide the user with information regarding healthy diet, safe exercises and what to expect during labour.	Inform	
	Kick Counter	Record the number of 'kicks' from the baby.	Monitoring	
	Contraction Timer	Allows the user to monitor frequency and length of contractions	Monitoring	
	Shopping List	Create and manage a shopping list	Management	
B20	Track Exercise	Allows the user to track distance, pace, steps calories burned, heart rate during exercise	Monitoring	
	Track Heart Rate	Allows the user to record resting and working heart rate	Monitoring	
	View Trends	Allows the user to view weekly, monthly and yearly trends.	Inform	
	Challenges	Unlock achievements and work towards milestones, to further motivate the user. The user may also define custom challenges to create competition between friends that also used the application	Management	
	Monitor Weight	Allows the user to manually input their weight and view progress over time.	Monitoring	
	Monitor H_20 Intake	Allows the user to manually input their H_20 Intake and view progress over time.	Monitoring	

A.5 Taxonomy testing: Test dataset

Test data and classification results used to evaluate the taxonomy.

T1	Title	One You Easy Meals	
Description		Our free Easy Meals app is a great way to eat foods that are healthier for you. You'll find delicious, easy meal ideas to get you going if you're ever short of inspiration. One You is here to help you live more healthily and make the changes that matter. Sometimes it's hard to know what to prepare, or think of new meal ideas. Remembering ingredients and keeping track of calories can be a hassle.	
Available from		Google Play	
Platform		Android	
Version		5.0.0	
Developer		Public Health England	
Function		Function Description	Type
Meal Ideas		Allows the user to search for meal ideas for breakfast, lunch, evening meal and puddings. Each meal is presents an image to the user along with an overview, ingredients and how to prepare. The user also has the option to add ingredients to a shopping cart or add as a favourite	Management
Favourites		Stores the user's favourite meals for easy and convenient access.	Management
Shopping List		The user is required to add to a shopping list a meal from the meal ideas or meal mixer function. This function then manages the users shopping list breaking down each meal and into its required ingredients.	Management
Meal Mixer		Randomly chooses a meal for breakfast, lunch and dinner. The user also has the option to add ingredients to a shopping cart or add as a favourite.	Management
Be Food Smart		Provides the user with educational information based around a healthy diet.	Educational
Seasonal Tips		Based on the current month, this function recommends fruit and vegetables and meals.	Management
Cooking Tips		Provides educational cooking information to the user	Instructional
Cooking Terms		Provides the user with a list of terms and descriptions of commonly used cooking terms	Educational
Notes			

T2	Title	NHS Child Health	
Description		Looking after your child's health, an NHS guide for parents and carers of children aged 0-5 years. Developed by hospital specialists, doctors, health visitors and pharmacists this app will give you lots of useful hints and tips from experts in child health from everything from oral health, upset tummies and diarrhoea to advice on bumps and bruises. This app is intended to help you know what to do, and where to go, when you are looking after an unwell child.	
Available from		Google Play	
Platform		Android	
Version		1.0.0	
Developer		Indigo Multimedia Ltd	
Function		Function Description	Type
Body Area		Allows the user to select an area of the body (head, chest stomach and bottom), that prompts a list of symptoms relating to that area. It then directs the user to advisory information relating to the symptom.	Informative
NHS Services		Provides a list of various types NHS services to the user. The user is required to select an option from a list, then function then provides an overview of the service and locates the nearest one to the user.	Informative
Looking after your child		Provides the user with various advisory information regarding 'looking after your child'. Examples include: Oral health, sun safety and immunisations.	Advisory
The first few months		Provides the user with various advisory information regarding 'The first few months'. Examples include: Crying, teething and feeding.	Advisory
Childhood illnesses		Provides the user with various advisory information regarding common 'childhood illnesses'. Examples include: Asthma, constipation, and rashes.	Advisory
Accidents and prevention		Provides the user with various advisory information regarding 'accidents and prevention'. Examples include: Bumps and bruises, burns and scalds and choking.	Advisory
Notes			

T3	Title	Symptomate Symptom Checker	
Description		Symptomate is an innovative symptom checker designed by doctors that will help you find out more about your symptoms. Just enter basic information about your health complaints and receive a list of potential diagnoses.	
Available from		Google Play / App store	
Platform		Android / iOS	
Version		1.2	
Developer		Infermedica	
Function		Function Description	Type
Diagnose Symptom		Presents the users with various questions relating to their illness to perform a diagnosis	Assessment
Results Details		Provides the user with an explanation of the potential illnesses and recommends the type of doctor you should consult regarding your symptoms	Informative
Notes			

T4	Title	NHS GO	
Description		NHS Go is a new initiative enabling young people to have greater access to medical information. Users can read health related articles, search for nearby services, and find out more information regarding their rights as an NHS customer.	
Available from		Google Play	
Platform		Android	
Version		1.5	
Developer		Soho Strategy	
Function		Function Description	Type
Health A-Z		Provides a complete list that includes educational resources for all the topics below	Educational
Young People Health		Targeted educational resources relating to young people's health. Areas include, registering with your GP, Ramadan, Teens (boys/girls).	Educational
Depression and Anxiety		Provides various educational resources relating to Depression and Anxiety.	Educational
Sleep		Provides various educational resource relating to sleep.	Educational
Sex and relationships		Provides various educational resource relating to Sex and relationships.	Educational
LGBT		Provides various educational resource relating to LGBT.	Educational
Puberty		Provides various educational resource relating to Puberty.	Educational
Eating healthily and exercise		Provides various educational resource relating to Eating healthily and exercise	Educational
Smoking, drugs and alcohol		Provides various educational resource relating to Smoking, drugs and alcohol	Educational
Long term conditions		Provides various educational resource relating to Long term conditions	Educational
Allergies, colds, flu and pains		Provides various educational resource relating to Allergies, colds, flu and pains	Educational
Family health		Provides various educational resource relating to Family health	Educational
Locate local NHS services		Based upon the user's location, the function presents on a map all the surrounding NHS services to the user. Services include GP practices, A&E, clinics, dentists and mental health trusts. The user can select a service and the function will provide information such as address, contact information and referral type.	Informative
Rights		Provides the user with a link to their rights under the NHS.	Informative
Make a complaint		Provides the user with informative information regarding the correct procedure of making a complaint.	Informative
Help		Provides the user with information regarding the NHS non-emergency number 111.	Informative
Notes			

T5	Title	Specsavers Hearing Check	
Description		The Specsavers Hearing Check App will quickly determine your level of hearing loss and whether you would benefit from a full free hearing test in store.	
Available from		Google Play	
Platform		Android	
Version		4.0.50	
Developer		Specsavers Opticians	
Function		Function Description	Type
Take a Hearing Test		Guides the user through a hearing test assessment, the user is required to listen to a sample conversation and respond to four tests by adjusting a slider. Based on the users input the function determines the potential level of hearing loss (Normal, slight loss, medium loss and significant loss)	Assessment
Contact your local Specsavers		Based upon the user's location the function provides details of surrounding Specsavers Stores	Informative
Book an appoint		Allows the user to schedule an appointment	Management
Learn more about hearing loss		Provides various information that informs the user regarding topic such as hearing loss, tell-tale signs and hearing aids	Educational
Notes			

T6	Title	Diabetes in Check: Blood Glucose & Carb Tracker	
Description		The most comprehensive type 2 diabetes app on the market, designed by a Certified Diabetes Educator. You'll get all the tools plus the most up-to-date information you need to control and manage your condition every day.	
Available from		App store	
Platform		iOS	
Version		3.4.2	
Developer		Everyday Health inc	
Function		Function Description	Type
Blood Glucose Tracking		Allows the user to store their blood glucose level throughout the day. The user is required to select what time the reading and the current blood glucose reading is.	Tracker
Medication Tracking		Allows the user to track their medication throughout the day. The user is required to input select from a list the time of day, the type of medication and dosage.	Tracker
Carbohydrate Tracking		Allows the user to keep track of the carbohydrate intake throughout the day. The user Is required to scan the barcode of the product and the function extracts the carbohydrate and records it.	Tracker
Exercise Tracker		Allows the user to keep track of their exercise. The user is required to select a type of exercises from a list and input the length of the exercise.	Tracker
Weight Tracker		Allows the user to track their weight. The user is required to input their current weight	Tracker
Create User Profile		Allows the user to create a user profile that includes their: name, age, height, weight, type of diabetes, email address.	Management
Articles		Provides various educational information relating to diabetes: Living with diabetes, blood sugar control, managing weight & food and fitness	Educational
Food Guide		the user can search from a list of recipes for meals throughout the day. It provides an overview of the meal, nutritional information and instructions on how to make it.	Instructional
Community		Allows users of the Application to post personal questions, share success stories and find support from people with diabetes	Communication
Notes			

T7	Title	Weight Loss Tracker, BMI	
Description		The APP provides you with a detailed body weight diary including graphical statistics on your body weight improvements. When updating your body weight, the APP will track and comment on your achievements and continuously encourage you to improve. With the BMI calculator you can easily calculate your BMI and find out from graphical interpretations whether you are overweight, underweight or have normal weight.	
Available from		Google Play / App store	
Platform		Android / iOS	
Version		1.2	
Developer		aktiWir GmbH	
Function		Function Description	Type
BMI Calculator		Allows the user to calculate their BMI and stores the result The user is required to input: their weight and height.	Assessment
User Profile		The user is required to configure a user profile that includes, Unit (Kg/lb), Gender, Weight and Height.	Management
Protocol		Provides the user with overview of their weight in graphical format	Statistical
Notes			

T8	Title	Stress Check	
Description		Stress Check is a tool developed by psychologists with expertise in stress management and workplace performance. The Stress Check assessment will provide you with an overall stress score that describes your current level of stress. After receiving your overall score, you can deepen your insight by examining the specific areas your stress is affecting you.	
Available from		Google Play / App store	
Platform		Android / iOS	
Version		2.0.0	
Developer		AIIR Consulting LLC	
Function		Function Description	Type
Stress Test		Assesses the level of stress the user is currently under. The user is required to answer a series of questions; their answers determine the 'Level' of stress they are currently experiencing	Assessment
Tracker		Allows the user to view their progress	Statistical
Stress Tools		Provides the user with information (meditation and office yoga) to help reduce the level of stress.	Advisory
Stress University		Provides the user with various informative articles regarding stress	Informative
Notes			

T9	Title	Young Epilepsy	
Description		Free, interactive and personalised app created by Young Epilepsy especially for young people with epilepsy, and parents or carers of a child with epilepsy. This app is multipurpose with an up-to-date information portal, video and diary that helps track and manage seizures and symptoms.	
Available from		App store	
Platform		iOS	
Version		1.8.0	
Developer		THE NATIONAL CENTRE FOR YOUNG PEOPLE WITH EPILEPSY	
Function		Function Description	Type
Create a user profile		Allows the user to create a profile that contains their personal information, condition, medication and emergency details.	Management
Call helpline		Allows the user to call a helpline	Communication
First aid		Provides the user with advice information regarding first aid	Advisory
Video a seizure		Allows the user to record a seizure	Tracker
Learn*		Provides the user with various educational information relating to epilepsy	Educational
Diary		Allows the user to monitor their epilepsy seizures and track elements such as the time of day they occurred, triggers, duration.	Tracker
View Diary		Allows the user to view their diary	Informative
Notes			
Learn functionality – contains various educational information such as: what is epilepsy, seizures, first aid, medication, living with epilepsy, epilepsy and alcohol, bullying, relationships, services available, depression.			

T10	Title	My Last Cigarette - Stop Smoking, Stay Quit	
Description		Quit smoking and stay quit with new My Last Cigarette™ for iOS, the original quit smoking software. Since 1999 My Last Cigarette has successfully helped 1000s of ex smokers stay quit. If you are serious about staying quit then this app could make all the difference!	
Available from		App store	
Platform		iOS	
Version		3.07	
Developer		Mastersoft Ltd	
Function		Function Description	Type
Daily Pic		Presents the user with educational information including images of the effects of smoking	Informative
Savings calculator		Presents the user with an chart with an estimation of their potential savings (per day, per month and per annum).	Statistical
Fatality counter		Presents the user with a counter showing the number of deaths caused by smoking related illnesses.	Statistical
Daily Tracker		Allows the user to track throughout the day the number of: cravings, cigarette smoked, cigarette not smoked as well as some notes.	Tracker
Notes			

Appendix B

Ontology appendices

A collection of appendices that are associated with the development of the PMAD Ontology model.

B.1 Android overview

Major Android operating systems, including versions and API level extracted from [53][1][2].

Codename	Version	API Level
Marshmallow	6	23
Lollipop	5.1	22
	5	21
KitKat	4.4 - 4.4.4	19
Jelly Bean	4.3.x	18
	4.2.x	17
	4.1.x	16
Ice Cream Sandwich	4.0.3 - 4.0.4	15
	4.0.1 - 4.0.2	14
Honeycomb	3.2.x	13
	3.1	12
	3	11
Gingerbread	2.3.3 - 2.3.7	10
	2.3 - 2.3.2	9
Froyo	2.2.x	8
Eclair	2.1	7
	2.0.1	6
	2	5
Donut	1.6	4
Cupcake	1.5	3

[1] Android 1 and Android 1.1 was never released to the public therefore was excluded from the table

[2] API Level 20 was not included in the table as this API revision introduced support for Android Wear devices

B.2 PMAD Ontology description logic expressivity

The table below breaks down the Description Logic expressivity of PMAD Ontology.

Dl Expressivity	Name	Description
$\mathcal{AL}$	Attributive Language	This is the base language which supports the use of the following constructors [183]: Atomic negation(¬), Concept Intersection(⊓), Universal Restriction (∀) and Existential Quantifier (∃).
$\mathcal{C}$	Complex Concept Negation	Is an extension of the base language which supports the use of the full existential quantification and the union (⊔) constructors [139]
$\mathcal{H}$	Role Hierarchy	Is an extension of the base language that supports role (property) subsumption axiom, i.e one role subsumes another [139, 166]
$\mathcal{F}$	Functional Role	Is an extension of the base language that supports functional properties, i.e states that for a given individual, a role can have no more than one value.

Appendix C

Evaluation

A collection of appendices that are associated with the Evaluation of the PMAD Ontology and PMAD Framework.

C.1 Subsumption check results

Presented in the table below are the results yielded from checking the inferred class hierarchy against the expected class hierarchy.

Classification Axiom			Present
AboutServiceProvider	SubClassOf	InformFunction	True
AndroidDevice	SubClassOf	MobileDevice	True
Appointments	SubClassOf	ManagementFunction	True
CallServiceProvider	SubClassOf	CommunicationFunction	True
CarePlanOverview	SubClassOf	InformFunction	True
ChildrensActivityGenerator	SubClassOf	ManagementFunction	True
CommunicationFunction	SubClassOf	Function	True
ComplicationsWithDiabetes	SubClassOf	InformFunction	True
ExerciseDiary	SubClassOf	MonitoringFunction	True
ExerciseInstructions	SubClassOf	InformFunction	True
HealthyMealRecipes	SubClassOf	ManagementFunction	True
HealthySnackMixer	SubClassOf	ManagementFunction	True
HeartAttackAssessment	SubClassOf	MonitoringFunction	True
HeartRateMonitoring	SubClassOf	MonitoringFunction	True
ImportanceOfHealthyEating	SubClassOf	InformFunction	True
InformFunction	SubClassOf	Function	True
IosDevice	SubClassOf	MobileDevice	True
LivingWithDiabetes	SubClassOf	InformFunction	True
ManagementFunction	SubClassOf	Function	True
MealMixer	SubClassOf	ManagementFunction	True
MedicationInformation	SubClassOf	InformFunction	True
MedicationReminder	SubClassOf	ManagementFunction	True
MedicationTracker	SubClassOf	MonitoringFunction	True
MonitoringFunction	SubClassOf	Function	True
Nexus5	SubClassOf	AndroidDevice	True
Nexus7	SubClassOf	AndroidDevice	True
PainDiary	SubClassOf	MonitoringFunction	True
PersonalisedMealPlan	SubClassOf	ManagementFunction	True
PresentHealthRelatedInformation	SubClassOf	InformFunction	True
SamsungGalaxyS4	SubClassOf	AndroidDevice	True
ShoppingList	SubClassOf	ManagementFunction	True
StepTracker	SubClassOf	MonitoringFunction	True

C.2 Disjoint test data and results

Presented in the table below are the results yielded from checking the presence of the disjointwith axiom.

Class	Disjoint With	Present
Accelerometer	AmbientTemperature, Barometer, Compass, Gyroscope, HeartRateMonitor, Hifi, LightSensor, ProximitySensor, RealitiveHumiditySensor, StepCounter, StepDetector	True
Advisory	Educational, Informative, Instructional, Statistical	True
AmbientTemperature	AccelerometerBarometer, Compass, Gyroscope, HeartRateMonitor, Hifi, LightSensor, ProximitySensor, RealitiveHumiditySensor, StepCounter, StepDetector	True
AndroidApi	IosApi	True
AndroidApiLevel10	AndroidApiLevel11, AndroidApiLevel12, AndroidApiLevel13, AndroidApiLevel14, AndroidApiLevel15, AndroidApiLevel16, AndroidApiLevel17, AndroidApiLevel18, AndroidApiLevel19, AndroidApiLevel21, AndroidApiLevel22, AndroidApiLevel23, AndroidApiLevel3, AndroidApiLevel4, AndroidApiLevel5, AndroidApiLevel6, AndroidApiLevel7, AndroidApiLevel8, AndroidApiLevel9	True
AndroidApiLevel11	AndroidApiLevel10AndroidApiLevel12, AndroidApiLevel13, AndroidApiLevel14, AndroidApiLevel15, AndroidApiLevel16, AndroidApiLevel17, AndroidApiLevel18, AndroidApiLevel19, AndroidApiLevel21, AndroidApiLevel22, AndroidApiLevel23, AndroidApiLevel3, AndroidApiLevel4, AndroidApiLevel5, AndroidApiLevel6, AndroidApiLevel7, AndroidApiLevel8, AndroidApiLevel9	True
AndroidApiLevel12	AndroidApiLevel10, AndroidApiLevel11AndroidApiLevel13, AndroidApiLevel14, AndroidApiLevel15, AndroidApiLevel16, AndroidApiLevel17, AndroidApiLevel18, AndroidApiLevel19, AndroidApiLevel21, AndroidApiLevel22, AndroidApiLevel23, AndroidApiLevel3, AndroidApiLevel4, AndroidApiLevel5, AndroidApiLevel6, AndroidApiLevel7, AndroidApiLevel8, AndroidApiLevel9	True

AndroidApiLevel13	AndroidApiLevel10, AndroidApiLevel11, AndroidApiLevel12AndroidApiLevel14, AndroidApiLevel15, AndroidApiLevel16, AndroidApiLevel17, AndroidApiLevel18, AndroidApiLevel19, AndroidApiLevel21, AndroidApiLevel22, AndroidApiLevel23, AndroidApiLevel3, AndroidApiLevel4, AndroidApiLevel5, AndroidApiLevel6, AndroidApiLevel7, AndroidApiLevel8, AndroidApiLevel9	True
AndroidApiLevel14	AndroidApiLevel10, AndroidApiLevel11, AndroidApiLevel12, AndroidApiLevel13AndroidApiLevel15, AndroidApiLevel16, AndroidApiLevel17, AndroidApiLevel18, AndroidApiLevel19, AndroidApiLevel21, AndroidApiLevel22, AndroidApiLevel23, AndroidApiLevel3, AndroidApiLevel4, AndroidApiLevel5, AndroidApiLevel6, AndroidApiLevel7, AndroidApiLevel8, AndroidApiLevel9	True
AndroidApiLevel15	AndroidApiLevel10, AndroidApiLevel11, AndroidApiLevel12, AndroidApiLevel13, AndroidApiLevel14AndroidApiLevel16, AndroidApiLevel17, AndroidApiLevel18, AndroidApiLevel19, AndroidApiLevel21, AndroidApiLevel22, AndroidApiLevel23, AndroidApiLevel3, AndroidApiLevel4, AndroidApiLevel5, AndroidApiLevel6, AndroidApiLevel7, AndroidApiLevel8, AndroidApiLevel9	True
AndroidApiLevel16	AndroidApiLevel10, AndroidApiLevel11, AndroidApiLevel12, AndroidApiLevel13, AndroidApiLevel14, AndroidApiLevel15AndroidApiLevel17, AndroidApiLevel18, AndroidApiLevel19, AndroidApiLevel21, AndroidApiLevel22, AndroidApiLevel23, AndroidApiLevel3, AndroidApiLevel4, AndroidApiLevel5, AndroidApiLevel6, AndroidApiLevel7, AndroidApiLevel8, AndroidApiLevel9	True

AndroidApiLevel17	AndroidApiLevel10, AndroidApiLevel11, AndroidApiLevel12, AndroidApiLevel13, AndroidApiLevel14, AndroidApiLevel15, AndroidApiLevel16AndroidApiLevel18, AndroidApiLevel19, AndroidApiLevel21, AndroidApiLevel22, AndroidApiLevel23, AndroidApiLevel3, AndroidApiLevel4, AndroidApiLevel5, AndroidApiLevel6, AndroidApiLevel7, AndroidApiLevel8, AndroidApiLevel9	True
AndroidApiLevel18	AndroidApiLevel10, AndroidApiLevel11, AndroidApiLevel12, AndroidApiLevel13, AndroidApiLevel14, AndroidApiLevel15, AndroidApiLevel16, AndroidApiLevel17AndroidApiLevel19, AndroidApiLevel21, AndroidApiLevel22, AndroidApiLevel23, AndroidApiLevel3, AndroidApiLevel4, AndroidApiLevel5, AndroidApiLevel6, AndroidApiLevel7, AndroidApiLevel8, AndroidApiLevel9	True
AndroidApiLevel19	AndroidApiLevel10, AndroidApiLevel11, AndroidApiLevel12, AndroidApiLevel13, AndroidApiLevel14, AndroidApiLevel15, AndroidApiLevel16, AndroidApiLevel17, AndroidApiLevel18AndroidApiLevel21, AndroidApiLevel22, AndroidApiLevel23, AndroidApiLevel3, AndroidApiLevel4, AndroidApiLevel5, AndroidApiLevel6, AndroidApiLevel7, AndroidApiLevel8, AndroidApiLevel9	True
AndroidApiLevel21	AndroidApiLevel10, AndroidApiLevel11, AndroidApiLevel12, AndroidApiLevel13, AndroidApiLevel14, AndroidApiLevel15, AndroidApiLevel16, AndroidApiLevel17, AndroidApiLevel18, AndroidApiLevel19AndroidApiLevel22, AndroidApiLevel23, AndroidApiLevel3, AndroidApiLevel4, AndroidApiLevel5, AndroidApiLevel6, AndroidApiLevel7, AndroidApiLevel8, AndroidApiLevel9	True

AndroidApiLevel22	AndroidApiLevel10, AndroidApiLevel11, AndroidApiLevel12, AndroidApiLevel13, AndroidApiLevel14, AndroidApiLevel15, AndroidApiLevel16, AndroidApiLevel17, AndroidApiLevel18, AndroidApiLevel19, AndroidApiLevel21AndroidApiLevel23, AndroidApiLevel3, AndroidApiLevel4, AndroidApiLevel5, AndroidApiLevel6, AndroidApiLevel7, AndroidApiLevel8, AndroidApiLevel9	True
AndroidApiLevel23	AndroidApiLevel10, AndroidApiLevel11, AndroidApiLevel12, AndroidApiLevel13, AndroidApiLevel14, AndroidApiLevel15, AndroidApiLevel16, AndroidApiLevel17, AndroidApiLevel18, AndroidApiLevel19, AndroidApiLevel21, AndroidApiLevel22AndroidApiLevel3, AndroidApiLevel4, AndroidApiLevel5, AndroidApiLevel6, AndroidApiLevel7, AndroidApiLevel8, AndroidApiLevel9	True
AndroidApiLevel3	AndroidApiLevel10, AndroidApiLevel11, AndroidApiLevel12, AndroidApiLevel13, AndroidApiLevel14, AndroidApiLevel15, AndroidApiLevel16, AndroidApiLevel17, AndroidApiLevel18, AndroidApiLevel19, AndroidApiLevel21, AndroidApiLevel22, AndroidApiLevel23AndroidApiLevel4, AndroidApiLevel5, AndroidApiLevel6, AndroidApiLevel7, AndroidApiLevel8, AndroidApiLevel9	True
AndroidApiLevel4	AndroidApiLevel10, AndroidApiLevel11, AndroidApiLevel12, AndroidApiLevel13, AndroidApiLevel14, AndroidApiLevel15, AndroidApiLevel16, AndroidApiLevel17, AndroidApiLevel18, AndroidApiLevel19, AndroidApiLevel21, AndroidApiLevel22, AndroidApiLevel23, AndroidApiLevel3AndroidApiLevel5, AndroidApiLevel6, AndroidApiLevel7, AndroidApiLevel8, AndroidApiLevel9	True

AndroidApiLevel5	AndroidApiLevel10, AndroidApiLevel11, AndroidApiLevel12, AndroidApiLevel13, AndroidApiLevel14, AndroidApiLevel15, AndroidApiLevel16, AndroidApiLevel17, AndroidApiLevel18, AndroidApiLevel19, AndroidApiLevel21, AndroidApiLevel22, AndroidApiLevel23, AndroidApiLevel3, AndroidApiLevel4AndroidApiLevel6, AndroidApiLevel7, AndroidApiLevel8, AndroidApiLevel9	True
AndroidApiLevel6	AndroidApiLevel10, AndroidApiLevel11, AndroidApiLevel12, AndroidApiLevel13, AndroidApiLevel14, AndroidApiLevel15, AndroidApiLevel16, AndroidApiLevel17, AndroidApiLevel18, AndroidApiLevel19, AndroidApiLevel21, AndroidApiLevel22, AndroidApiLevel23, AndroidApiLevel3, AndroidApiLevel4, AndroidApiLevel5AndroidApiLevel7, AndroidApiLevel8, AndroidApiLevel9	True
AndroidApiLevel7	AndroidApiLevel10, AndroidApiLevel11, AndroidApiLevel12, AndroidApiLevel13, AndroidApiLevel14, AndroidApiLevel15, AndroidApiLevel16, AndroidApiLevel17, AndroidApiLevel18, AndroidApiLevel19, AndroidApiLevel21, AndroidApiLevel22, AndroidApiLevel23, AndroidApiLevel3, AndroidApiLevel4, AndroidApiLevel5, AndroidApiLevel6AndroidApiLevel8, AndroidApiLevel9	True
AndroidApiLevel8	AndroidApiLevel10, AndroidApiLevel11, AndroidApiLevel12, AndroidApiLevel13, AndroidApiLevel14, AndroidApiLevel15, AndroidApiLevel16, AndroidApiLevel17, AndroidApiLevel18, AndroidApiLevel19, AndroidApiLevel21, AndroidApiLevel22, AndroidApiLevel23, AndroidApiLevel3, AndroidApiLevel4, AndroidApiLevel5, AndroidApiLevel6, AndroidApiLevel7AndroidApiLevel9	True

AndroidApiLevel9	AndroidApiLevel10, AndroidApiLevel11, AndroidApiLevel12, AndroidApiLevel13, AndroidApiLevel14, AndroidApiLevel15, AndroidApiLevel16, AndroidApiLevel17, AndroidApiLevel18, AndroidApiLevel19, AndroidApiLevel21, AndroidApiLevel22, AndroidApiLevel23, AndroidApiLevel3, AndroidApiLevel4, AndroidApiLevel5, AndroidApiLevel6, AndroidApiLevel7, AndroidApiLevel8	True
AndroidDevice	IosDevice	True
Api	FunctionLogic, FunctionType, Hardware, Manufacturer, PersonalisedComponent	True
Apple	Asus, LG, Samsung	True
AppointmentScheduleLogic	ChildrenActivityGeneratorLogic, ExerciseDiaryLogic, HealthySnackMixerLogic, HeartAttackAssessmentLogic, HeartRateMonitoringLogic, IngredientsShoppingListLogic, MakeCallLogic, MealMixerLogic, MealPlanLogic, MedicationReminderLogic, MedicationTrackerLogic, PainDiaryLogic, PresentOffilineInformationLogic, PresentWebBasedInformationLogic, RecipesLogic, StepTrackerLogic	True
Assessment	Tracking	True
Asus	AppleLG, Samsung, LG	True
Audio	Bluetooth, Camera, Display, FingerprintReader, Infrared, Location, Nfc, Sensor, Telephony, Usb, Wifi	True
Barometer	Accelerometer, AmbientTemperatureCompass, Gyroscope, HeartRateMonitor, Hifi, LightSensor, ProximitySensor, RealitiveHumiditySensor, StepCounter, StepDetector	True
Bluetooth	AudioCamera, Display, FingerprintReader, Infrared, Location, Nfc, Sensor, Telephony, Usb, Wifi	True
Camera	Audio, BluetoothDisplay, FingerprintReader, Infrared, Location, Nfc, Sensor, Telephony, Usb, Wifi	True
Cdma	Gsm	True
ChildrenActivityGeneratorLogic	AppointmentScheduleLogicExerciseDiaryLogic, HealthySnackMixerLogic, HeartAttackAssessmentLogic, HeartRateMonitoringLogic, IngredientsShoppingListLogic, MakeCallLogic, MealMixerLogic, MealPlanLogic, MedicationReminderLogic, MedicationTrackerLogic, PainDiaryLogic, PresentOffilineInformationLogic, PresentWebBasedInformationLogic, RecipesLogic, StepTrackerLogic	True
Communication	Inform, Management, Monitoring	True

CommunicationFunction	InformFunction, ManagementFunction, MonitoringFunction	True
Compass	Accelerometer, AmbientTemperature, BarometerGyroscope, HeartRateMonitor, Hifi, LightSensor, ProximitySensor, RealitiveHumiditySensor, StepCounter, StepDetector	True
Display	Audio, Bluetooth, CameraFingerprintReader, Infrared, Location, Nfc, Sensor, Telephony, Usb, Wifi	True
Educational	AdvisoryInformative, Instructional, Statistical	True
ExerciseDiaryLogic	AppointmentScheduleLogic, ChildrenActivityGenerator-LogicHealthySnackMixerLogic, HeartAttackAssessmentLogic, HeartRateMonitoringLogic, IngredientsShoppingListLogic, MakeCallLogic, MealMixerLogic, MealPlanLogic, MedicationReminderLogic, MedicationTrackerLogic, PainDiaryLogic, PresentOffilineInformationLogic, PresentWebBasedInformationLogic, RecipesLogic, StepTrackerLogic	True
FingerprintReader	Audio, Bluetooth, Camera, DisplayInfrared, Location, Nfc, Sensor, Telephony, Usb, Wifi	True
Function	MobileDevice	True
FunctionLogic	ApiFunctionType, Hardware, Manufacturer, PersonalisedComponent	True
FunctionType	Api, FunctionLogicHardware, Manufacturer, PersonalisedComponent	True
Gsm	Cdma	True
Gyroscope	Accelerometer, AmbientTemperature, Barometer, CompassHeartRateMonitor, Hifi, LightSensor, ProximitySensor, RealitiveHumiditySensor, StepCounter, StepDetector	True
Hardware	Api, FunctionLogic, FunctionTypeManufacturer, PersonalisedComponent	True
HealthySnackMixerLogic	AppointmentScheduleLogic, ChildrenActivityGeneratorLogic, ExerciseDiaryLogicHeartAttackAssessmentLogic, HeartRateMonitoringLogic, IngredientsShoppingListLogic, MakeCallLogic, MealMixerLogic, MealPlanLogic, MedicationReminderLogic, MedicationTrackerLogic, PainDiaryLogic, PresentOffilineInformationLogic, PresentWebBasedInformationLogic, RecipesLogic, StepTrackerLogic	True

HeartAttackAssessmentLogic	AppointmentScheduleLogic, ChildrenActivityGeneratorLogic, ExerciseDiaryLogic, HealthySnackMixerLogicHeartRateMonitoringLogic, IngredientsShoppingListLogic, MakeCallLogic, MealMixerLogic, MealPlanLogic, MedicationReminderLogic, MedicationTrackerLogic, PainDiaryLogic, PresentOffilineInformationLogic, PresentWebBasedInformationLogic, RecipesLogic, StepTrackerLogic	True
HeartRateMonitor	Accelerometer, AmbientTemperature, Barometer, Compass, GyroscopeHifi, LightSensor, ProximitySensor, RealitiveHumiditySensor, StepCounter, StepDetector	True
HeartRateMonitoringLogic	AppointmentScheduleLogic, ChildrenActivityGeneratorLogic, ExerciseDiaryLogic, HealthySnackMixerLogic, HeartAttackAssessment-LogicIngredientsShoppingListLogic, MakeCallLogic, MealMixerLogic, MealPlanLogic, MedicationReminderLogic, MedicationTrackerLogic, PainDiaryLogic, PresentOffilineInformationLogic, PresentWebBasedInformationLogic, RecipesLogic, StepTrackerLogic	True
Hifi	Accelerometer, AmbientTemperature, Barometer, Compass, Gyroscope, HeartRateMonitorLightSensor, ProximitySensor, RealitiveHumiditySensor, StepCounter, StepDetector	True
Inform	CommunicationManagement, Monitoring	True
Informative	Advisory, EducationalInstructional, Statistical	True
InformFunction	CommunicationFunctionManagementFunction, MonitoringFunction	True
Infrared	Audio, Bluetooth, Camera, Display, FingerprintReaderLocation, Nfc, Sensor, Telephony, Usb, Wifi	True
IngredientsShoppingListLogic	AppointmentScheduleLogic, ChildrenActivityGeneratorLogic, ExerciseDiaryLogic, HealthySnackMixerLogic, HeartAttackAssessmentLogic, HeartRateMonitoringLogicMakeCallLogic, MealMixerLogic, MealPlanLogic, MedicationReminderLogic, MedicationTrackerLogic, PainDiaryLogic, PresentOffilineInformationLogic, PresentWebBasedInformationLogic, RecipesLogic, StepTrackerLogic	True
Instructional	Advisory, Educational, InformativeStatistical	True
IosApi	AndroidApi	True
IosDevice	AndroidDevice	True
LG	Apple, Asus, Samsung	True

LightSensor	Accelerometer, AmbientTemperature, Barometer, Compass, Gyroscope, HeartRateMonitor, HifiProximitySensor, RealitiveHumiditySensor, StepCounter, StepDetector	True
Location	Audio, Bluetooth, Camera, Display, FingerprintReader, InfraredNfc, Sensor, Telephony, Usb, Wifi	True
Loudspeaker	Microphone	True
MakeCallLogic	AppointmentScheduleLogic, ChildrenActivityGeneratorLogic, ExerciseDiaryLogic, HealthySnackMixerLogic, HeartAttackAssessmentLogic, HeartRateMonitoringLogic, IngredientsShoppingListLogicMealMixerLogic, MealPlanLogic, MedicationReminderLogic, MedicationTrackerLogic, PainDiaryLogic, PresentOffilineInformationLogic, PresentWebBasedInformationLogic, RecipesLogic, StepTrackerLogic	True
Management	Communication, InformMonitoring	True
ManagementFunction	CommunicationFunction, InformFunctionMonitoringFunction	True
Manufacturer	Api, FunctionLogic, FunctionType, HardwarePersonalisedComponent	True
MealMixerLogic	AppointmentScheduleLogic, ChildrenActivityGeneratorLogic, ExerciseDiaryLogic, HealthySnackMixerLogic, HeartAttackAssessmentLogic, HeartRateMonitoringLogic, IngredientsShoppingListLogic, MakeCallLogicMealPlanLogic, MedicationReminderLogic, MedicationTrackerLogic, PainDiaryLogic, PresentOffilineInformationLogic, PresentWebBasedInformationLogic, RecipesLogic, StepTrackerLogic	True
MealPlanLogic	AppointmentScheduleLogic, ChildrenActivityGeneratorLogic, ExerciseDiaryLogic, HealthySnackMixerLogic, HeartAttackAssessmentLogic, HeartRateMonitoringLogic, IngredientsShoppingListLogic, MakeCallLogic, MealMixerLogicMedicationReminderLogic, MedicationTrackerLogic, PainDiaryLogic, PresentOffilineInformationLogic, PresentWebBasedInformationLogic, RecipesLogic, StepTrackerLogic	True

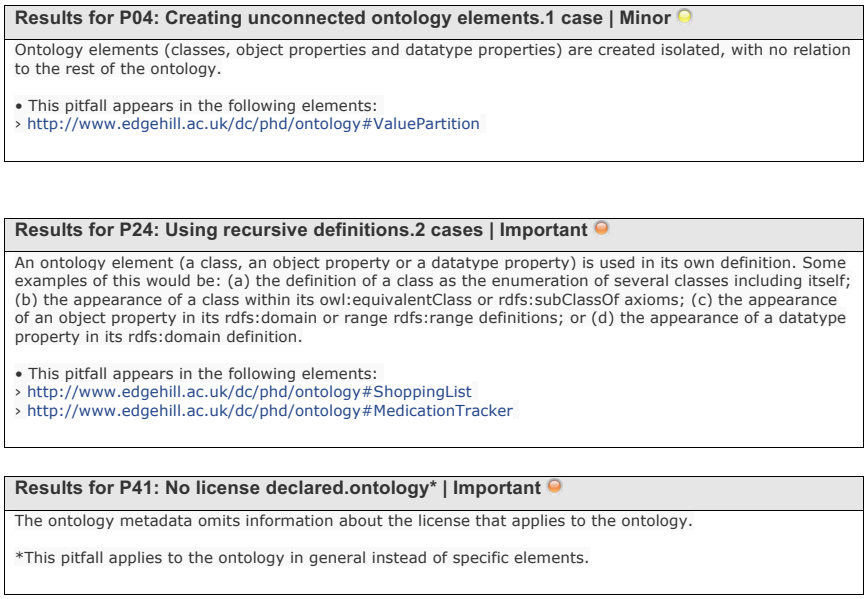
MedicationReminderLogic	AppointmentScheduleLogic, ChildrenActivityGeneratorLogic, ExerciseDiaryLogic, HealthySnackMixerLogic, HeartAttackAssessmentLogic, HeartRateMonitoringLogic, IngredientsShoppingListLogic, MakeCallLogic, MealMixerLogic, MealPlanLogicMedicationTrackerLogic, PainDiaryLogic, PresentOffilineInformationLogic, PresentWebBasedInformationLogic, RecipesLogic, StepTrackerLogic	True
MedicationTrackerLogic	AppointmentScheduleLogic, ChildrenActivityGeneratorLogic, ExerciseDiaryLogic, HealthySnackMixerLogic, HeartAttackAssessmentLogic, HeartRateMonitoringLogic, IngredientsShoppingListLogic, MakeCallLogic, MealMixerLogic, MealPlanLogic, MedicationReminderLogicPainDiaryLogic, PresentOffilineInformationLogic, PresentWebBasedInformationLogic, RecipesLogic, StepTrackerLogic	True
Microphone	Loudspeaker	True
MobileDevice	Function	True
Monitoring	Communication, Inform, Management	True
MonitoringFunction	CommunicationFunction, InformFunction, ManagementFunction	True
Nfc	Audio, Bluetooth, Camera, Display, FingerprintReader, Infrared, LocationSensor, Telephony, Usb, Wifi	True
PainDiaryLogic	AppointmentScheduleLogic, ChildrenActivityGeneratorLogic, ExerciseDiaryLogic, HealthySnackMixerLogic, HeartAttackAssessmentLogic, HeartRateMonitoringLogic, IngredientsShoppingListLogic, MakeCallLogic, MealMixerLogic, MealPlanLogic, MedicationReminderLogic, MedicationTrackerLogicP-resentOffilineInformationLogic, PresentWebBasedInformationLogic, RecipesLogic, StepTrackerLogic	True
PersonalisedComponent	Api, FunctionLogic, FunctionType, Hardware, Manufacturer	True

PresentOffilineInformationLogic	AppointmentScheduleLogic, ChildrenActivityGeneratorLogic, ExerciseDiaryLogic, HealthySnackMixerLogic, HeartAttackAssessmentLogic, HeartRateMonitoringLogic, IngredientsShoppingListLogic, MakeCallLogic, MealMixerLogic, MealPlanLogic, MedicationReminderLogic, MedicationTrackerLogic, PainDiaryLogicPresentWebBasedInformationLogic, RecipesLogic, StepTrackerLogic	True
PresentWebBasedInformationLogic	AppointmentScheduleLogic, ChildrenActivityGeneratorLogic, ExerciseDiaryLogic, HealthySnackMixerLogic, HeartAttackAssessmentLogic, HeartRateMonitoringLogic, IngredientsShoppingListLogic, MakeCallLogic, MealMixerLogic, MealPlanLogic, MedicationReminderLogic, MedicationTrackerLogic, PainDiaryLogic, PresentOffilineInformationLogicRecipesLogic, StepTrackerLogic	True
ProximitySensor	Accelerometer, AmbientTemperature, Barometer, Compass, Gyroscope, HeartRateMonitor, Hifi, LightSensorRealitiveHumiditySensor, StepCounter, StepDetector	True
RealitiveHumiditySensor	Accelerometer, AmbientTemperature, Barometer, Compass, Gyroscope, HeartRateMonitor, Hifi, LightSensor, ProximitySensorStepCounter, StepDetector	True
RecipesLogic	AppointmentScheduleLogic, ChildrenActivityGeneratorLogic, ExerciseDiaryLogic, HealthySnackMixerLogic, HeartAttackAssessmentLogic, HeartRateMonitoringLogic, IngredientsShoppingListLogic, MakeCallLogic, MealMixerLogic, MealPlanLogic, MedicationReminderLogic, MedicationTrackerLogic, PainDiaryLogic, PresentOffilineInformationLogic, PresentWebBasedInformationLogicStepTrackerLogic	True
Samsung	Apple, Asus, LG,LG	True
Sensor	Audio, Bluetooth, Camera, Display, FingerprintReader, Infrared, Location, NfcTelephony, Usb, Wifi	True
Statistical	Advisory, Educational, Informative, Instructional	True
StepCounter	Accelerometer, AmbientTemperature, Barometer, Compass, Gyroscope, HeartRateMonitor, Hifi, LightSensor, ProximitySensor, RealitiveHumiditySensorStepDetector	True
StepDetector	Accelerometer, AmbientTemperature, Barometer, Compass, Gyroscope, HeartRateMonitor, Hifi, LightSensor, ProximitySensor, RealitiveHumiditySensor, StepCounter	True

StepTrackerLogic	AppointmentScheduleLogic, ChildrenActivityGeneratorLogic, ExerciseDiaryLogic, HealthySnackMixerLogic, HeartAttackAssessmentLogic, HeartRateMonitoringLogic, IngredientsShoppingListLogic, MakeCallLogic, MealMixerLogic, MealPlanLogic, MedicationReminderLogic, MedicationTrackerLogic, PainDiaryLogic, PresentOffilineInformationLogic, PresentWebBasedInformationLogic, RecipesLogic	True
Telephony	Audio, Bluetooth, Camera, Display, FingerprintReader, Infrared, Location, Nfc, SensorUsb, Wifi	True
Tracking	Assessment	True
Usb	Audio, Bluetooth, Camera, Display, FingerprintReader, Infrared, Location, Nfc, Sensor, TelephonyWifi	True
Wifi	Audio, Bluetooth, Camera, Display, FingerprintReader, Infrared, Location, Nfc, Sensor, Telephony, Usb	True

C.3 OOPS! validation results

Presented below is a screen shot of the OOPS! evaluation report for the PMAD Ontology.



Results for P04: Creating unconnected ontology elements.1 case | Minor

Ontology elements (classes, object properties and datatype properties) are created isolated, with no relation to the rest of the ontology.

• This pitfall appears in the following elements:
› http://www.edgehill.ac.uk/dc/phd/ontology#ValuePartition

Results for P24: Using recursive definitions.2 cases | Important

An ontology element (a class, an object property or a datatype property) is used in its own definition. Some examples of this would be: (a) the definition of a class as the enumeration of several classes including itself; (b) the appearance of a class within its owl:equivalentClass or rdfs:subClassOf axioms; (c) the appearance of an object property in its rdfs:domain or range rdfs:range definitions; or (d) the appearance of a datatype property in its rdfs:domain definition.

• This pitfall appears in the following elements:
› http://www.edgehill.ac.uk/dc/phd/ontology#ShoppingList
› http://www.edgehill.ac.uk/dc/phd/ontology#MedicationTracker

Results for P41: No license declared.ontology* | Important

The ontology metadata omits information about the license that applies to the ontology.

*This pitfall applies to the ontology in general instead of specific elements.

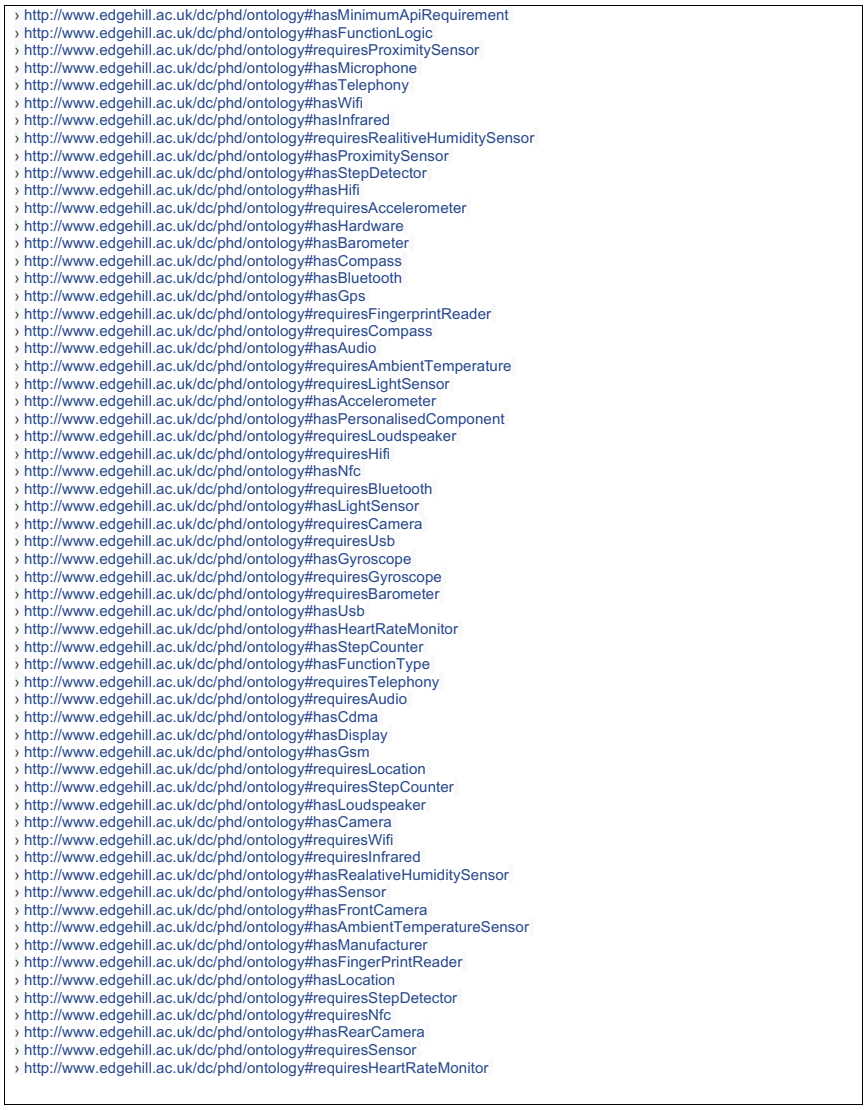
Results for P08: Missing annotations.203 cases | Minor

This pitfall consists in creating an ontology element and failing to provide human readable annotations attached to it. Consequently, ontology elements lack annotation properties that label them (e.g. rdfs:label, lemon:LexicalEntry, skos:prefLabel or skos:altLabel) or that define them (e.g. rdfs:comment or dc:description). This pitfall is related to the guidelines provided in [5].

• The following elements have neither rdfs:label or rdfs:comment (nor skos:definition) defined:
› http://www.edgehill.ac.uk/dc/phd/ontology#Accelerometer
› http://www.edgehill.ac.uk/dc/phd/ontology#Communication
› http://www.edgehill.ac.uk/dc/phd/ontology#Inform
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresMicrophone

• The following elements have no rdfs:label defined:
› http://www.edgehill.ac.uk/dc/phd/ontology#LivingWithDiabetesInformation
› http://www.edgehill.ac.uk/dc/phd/ontology#AndroidApiLevel6
› http://www.edgehill.ac.uk/dc/phd/ontology#Appointments
› http://www.edgehill.ac.uk/dc/phd/ontology#ValuePartition
› http://www.edgehill.ac.uk/dc/phd/ontology#Compass
› http://www.edgehill.ac.uk/dc/phd/ontology#RearCamera
› http://www.edgehill.ac.uk/dc/phd/ontology#Usb
› http://www.edgehill.ac.uk/dc/phd/ontology#ExerciseInstructions
› http://www.edgehill.ac.uk/dc/phd/ontology#Infrared
› http://www.edgehill.ac.uk/dc/phd/ontology#PainDiaryLogic
› http://www.edgehill.ac.uk/dc/phd/ontology#MedicationInformation
› http://www.edgehill.ac.uk/dc/phd/ontology#Wifi
› http://www.edgehill.ac.uk/dc/phd/ontology#ChildrensActivityGenerator
› http://www.edgehill.ac.uk/dc/phd/ontology#DrugFrequency
› http://www.edgehill.ac.uk/dc/phd/ontology#Advisory
› http://www.edgehill.ac.uk/dc/phd/ontology#AndroidApiLevel7
› http://www.edgehill.ac.uk/dc/phd/ontology#Tracking
› http://www.edgehill.ac.uk/dc/phd/ontology#Nexus5
› http://www.edgehill.ac.uk/dc/phd/ontology#MedicationTracker
› http://www.edgehill.ac.uk/dc/phd/ontology#HealthyMealRecipes
› http://www.edgehill.ac.uk/dc/phd/ontology#CommunicationFunction
› http://www.edgehill.ac.uk/dc/phd/ontology#Cdma
› http://www.edgehill.ac.uk/dc/phd/ontology#StepTrackerLogic
› http://www.edgehill.ac.uk/dc/phd/ontology#HealthySnackMixer
› http://www.edgehill.ac.uk/dc/phd/ontology#AboutServiceProvider
› http://www.edgehill.ac.uk/dc/phd/ontology#ComplicationsWithDiabetes
› http://www.edgehill.ac.uk/dc/phd/ontology#IosApi
› http://www.edgehill.ac.uk/dc/phd/ontology#ExerciseDiaryLogic
› http://www.edgehill.ac.uk/dc/phd/ontology#IosDevice
› http://www.edgehill.ac.uk/dc/phd/ontology#AndroidApiLevel3
› http://www.edgehill.ac.uk/dc/phd/ontology#Telephony
› http://www.edgehill.ac.uk/dc/phd/ontology#Instructional
› http://www.edgehill.ac.uk/dc/phd/ontology#AndroidApiLevel10
› http://www.edgehill.ac.uk/dc/phd/ontology#Informative
› http://www.edgehill.ac.uk/dc/phd/ontology#SamsungGlaxayS4
› http://www.edgehill.ac.uk/dc/phd/ontology#HeartRateMonitoring
› http://www.edgehill.ac.uk/dc/phd/ontology#MealPlan
› http://www.edgehill.ac.uk/dc/phd/ontology#StepCounter
› http://www.edgehill.ac.uk/dc/phd/ontology#HeartRateMonitor
› http://www.edgehill.ac.uk/dc/phd/ontology#StepTracker
› http://www.edgehill.ac.uk/dc/phd/ontology#AndroidApiLevel9
› http://www.edgehill.ac.uk/dc/phd/ontology#Nexus7
› http://www.edgehill.ac.uk/dc/phd/ontology#RecipesLogic
› http://www.edgehill.ac.uk/dc/phd/ontology#Asus
› http://www.edgehill.ac.uk/dc/phd/ontology#AndroidApiLevel21
› http://www.edgehill.ac.uk/dc/phd/ontology#AndroidApiLevel16
› http://www.edgehill.ac.uk/dc/phd/ontology#LG
› http://www.edgehill.ac.uk/dc/phd/ontology#Management
› http://www.edgehill.ac.uk/dc/phd/ontology#Hifi
› http://www.edgehill.ac.uk/dc/phd/ontology#AndroidDevice
› http://www.edgehill.ac.uk/dc/phd/ontology#PersonalisedMealPlan
› http://www.edgehill.ac.uk/dc/phd/ontology#Hardware
› http://www.edgehill.ac.uk/dc/phd/ontology#HeartRateMonitoringLogic
› http://www.edgehill.ac.uk/dc/phd/ontology#AndroidApiLevel8
› http://www.edgehill.ac.uk/dc/phd/ontology#MonitoringFunction
› http://www.edgehill.ac.uk/dc/phd/ontology#Statistical
› http://www.edgehill.ac.uk/dc/phd/ontology#AndroidApiLevel15
› http://www.edgehill.ac.uk/dc/phd/ontology#Monitoring
› http://www.edgehill.ac.uk/dc/phd/ontology#Gsm
› http://www.edgehill.ac.uk/dc/phd/ontology#AndroidApiLevel12
› http://www.edgehill.ac.uk/dc/phd/ontology#PatientCarePlan

› http://www.edgehill.ac.uk/dc/phd/ontology#Api
› http://www.edgehill.ac.uk/dc/phd/ontology#FunctionLogic
› http://www.edgehill.ac.uk/dc/phd/ontology#ExercisePlan
› http://www.edgehill.ac.uk/dc/phd/ontology#MobileDevice
› http://www.edgehill.ac.uk/dc/phd/ontology#ChildrenActivityGeneratorLogic
› http://www.edgehill.ac.uk/dc/phd/ontology#ManagementFunction
› http://www.edgehill.ac.uk/dc/phd/ontology#LightSensor
› http://www.edgehill.ac.uk/dc/phd/ontology#StepDetector
› http://www.edgehill.ac.uk/dc/phd/ontology#PersonalisedComponent
› http://www.edgehill.ac.uk/dc/phd/ontology#AndroidApiLevel11
› http://www.edgehill.ac.uk/dc/phd/ontology#MedicationReminderLogic
› http://www.edgehill.ac.uk/dc/phd/ontology#AndroidApiLevel23
› http://www.edgehill.ac.uk/dc/phd/ontology#DrugInformation
› http://www.edgehill.ac.uk/dc/phd/ontology#Loudspeaker
› http://www.edgehill.ac.uk/dc/phd/ontology#MakeCallLogic
› http://www.edgehill.ac.uk/dc/phd/ontology#LivingWithDiabetes
› http://www.edgehill.ac.uk/dc/phd/ontology#MealMixer
› http://www.edgehill.ac.uk/dc/phd/ontology#HealthySnackMixerLogic
› http://www.edgehill.ac.uk/dc/phd/ontology#FingerprintReader
› http://www.edgehill.ac.uk/dc/phd/ontology#CarePlanOverview
› http://www.edgehill.ac.uk/dc/phd/ontology#AndroidApiLevel22
› http://www.edgehill.ac.uk/dc/phd/ontology#AndroidApiLevel17
› http://www.edgehill.ac.uk/dc/phd/ontology#IngredientsShoppingListLogic
› http://www.edgehill.ac.uk/dc/phd/ontology#Apple
› http://www.edgehill.ac.uk/dc/phd/ontology#Samsung
› http://www.edgehill.ac.uk/dc/phd/ontology#AndroidApiLevel18
› http://www.edgehill.ac.uk/dc/phd/ontology#DrugDosage
› http://www.edgehill.ac.uk/dc/phd/ontology#ShoppingList
› http://www.edgehill.ac.uk/dc/phd/ontology#AndroidApiLevel13
› http://www.edgehill.ac.uk/dc/phd/ontology#AndroidApiLevel5
› http://www.edgehill.ac.uk/dc/phd/ontology#Gyroscope
› http://www.edgehill.ac.uk/dc/phd/ontology#Sensor
› http://www.edgehill.ac.uk/dc/phd/ontology#Barometer
› http://www.edgehill.ac.uk/dc/phd/ontology#RealitiveHumiditySensor
› http://www.edgehill.ac.uk/dc/phd/ontology#Educational
› http://www.edgehill.ac.uk/dc/phd/ontology#ServiceProviderTelephoneNumber
› http://www.edgehill.ac.uk/dc/phd/ontology#Manufacturer
› http://www.edgehill.ac.uk/dc/phd/ontology#HeartAttackAssessment
› http://www.edgehill.ac.uk/dc/phd/ontology#FrontCamera
› http://www.edgehill.ac.uk/dc/phd/ontology#FunctionType
› http://www.edgehill.ac.uk/dc/phd/ontology#ServiceProviderInformation
› http://www.edgehill.ac.uk/dc/phd/ontology#ImportanceOfHealthyEating
› http://www.edgehill.ac.uk/dc/phd/ontology#PainDiary
› http://www.edgehill.ac.uk/dc/phd/ontology#Drug
› http://www.edgehill.ac.uk/dc/phd/ontology#AppointmentSchedule
› http://www.edgehill.ac.uk/dc/phd/ontology#AndroidApiLevel14
› http://www.edgehill.ac.uk/dc/phd/ontology#AndroidApiLevel4
› http://www.edgehill.ac.uk/dc/phd/ontology#CallServiceProvider
› http://www.edgehill.ac.uk/dc/phd/ontology#Assessment
› http://www.edgehill.ac.uk/dc/phd/ontology#Bluetooth
› http://www.edgehill.ac.uk/dc/phd/ontology#Location
› http://www.edgehill.ac.uk/dc/phd/ontology#HeartAttackAssessmentLogic
› http://www.edgehill.ac.uk/dc/phd/ontology#Audio
› http://www.edgehill.ac.uk/dc/phd/ontology#Display
› http://www.edgehill.ac.uk/dc/phd/ontology#Microphone
› http://www.edgehill.ac.uk/dc/phd/ontology#HealthyEatingInformation
› http://www.edgehill.ac.uk/dc/phd/ontology#Camera
› http://www.edgehill.ac.uk/dc/phd/ontology#PresentOffilineInformationLogic
› http://www.edgehill.ac.uk/dc/phd/ontology#InformFunction
› http://www.edgehill.ac.uk/dc/phd/ontology#Nfc
› http://www.edgehill.ac.uk/dc/phd/ontology#MedicationReminder
› http://www.edgehill.ac.uk/dc/phd/ontology#AppointmentScheduleLogic
› http://www.edgehill.ac.uk/dc/phd/ontology#MedicationTrackerLogic
› http://www.edgehill.ac.uk/dc/phd/ontology#AndroidApi
› http://www.edgehill.ac.uk/dc/phd/ontology#ComplicationsWithDiabetesInformation
› http://www.edgehill.ac.uk/dc/phd/ontology#PresentWebBasedInformationLogic
› http://www.edgehill.ac.uk/dc/phd/ontology#MealMixerLogic
› http://www.edgehill.ac.uk/dc/phd/ontology#AmbientTemperature
› http://www.edgehill.ac.uk/dc/phd/ontology#Function
› http://www.edgehill.ac.uk/dc/phd/ontology#MealPlanLogic
› http://www.edgehill.ac.uk/dc/phd/ontology#Gps
› http://www.edgehill.ac.uk/dc/phd/ontology#ProximitySensor
› http://www.edgehill.ac.uk/dc/phd/ontology#ExerciseDiary
› http://www.edgehill.ac.uk/dc/phd/ontology#AndroidApiLevel19
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresHardware


› http://www.edgehill.ac.uk/dc/phd/ontology#hasMinimumApiRequirement
› http://www.edgehill.ac.uk/dc/phd/ontology#hasFunctionLogic
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresProximitySensor
› http://www.edgehill.ac.uk/dc/phd/ontology#hasMicrophone
› http://www.edgehill.ac.uk/dc/phd/ontology#hasTelephony
› http://www.edgehill.ac.uk/dc/phd/ontology#hasWifi
› http://www.edgehill.ac.uk/dc/phd/ontology#hasInfrared
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresRealitiveHumiditySensor
› http://www.edgehill.ac.uk/dc/phd/ontology#hasProximitySensor
› http://www.edgehill.ac.uk/dc/phd/ontology#hasStepDetector
› http://www.edgehill.ac.uk/dc/phd/ontology#hasHifi
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresAccelerometer
› http://www.edgehill.ac.uk/dc/phd/ontology#hasHardware
› http://www.edgehill.ac.uk/dc/phd/ontology#hasBarometer
› http://www.edgehill.ac.uk/dc/phd/ontology#hasCompass
› http://www.edgehill.ac.uk/dc/phd/ontology#hasBluetooth
› http://www.edgehill.ac.uk/dc/phd/ontology#hasGps
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresFingerprintReader
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresCompass
› http://www.edgehill.ac.uk/dc/phd/ontology#hasAudio
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresAmbientTemperature
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresLightSensor
› http://www.edgehill.ac.uk/dc/phd/ontology#hasAccelerometer
› http://www.edgehill.ac.uk/dc/phd/ontology#hasPersonalisedComponent
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresLoudspeaker
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresHifi
› http://www.edgehill.ac.uk/dc/phd/ontology#hasNfc
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresBluetooth
› http://www.edgehill.ac.uk/dc/phd/ontology#hasLightSensor
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresCamera
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresUsb
› http://www.edgehill.ac.uk/dc/phd/ontology#hasGyroscope
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresGyroscope
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresBarometer
› http://www.edgehill.ac.uk/dc/phd/ontology#hasUsb
› http://www.edgehill.ac.uk/dc/phd/ontology#hasHeartRateMonitor
› http://www.edgehill.ac.uk/dc/phd/ontology#hasStepCounter
› http://www.edgehill.ac.uk/dc/phd/ontology#hasFunctionType
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresTelephony
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresAudio
› http://www.edgehill.ac.uk/dc/phd/ontology#hasCdma
› http://www.edgehill.ac.uk/dc/phd/ontology#hasDisplay
› http://www.edgehill.ac.uk/dc/phd/ontology#hasGsm
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresLocation
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresStepCounter
› http://www.edgehill.ac.uk/dc/phd/ontology#hasLoudspeaker
› http://www.edgehill.ac.uk/dc/phd/ontology#hasCamera
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresWifi
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresInfrared
› http://www.edgehill.ac.uk/dc/phd/ontology#hasRealativeHumiditySensor
› http://www.edgehill.ac.uk/dc/phd/ontology#hasSensor
› http://www.edgehill.ac.uk/dc/phd/ontology#hasFrontCamera
› http://www.edgehill.ac.uk/dc/phd/ontology#hasAmbientTemperatureSensor
› http://www.edgehill.ac.uk/dc/phd/ontology#hasManufacturer
› http://www.edgehill.ac.uk/dc/phd/ontology#hasFingerPrintReader
› http://www.edgehill.ac.uk/dc/phd/ontology#hasLocation
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresStepDetector
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresNfc
› http://www.edgehill.ac.uk/dc/phd/ontology#hasRearCamera
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresSensor
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresHeartRateMonitor

Results for P11: Missing domain or range in properties.63 cases | Important

Object and/or datatype properties without domain or range (or none of them) are included in the ontology.

• This pitfall appears in the following elements:
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresHeartRateMonitor
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresSensor
› http://www.edgehill.ac.uk/dc/phd/ontology#hasRearCamera
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresNfc
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresStepDetector
› http://www.edgehill.ac.uk/dc/phd/ontology#hasLocation
› http://www.edgehill.ac.uk/dc/phd/ontology#hasFingerPrintReader
› http://www.edgehill.ac.uk/dc/phd/ontology#hasManufacturer
› http://www.edgehill.ac.uk/dc/phd/ontology#hasAmbientTemperatureSensor
› http://www.edgehill.ac.uk/dc/phd/ontology#hasFrontCamera
› http://www.edgehill.ac.uk/dc/phd/ontology#hasSensor
› http://www.edgehill.ac.uk/dc/phd/ontology#hasRealativeHumiditySensor
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresInfrared
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresWifi
› http://www.edgehill.ac.uk/dc/phd/ontology#hasCamera
› http://www.edgehill.ac.uk/dc/phd/ontology#hasLoudspeaker
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresStepCounter
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresLocation
› http://www.edgehill.ac.uk/dc/phd/ontology#hasGsm
› http://www.edgehill.ac.uk/dc/phd/ontology#hasDisplay
› http://www.edgehill.ac.uk/dc/phd/ontology#hasCdma
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresAudio
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresTelephony
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresMicrophone
› http://www.edgehill.ac.uk/dc/phd/ontology#hasFunctionType
› http://www.edgehill.ac.uk/dc/phd/ontology#hasStepCounter
› http://www.edgehill.ac.uk/dc/phd/ontology#hasHeartRateMonitor
› http://www.edgehill.ac.uk/dc/phd/ontology#hasUsb
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresBarometer
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresGyroscope
› http://www.edgehill.ac.uk/dc/phd/ontology#hasGyroscope
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresUsb
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresCamera
› http://www.edgehill.ac.uk/dc/phd/ontology#hasLightSensor
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresBluetooth
› http://www.edgehill.ac.uk/dc/phd/ontology#hasNfc
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresHifi
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresLoudspeaker
› http://www.edgehill.ac.uk/dc/phd/ontology#hasPersonalisedComponent
› http://www.edgehill.ac.uk/dc/phd/ontology#hasAccelerometer
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresLightSensor
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresAmbientTemperature
› http://www.edgehill.ac.uk/dc/phd/ontology#hasAudio
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresCompass
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresFingerprintReader
› http://www.edgehill.ac.uk/dc/phd/ontology#hasGps
› http://www.edgehill.ac.uk/dc/phd/ontology#hasBluetooth
› http://www.edgehill.ac.uk/dc/phd/ontology#hasCompass
› http://www.edgehill.ac.uk/dc/phd/ontology#hasBarometer
› http://www.edgehill.ac.uk/dc/phd/ontology#hasHardware
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresAccelerometer
› http://www.edgehill.ac.uk/dc/phd/ontology#hasHifi
› http://www.edgehill.ac.uk/dc/phd/ontology#hasStepDetector
› http://www.edgehill.ac.uk/dc/phd/ontology#hasProximitySensor
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresRealitiveHumiditySensor
› http://www.edgehill.ac.uk/dc/phd/ontology#hasInfrared
› http://www.edgehill.ac.uk/dc/phd/ontology#hasWifi
› http://www.edgehill.ac.uk/dc/phd/ontology#hasTelephony
› http://www.edgehill.ac.uk/dc/phd/ontology#hasMicrophone
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresProximitySensor
› http://www.edgehill.ac.uk/dc/phd/ontology#hasFunctionLogic
› http://www.edgehill.ac.uk/dc/phd/ontology#hasMinimumApiRequirement
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresHardware

Results for P13: Inverse relationships not explicitly declared.63 cases | Minor

This pitfall appears when any relationship (except for those that are defined as symmetric properties using owl:SymmetricProperty) does not have an inverse relationship (owl:inverseOf) defined within the ontology.

• This pitfall appears in the following elements:
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresHardware
› http://www.edgehill.ac.uk/dc/phd/ontology#hasMinimumApiRequirement
› http://www.edgehill.ac.uk/dc/phd/ontology#hasFunctionLogic
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresProximitySensor
› http://www.edgehill.ac.uk/dc/phd/ontology#hasMicrophone
› http://www.edgehill.ac.uk/dc/phd/ontology#hasTelephony
› http://www.edgehill.ac.uk/dc/phd/ontology#hasWifi
› http://www.edgehill.ac.uk/dc/phd/ontology#hasInfrared
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresRealitiveHumiditySensor
› http://www.edgehill.ac.uk/dc/phd/ontology#hasProximitySensor
› http://www.edgehill.ac.uk/dc/phd/ontology#hasStepDetector
› http://www.edgehill.ac.uk/dc/phd/ontology#hasHifi
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresAccelerometer
› http://www.edgehill.ac.uk/dc/phd/ontology#hasHardware
› http://www.edgehill.ac.uk/dc/phd/ontology#hasBarometer
› http://www.edgehill.ac.uk/dc/phd/ontology#hasCompass
› http://www.edgehill.ac.uk/dc/phd/ontology#hasBluetooth
› http://www.edgehill.ac.uk/dc/phd/ontology#hasGps
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresFingerprintReader
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresCompass
› http://www.edgehill.ac.uk/dc/phd/ontology#hasAudio
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresAmbientTemperature
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresLightSensor
› http://www.edgehill.ac.uk/dc/phd/ontology#hasAccelerometer
› http://www.edgehill.ac.uk/dc/phd/ontology#hasPersonalisedComponent
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresLoudspeaker
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresHifi
› http://www.edgehill.ac.uk/dc/phd/ontology#hasNfc
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresBluetooth
› http://www.edgehill.ac.uk/dc/phd/ontology#hasLightSensor
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresCamera
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresUsb
› http://www.edgehill.ac.uk/dc/phd/ontology#hasGyroscope
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresGyroscope
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresBarometer
› http://www.edgehill.ac.uk/dc/phd/ontology#hasUsb
› http://www.edgehill.ac.uk/dc/phd/ontology#hasHeartRateMonitor
› http://www.edgehill.ac.uk/dc/phd/ontology#hasStepCounter
› http://www.edgehill.ac.uk/dc/phd/ontology#hasFunctionType
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresMicrophone
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresTelephony
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresAudio
› http://www.edgehill.ac.uk/dc/phd/ontology#hasCdma
› http://www.edgehill.ac.uk/dc/phd/ontology#hasDisplay
› http://www.edgehill.ac.uk/dc/phd/ontology#hasGsm
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresLocation
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresStepCounter
› http://www.edgehill.ac.uk/dc/phd/ontology#hasLoudspeaker
› http://www.edgehill.ac.uk/dc/phd/ontology#hasCamera
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresWifi
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresInfrared
› http://www.edgehill.ac.uk/dc/phd/ontology#hasRealitiveHumiditySensor
› http://www.edgehill.ac.uk/dc/phd/ontology#hasSensor
› http://www.edgehill.ac.uk/dc/phd/ontology#hasFrontCamera
› http://www.edgehill.ac.uk/dc/phd/ontology#hasAmbientTemperatureSensor
› http://www.edgehill.ac.uk/dc/phd/ontology#hasManufacturer
› http://www.edgehill.ac.uk/dc/phd/ontology#hasFingerPrintReader
› http://www.edgehill.ac.uk/dc/phd/ontology#hasLocation
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresStepDetector
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresNfc
› http://www.edgehill.ac.uk/dc/phd/ontology#hasRearCamera
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresSensor
› http://www.edgehill.ac.uk/dc/phd/ontology#requiresHeartRateMonitor