

# External Software Reuse

*Software reuse by external developers in  
public good software ecosystems*

Terje Uglebakken



Thesis submitted for the degree of  
Informatics: Programming and System Architecture  
60 credits

Department of Informatics

Faculty of Mathematics and Natural Sciences

UNIVERSITY OF OSLO

Spring 2020



# **External Software Reuse**

*Software reuse by external developers in public  
good software ecosystems*

Terje Uglebakken

2020

© Terje Uglebakken 2020

External Software Reuse

<http://www.duo.uio.no/>

# Abstract

A software ecosystem consists of a shared software platform, where internal and external developers interact to deliver software to customers. Software ecosystems are reliant on external developers creating innovation for the software ecosystem to thrive. However, we lack knowledge about how to attract external developers and how to enable them to innovate. Research has shown that software reuse can increase productivity. This thesis explores how external developers reuse software in a public software ecosystem, where resources may be scarce and the aim is not revenue growth. The following research question is asked: *What are some factors that characterize software reuse by external developers in a public good software ecosystem?*

The Health Information Systems Programme (HISP) is a global network to strengthen health information systems. One of their most important contributions is District Health Information System 2 (DHIS2). Through the use of an Application Programming Interface (API), developers can extend DHIS2 by creating web applications. HISP India is one of the nodes in the HISP network which implements DHIS2.

In an Action Case research project, I have participated as a developer in the HISP India team to implement the AMR Surveillance System based on DHIS2. I have created software packages and other resources aimed at aiding the development of DHIS2 applications.

This thesis contributes to software ecosystem literature through factors that affect software reuse by external developers in public good software ecosystems. The research has also lead to practical contributions including DHIS2 applications and software packages to HISP India and the HISP network.

**Keywords:** *software reuse, software packages, software ecosystems, DHIS2*



# Acknowledgements

I want to thank my supervisor Sundeep Sahay and my co-supervisor Magnus Li for all their insight. I also want to thank everyone in the DHIS2 Design Lab. Special thanks to Elisabeth, Fredrik, Kristine, and Rebekka their help and companionship in India. I am eternally grateful to everyone at HISP India for their hospitality. Finally, I want to thank everyone in the HISP network who participated in the research.

Terje Uglebakken  
University of Oslo  
June 2020





# Table of Contents

- List of Figures ..... XI
- List of Tables.....XII
- Abbreviations ..... XIII
- 1 Introduction ..... 1
  - 1.1 Motivation ..... 1
  - 1.2 Research Question ..... 2
  - 1.3 Thesis Structure ..... 2
- 2 Background ..... 4
  - 2.1 HISP..... 4
  - 2.2 DHIS2..... 5
  - 2.3 India ..... 7
  - 2.4 AMR Surveillance System ..... 8
  - 2.5 Summary..... 8
- 3 Related Literature ..... 10
  - 3.1 Software Ecosystems ..... 10
    - 3.1.1 Actors in Software Ecosystems..... 10
  - 3.2 Software Reuse ..... 11
    - 3.2.1 Design Infrastructure..... 12
    - 3.2.2 Design Systems ..... 13
  - 3.3 Summary of Concepts ..... 14
- 4 Methodology ..... 17
  - 4.1 Epistemology ..... 17
  - 4.2 Research Design ..... 18
  - 4.3 Methods ..... 19
    - 4.3.1 Data Collection..... 20
    - 4.3.2 Analysis..... 24
  - 4.4 Research Ethics..... 24
    - 4.4.1 Data Protection ..... 24
    - 4.4.2 Research Outcomes ..... 25
- 5 Results ..... 26
  - 5.1 DHIS2 Software Ecosystem ..... 26

5.1.1	DHIS2 Apps .....	26
5.1.2	Customization in DHIS2 .....	28
5.2	DHIS2 App Design Infrastructure .....	28
5.2.1	Reusable Software .....	29
5.2.2	Software Reuse Tools.....	32
5.3	AMR Surveillance System .....	34
5.4	Software Reuse in HISP India .....	35
6	Analysis and Discussion.....	37
6.1	Factors Affecting External Software Reuse .....	37
6.1.1	Knowledge Dependencies .....	38
6.1.2	Software Reuse Tools.....	39
6.1.3	Internal to External.....	40
6.1.4	Self-reinforcement.....	41
6.2	App Development by External Developers .....	41
6.3	Reflections on the Research Process .....	42
6.3.1	Challenges .....	42
7	Conclusion.....	45
7.1	External Software Reuse Factors.....	45
7.2	Theoretical Implications and Future Research .....	46
7.3	Practical Implications .....	47
	References .....	48
	Appendix 1 AMR Surveillance System .....	56

# List of Figures

Figure 2-1: DHIS2 implanted as a HIS (from DHIS2, n.d.-b). Dark green = national. Medium green = Indian state. Light green = pilot. .... 5

Figure 2-2: The DHIS2 dashboard app (version 2.34.0 from DHIS2, n.d.-c)..... 6

Figure 3-1: Levels and types of design (from Li & Nielsen, 2019)..... 13

Figure 3-2: A region built from a series of components (from Vesselov & Davis, 2019, p.15) ..... 14

Figure 3-3: Interaction between actors through artifacts in software ecosystems ..... 16

Figure 4-1: Field trips to India and important events in 2019..... 18

Figure 4-2: Action case as a hybrid methodology (from Braa & Vidgen, 1999)..... 19

Figure 4-3: Field visit to a hospital using DHIS2 apps ..... 21

Figure 4-4: The DHIS2 Application Platform Workshop..... 22

Figure 4-5: Group discussion during the Design Lab Workshop..... 23

Figure 5-1: Parts which make up DHIS2 apps (from McGee, 2019)..... 27

Figure 5-2: The DHIS2 header bar..... 29

Figure 5-3: A portion from the instructions to use a button in the DHIS2 Design System ..... 33

Figure 6-1: Air Quality Index during November 2019 ..... 43

# List of Tables

Table 3-1 Concepts related to software reuse by external developers in software ecosystems	16
Table 6-1: Themes and key findings .....	38
Table 7-1: Factors of software reuse by external developers in a public good software ecosystem .....	46

# Abbreviations

AC	Action Case
AMR	Antimicrobial resistance
AMRSS	AMR Surveillance System
API	Application Programming Interface
AQI	Air Quality Index
AR	Action Research
DHIS2	District Health Information Software 2
FOSS	Free and Open-Source
HIS	Health Information System
HISP	Health Information Systems Program
ICMR	Indian Council of Medical Research
IS	Information System
NORAD	Norwegian Agency for Development Cooperation
NSD	Norwegian Centre for Research Data
SE	Software Engineering
UI	User interface
UiO	University of Oslo

# 1 Introduction

Software platforms such as Google's Android, Apple's iOS, and Microsoft's Windows have become juggernauts. The individual software projects built for a platform are not developed in isolation. Software is shared and reused, forming software ecosystems (Bosch & Bosch-Sijtsema, 2010; Jansen, Finkelstein, & Brinkkemper, 2009; Manikas & Hansen, 2013). Bosch & Bosch-Sijtsema (2010) define software ecosystems as "a software platform, a set of internal and external developers and a community of domain experts in service to a community of users that compose relevant solution elements to satisfy their needs" (p. 68). The platforms mentioned above all succeeded in enabling innovation from external developers for their platforms.

## 1.1 Motivation

It can be difficult for central actors such as platform owners or internal developers to anticipate the needs of a heterogeneous user-base (Bosch, 2009). Software ecosystems are reliant on external developers providing innovations for the platform to thrive (Bosch, 2009). External developers may also have better opportunities to utilize user participation, which is linked to higher-quality software (Bosch, 2009). Although important, the participation of these actors has received little research (van den Berk, Jansen, & Luinenburg, 2010, p. 134). The focus of this thesis is to explore software reuse by external developers in public good software ecosystems, in contexts where resources are limited.

The empirical case of the research is District Health Information Software 2 (DHIS2). DHIS2 is a health management information system platform. By utilizing its Application Programming Interface (API), actors can develop web applications, hereby referred to as *apps*, that can be installed in the system. DHIS2 is developed by the Health Information Systems Program (HISP) at the University of Oslo (UiO). In recent years HISP UiO has put increasingly more effort into developing reusable software and other related resources to encourage external developers to create more apps for the DHIS2 software ecosystem. The HISP network includes other nodes located in developing countries that implement DHIS2 and develop DHIS2 applications.

Through three field visits, I have participated and observed in HISP India, one of the oldest nodes in the HISP network. During the fieldwork with a combined length of 15 weeks, as well as remotely, I have been part of their team as a developer in one of their projects implementing

DHIS2. Also, I have helped HISP UiO develop reusable software and introduce them to HISP India. Finally, I have been working as a teacher's assistant in UiO master's level course where students develop DHIS2 applications.

## 1.2 Research Question

To address the gap in the literature as well as the practical problem in the HISP project the following research question was formed: *What are some factors that characterize software reuse by external developers in a public good software ecosystem?*

*Software reuse* refers to the use of software which is not created to be used directly by end-users, but rather to be used by developers to create new software. Thus, software reuse does not include modifying the source code of an existing software application in the context of this thesis. *External developers* refer to software developers that are disconnected from where the shared software platform is developed in the software ecosystem. *Public goods* are described by Sahay (2019) as “those whose benefits cannot be confined to a single or a set of buyers, for example, street names and a clean environment” (p. 1). This limits the research question to software ecosystems which are not based on a goal of generating revenue.

I explore the research question by 1) investigating the DHIS2 software ecosystem and software reuse by external developers in India, 2) by participating in the development of new software packages for the DHIS2 software ecosystem and introducing these to external developers in India.

By answering the research question, I hope to contribute to the software reuse literature, as well as to software ecosystem literature which can be found in the field of both Information Systems (IS) and Software Engineering (SE). Practically, it could help platform owners to design reusable software and related tools for enabling innovation in software ecosystems. Finally, I hope the results may aid the HISP project to empower the global south to develop innovative solutions.

## 1.3 Thesis Structure

### Chapter 2 - Background

Information about the context surrounding the empirical case of the research. This includes the HISP network, the DHIS2 platform, a HISP India project I participated in, and India.

### **Chapter 3 - Related Literature**

Fundamental theoretical concepts within software ecosystems. Concepts surrounding software reuse are introduced including design infrastructure and design systems. The theoretical lens is established.

### **Chapter 4 - Methodology**

Describes how the research was conducted including the epistemological basis, research design, and methods used for data collection and analysis. Concludes with ethical remarks.

### **Chapter 5 - Results**

Presents data about the DHIS2 software ecosystem, reusable software and software reuse tools for DHIS2 app development. Experiences from HISP India project implementing DHIS2 and software reuse practices in HISP India is also described.

### **Chapter 6 - Analysis & Discussion**

Analysis and discussion based on the results. Factors which characterize software reuse by external developers in public good ecosystems are presented.

### **Chapter 7 - Conclusion**

The research is summarized, and the research question is answered. Reflects on theoretical and practical implications.



## 2 Background

The context surrounding the software ecosystem behind the empirical case the research project is described in this chapter. First, with the history and the current form of the HISP network with a focus on HISP India and HISP UiO, where most of the data was gathered. Then the DHIS2 platform is outlined. The chapter also includes background information on India and HISP India projected I participated in.

### 2.1 HISP

The Health Information System Program (HISP) started in postapartheid South Africa during the 1990s (Braa & Hedberg, 2002). It was founded by activists in the antiapartheid movement. The goal was to develop a Health Information System (HIS) supporting the district level of the health sector.

District Health Information Software (DHIS) was developed and implemented in South Africa in 1998 (Braa, Monteiro, & Sahay, 2004). After spreading to other countries in the global south, an international version of DHIS was released (Titlestad, Staring, & Braa, 2009). While the software itself was open source, it used the proprietary database system Microsoft (MS) Access. Also, users needed to have MS Windows and MS Office. In 2004 the development of DHIS2 started at the University of Oslo (UiO) (Braa & Sahay, 2017).

HISP is a network consisting of ten HISP nodes in the global south, along with HISP UiO located in Oslo (Nielsen, 2019). Members in the HISP nodes include health professionals, software developers, students, researchers, and others. Some HISP nodes are linked directly to universities, while others are independent organizations. An international PhD program and several master's programs in developing countries have been funded through the HISP network (Titlestad et al., 2009). Additionally, several courses at the Department of Informatics of UiO involve HISP in some way.

The development of DHIS2 is funded by donors such as the Norwegian Agency for Development Cooperation (Norad), PEPFAR, and the Global Fund. DHIS2 is not used by HISP UiO for revenue growth, but rather as a public good. The other HISP nodes do not receive funding through the funding of the DHIS2 development and are financially independent of

HISP UiO. They are largely sustained through contracts with governmental and non-governmental organizations.

HISP India was involved when DHIS2 was first piloted in India in 2006 (Titlestad et al., 2009). It is a non-profit organization with the main office in a city right outside New Delhi. HISP India has around 30 employees. Two of their six developers can be considered senior developers with many years of experience in the organization. The majority of the other employees are health information experts. The rest include administrative workers and system administrators.

## 2.2 DHIS2

HISP considers DHIS2 a public good, derived from the aim “to support local management of health care delivery and information flows in selected health facilities, districts, and provinces, and its further spread within and across developing countries.” DHIS2 was originally designed as a HIS to report and analyze aggregated health data. It has since then evolved to track patient-wise data. DHIS2 is mostly implemented in the global south, where resources can be scarce (Msiska & Nielsen, 2018, p. 399). Figure 2-1 displays where DHIS2 is implemented as a HIS.

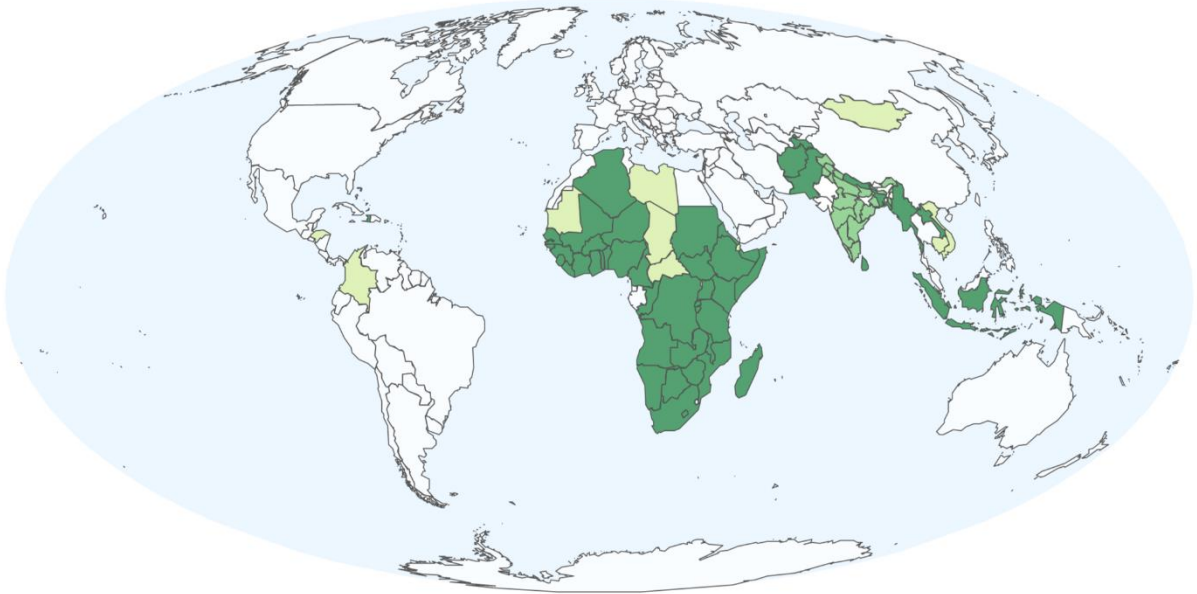


Figure 2-1: DHIS2 implanted as a HIS (from DHIS2, n.d.-b). Dark green = national. Medium green = Indian state. Light green = pilot.

DHIS2 is a generic system, in that it can be used for many purposes. Lately, it has even been used outside the health domain, in the education sector in Gambia and Uganda. DHIS2 cannot

be used out-of-box. Implementers must configure its metadata. Implementing DHIS2 does not necessarily involve any software development. It is not uncommon for implementers to use DHIS2 exclusively by exclusively configuring metadata. This implementation approach has a low cost associated with it, due to not needing software developers. For some use cases, there have also been released preconfigured metadata packages. HISP has released the COVID-19 Surveillance Digital Data Package (DHIS2, n.d.-a) and the World Health Organization has released configuration packages for several use-cases (World Health Organization, n.d.).

The DHIS2 platform has a modular architecture. The backend of DHIS2 is written in Java. It supports multiple database systems, although only use with PostgreSQL is officially documented. The frontend consists of several apps. These communicate with the backend through an API using the REST architectural style. The API is a module which allows systems to access and manipulate data in the backend. HISP UiO maintains the DHIS platform and a series of bundled apps, which are preinstalled in DHIS2. For this reason, the bundled apps can be considered to be part of the platform itself. A screenshot of the Dashboard app, a bundled apps which acts as the default “home” app can be seen in figure 2-2. Major versions of DHIS2 are released about two times a year.

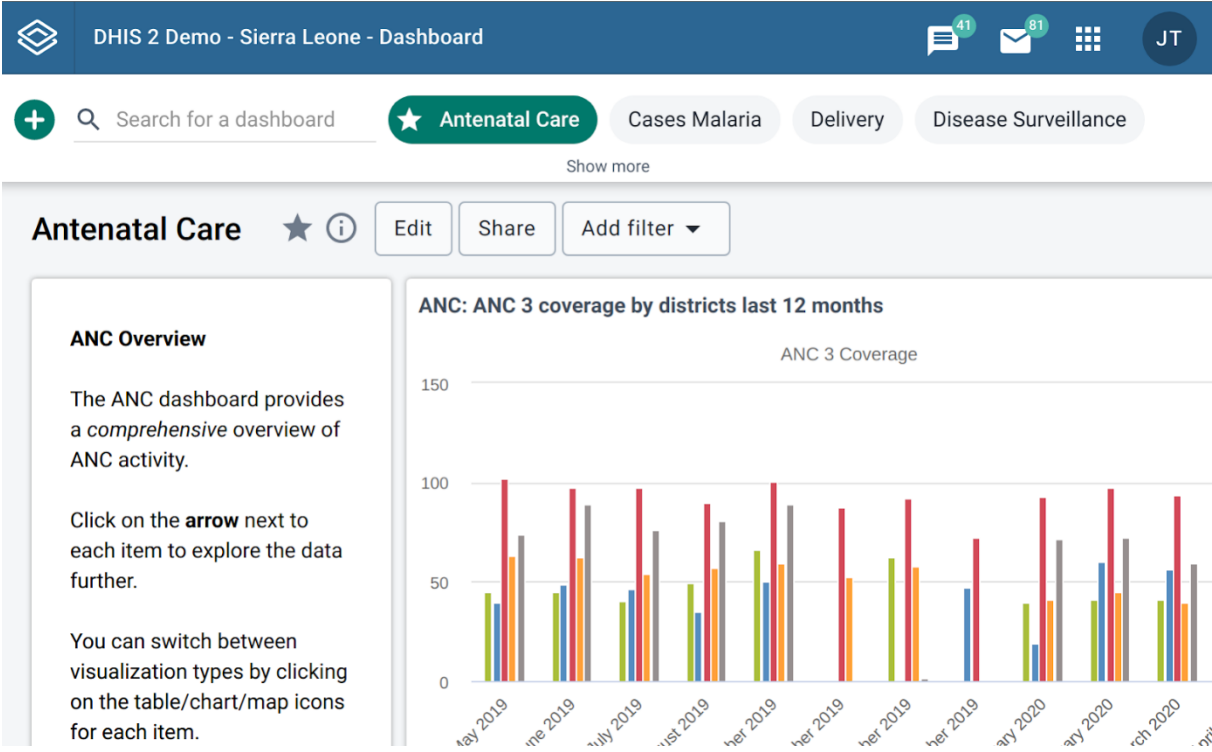


Figure 2-2: The DHIS2 dashboard app (version 2.34.0 from DHIS2, n.d.-c)

DHIS2 is Free and Open-Source (FOSS). It is licensed with the permissive BSD 3-Clause license, which has minimal restrictions on the use and distribution of the software. This license allows anyone to access, alter, republish, and profit monetarily from the source code. The bundled apps and other software maintained by HISP UiO are typically also permissively licensed. This allows implementers of DHIS2 to modify and replace the bundled apps.

The DHIS2 API is can be accessed through network requests, allowing external developers to create their apps, which can be installed in the system. There is also the DHIS2 App Hub, where third-party apps can be published, allowing others to install them in their DHIS2 instance.

Several Android applications and resources for creating them also exist for DHIS2. However, the DHIS2 Android software ecosystem is not covered in this thesis.

## **2.3 India**

The Republic of India became an independent nation in 1947. The country split into India and Pakistan, while Bangladesh later became independent of Pakistan. Territorial disputes have since then led to several wars with Pakistan. India shares its borders with Pakistan, China, Nepal, Bhutan, Bangladesh, and Myanmar. There are several hundred languages spoken in India. Each state in India has one or more official languages. Hindi is spoken by the majority of the population, while English is often used for business in some areas. The largest religion in India is by far Hinduism, but Islam also has a substantial presence. Christianity, Sikhism, Buddhism, Jainism, and other religions are present.

As of 2019, India is the second-most populous country in the world (United Nations, 2019, p. 12). It currently has about 1,37 inhabitants (18 % of the global total) and has been projected to surpass China as the world's most populous country around the year 2027 (United Nations, 2019, p. 12). India has in recent years had one of the fastest-growing economies in the world (International Monetary Fund, 2019, p. 5). However, the economic growth has slowed down, unemployment is high, and labour force participation has decreased (International Monetary Fund, 2019, p. 5).

In the 1980's India softened its restrictive IT policies, leading to better availability of computers in the country (Subramanian, 2006, p. 38). Moreover, the new policies helped in slowing down the "brain drain" (Subramanian, 2006, p. 38) or "body-shopping", where India lost many IT

experts to the United States and Europe (Nicholson, Sahay, & Heeks, 2018, p. 532). Since then, the IT and software development sectors have been among the fastest-growing sectors in India (D’Mello, 2005, p. 7; Garg & Varma, 2008). However, the authors state:

*Recent trends and studies exhibit that the lack of adequately trained professionals will be a major roadblock in sustenance and further growth of this industry in India. This lacuna is a direct result of poor Software Engineering (SE) education and training infrastructure in the country. (Garg & Varma, 2008, p. 110)*

## **2.4 AMR Surveillance System**

Antimicrobial resistance (AMR) is the ability of microorganisms to develop partial or complete immunity to antimicrobial drugs. It is among the recently published thirteen health challenges for the next decade by the World Health Organization (2020, January 13). A review commissioned from the United Kingdom government from 2014 estimates that AMR could cause 10 million deaths by 2050 if no preventative action is taken (O’Neill, 2014).

In 2013 the Indian Council of Medical Research (ICMR) launched the Antimicrobial Resistance Surveillance and Research Network for the “purpose of strengthening surveillance and rationalizing exploratory research” (Indian Council of Medical Research, n.d.-a). ICMR developed a web-based system to enter and analyze AMR data. ICMR was happy about the system and considered it “user-friendly” (Indian Council of Medical Research, n.d.-b).

However, ICMR had ambitions for the system to be used across all of India and internationally. In 2018 ICMR and HISP India started a project to develop a new system, based on the DHIS2 platform, to replace ICMR’s existing system. The decision to migrate to a DHIS2 based system was motivated by arguments that the platform architecture of DHIS2 would make further development of the system more sustainable. This system became known as the AMR Surveillance System (AMRSS).

## **2.5 Summary**

HISP is a global network of researchers and practitioners where the goal is to strengthen Health Information Systems. The main output of the network is the DHIS2 platform. It is a public good, used mainly in the global south. DHIS2 has a modular architecture and its API supports

the development of apps. India is a developing country with a large and growing software sector. AMRSS is a HISP India project based on the DHIS2 platform.

## 3 Related Literature

This thesis aims to explore the role of software reuse in enabling innovation by external developers in public good software ecosystems. To do so, it is necessary to clarify exactly what is meant by software ecosystems. Concepts related to software reuse in software ecosystems are also introduced, including design infrastructure, design systems, and software packages. The chapter ends with a summary of the concepts used as the theoretical lens for the research.

### 3.1 Software Ecosystems

In the context of this thesis, software ecosystems are collections of software that evolve together through a common software platform. Specifically, software ecosystems consist of “a software platform, a set of internal and external developers and a community of domain experts in service to a community of users that compose relevant solution elements to satisfy their needs” (Bosch & Bosch-Sijtsema, 2010, p. 68).

A typical reason for a product to evolve into a software ecosystem is that the amount of functionality that the users need is not feasible to develop without contribution from external developers (Bosch, 2009, p. 111; van den Berk et al., 2010, p. 127). Similarly, the mass customization trend where single users or groups of users want something specifically tailored to them (Bosch, 2009, p. 111). Again, with a large heterogeneous user group, it is difficult for an organization to deliver products on this scale. External developers contribute to the ecosystem by developing extensions, or *apps*. However, the apps which are useful to large amounts of users can be absorbed into the platform (Bosch, 2009, p. 111). Thus, external developers can also contribute indirectly to the platform itself, by apps being platformized (Bosch, 2009, p. 111).

#### 3.1.1 Actors in Software Ecosystems

Actors and their interaction are an important part of software ecosystems (Manikas & Hansen, 2013, p. 1300). The orchestrator, the niche player, and the customer are three common actors referred to in the software ecosystem literature (Manikas & Hansen, 2013, p. 1301).

The orchestrator runs the platform of the ecosystem and manages the ecosystem. Others often refer to the orchestrator as the platform owner. Orchestrators can develop strategies to keep the

software ecosystem health by maximizing the profitability of other actors in the ecosystem (Jansen et al., 2009, p. 187). As an example, how should the platform be open to external actors? Strategies for opening up a platform include using a permissive open source licensing and by open APIs (Alsbaugh, et al., 2009; Jansen et al., 2009, p. 188).

External developers, which this thesis uses as the term for niche players, add new apps to the platform. The apps are an important part of the software ecosystem as they have a major influence on the platform's success. Policies and strategies from the orchestrator enable and constrain niche players. Apple enforces a strict approval policy for apps entering the iOS App Store to assure quality. "Choosing the right balance between quality and innovation is vital for ecosystem health" (van den Berk et al., 2010, p. 130). Software ecosystems both need to attract external developers and maintain the quality of the software created by external developers.

The customer is the actor who acquires and uses the apps in the software ecosystem. This can be a person or an organization.

## **3.2 Software Reuse**

The architecture of a software platform, which can include reusable software, has a role for the orchestrator's management of external developers in a software ecosystem (van den Berk et al., 2010). This section presents the literature used for software reuse. Specifically, software "designed to be reused and to provide functionality to other software projects" (Haefliger, von Krogh, & Spaeth, 2008, p. 180), in contrast to software which can be used by end-users directly. Also, two highly relevant concepts related to software reuse for the empirical case are introduced; design infrastructure and design systems.

It has been argued that reusing software can improve productivity (Banker & Kauffman, 1991) and reduce defects (Basili, Briand, & Melo, 1996; Lim, 1994; Mohagheghi, Conradi, Killi, & Schwarz, 2004), even to the extent of reducing the cost and time to market by a factor of ten or more (Schmid & Verlage, 2002, p. 50).

However, the act of developing and maintaining reusable software is not free (Lim, 1994), and is an investment that may or may not be successful (Barns & Bollinger, 1991). There are also possible negative aspects of reusing software. Some risks include "security vulnerabilities, license violations, and breaking changes" (Eghan, Alqahtani, Forbes, & Rilling, 2019, p. 1109).



The most famous example of software reuse gone wrong is perhaps the Heartbleed vulnerability in OpenSSL (Carvalho, DeMott, Ford, & Wheeler, 2014). Another example is the “leftpad incident” which caused issues for many popular services such as Facebook and Netflix (Hejderup, van Deursen, & Gousios, 2018, p. 101). The incident was caused by the removal of a version of a software package, which consisted of just three lines of code. Many developers reuse trivial software packages, believing they are well tested, despite it often not being the case (Abdalkareem, Oda, Mujahid, & Shihab, 2020).

Barns & Bollinger (1991) view reusable software as a subset of reusable work products. As examples of reusable work products, the article mentions requirements specifications, designs, code modules, documentation, test data, and customized tools (p. 14). Fischer (1987) argues that software reuse tools are necessary for developers to understand software components and how to use them. Fischer (1987, p. 71) states: “Knowing about the existence of components is not trivial, especially as the number of components grows. And if you do find a potentially useful component, you must determine how it must be used and combined with the other components. You must understand its functionality and its properties [...]” Lim (1994, p. 27) points to a lack of tools for reuse to cause decreased productivity when reusing work.

Software packages are reusable software which published, making it available for other actors. One way of distributing software packages is through a package manager. Many programming languages have a package manager. For instance, npm for JavaScript and PyPI for Python. These package managers can be considered platforms which form software ecosystems. While package managers may make software packages easily available, selecting the most appropriate software package is not a trivial task (Mili, H., Mili, F., & Mili, A., 1995; de la Mora & Nadi, 2018).

### **3.2.1 Design Infrastructure**

Li & Nielsen (2019) introduced a framework to analyze generic software from a design perspective. They argue that design often happens at two levels. First, generic-level design, which is “the design process unfolding during the development of the generic software product”. Secondly, implementation-level design, which unfolds “during the implementation of the generic software product”. Design infrastructure is the technical and social resources that can be used to customize the product during implementation-level design. Examples of these resources include APIs, documentation, software packages, and training.

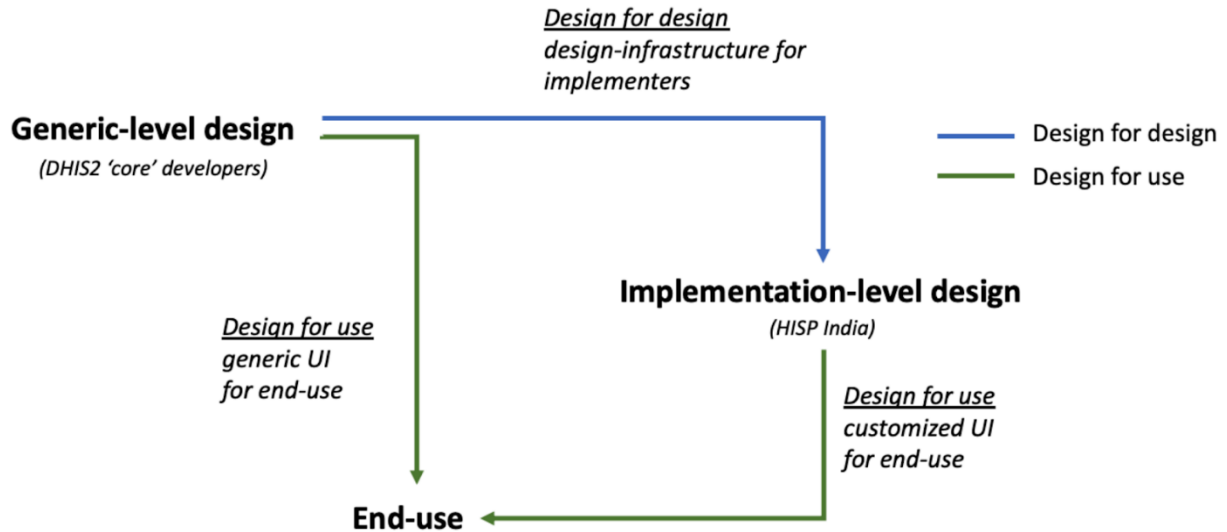


Figure 3-1: Levels and types of design (from Li & Nielsen, 2019)

Li & Nielsen (2019) argues that designers at the generic level should focus on the meta-usability of the product. That is, to provide a good design infrastructure for the implementers of the product. Creating extensions of the software in the form of apps may be one solution to mitigate the usability problems we see in generic software (Li, 2019a; Li, 2019b). However, this option has its own set own challenges such as requiring programming competence and maintaining the software (Li, forthcoming). The difference in terms of resources needed between implementing software with configuration alone and by customization with code can be large, a transition sometimes referred to as the customization cliff (Czarnecki et al., 2006). Design systems have been suggested as an example of a tool that may lower the resources needed to develop apps and to help strike a balance between generic functionality and usability (Li, 2019a; Li, 2019b).

### 3.2.2 Design Systems

Vesselov & Davis (2019) describe design systems as “a series of documented elements, components, and regions that include both design and front-end guidelines” (p. 16). Figure 3-2 as a whole, is an example of such a region. The region contains multiple components, like buttons. The buttons, in turn, contains labels and borders, or elements. This way of breaking down a user interface (UI) into smaller building blocks is similar to atomic design as introduced by Frost (2016). In atomic design, these building blocks are described as *atoms*, *molecules*, and *organisms*.



Figure 3-2: A region built from a series of components (from Vesselov & Davis, 2019, p.15)

*Component libraries*, which is the term this thesis uses to refer to software packages of UI modules of varying sizes, are perhaps the most important product of a design system. Component libraries are extremely popular. As of January 2020, the most popular of the myriad of component libraries that exist, Bootstrap, is the sixth most forked and starred repository in GitHub. Frost (2016) points to development speed and design consistency being two of the most important perceived benefits. On the other hand, different products may look too generic if they use the same components, the size of the package may lead to worse performance, and the time spent modifying or customizing the components may outweigh the gains you get from using the component library in the first place. Vesselov and Davis (2019, pp. 25-50) also look to speed and consistency. Used as an example, you can make changes across multiple pages or products when you make a change to a single component. This is the essence of the DRY (Don't Repeat Yourself) principle (Hunt & Thomas, 2000).

Apart from the component library, a design system typically includes what is often referred to as a style guide (Vesselov & Davis, 2019, pp. 15-16). While a component library itself should include documentation of how to use the components and how to create new ones, a style guide documents design principles. These principles are guidelines for how the products should work and look. For example, you may define which colours you should use or how to communicate with the user.

Both Frost (2016) and Vesselov & Davis (2019, pp. 25-50) stresses the importance of getting buy-in from all the important actors. Frost (2016) says that even though third parties have limited influence on design systems due to being outside the organization, they are in a good position to provide a perspective that is hard for insiders to have.

### 3.3 Summary of Concepts

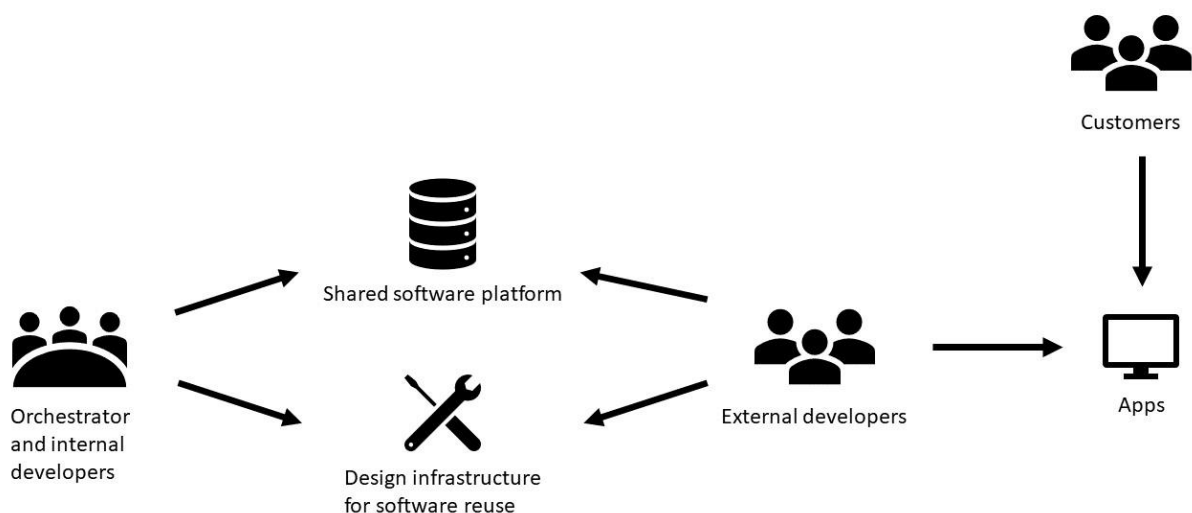
This section summarizes the central concepts used as the theoretical lens for this thesis. Table 3-1 describes concepts related to software reuse by external developers in software ecosystems.

Software ecosystem	Internal and external developers creating software based on a shared software platform for customers
Shared software platform	The shared software system apps are based on
Orchestrator	The actor in the software ecosystem which manages the software ecosystem
Internal developers	The actors that develop the shared software platform. The orchestrator often includes internal developers.
External developers	The actors that develop apps for the software ecosystem. The orchestrator does not include external developers.
Customers	The actors that acquire or uses apps in the software ecosystem.
Software reuse	Software that is designed to be reused to develop new software, rather than designed for end-use
Software package	Reusable software that is published
Software reuse tools	Resources that help developers understand and use reusable software
Design infrastructure	The technical and social resources that are used to implement software
Design system	A documented UI component library and a style guide

Atomic principles	Reusing smaller UI components to create larger UI components. The UI components are organized into atoms, molecules, and organisms.
UI component library	Reusable software, often a software package, that includes UI modules of varying sizes
Style guide	Documentation of design principles for how software should look and work

*Table 3-1 Concepts related to software reuse by external developers in software ecosystems*

Figure 3-3 visualizes the interaction between actors through artifacts in a software ecosystem. The orchestrator manages the software ecosystem. They also maintain the shared software platform through their internal developers. Internal developers also manage the design infrastructure for software reuse. That is, the reusable software and the tools that help developers understand and use it. The external developers develop apps using the software platform and the design infrastructure for software reuse. Customers acquire and use the apps.



*Figure 3-3: Interaction between actors through artifacts in software ecosystems*

# 4 Methodology

This research project aims to explore the role of resources in the fringes of software ecosystems. DHIS2, HISP India, and wider HISP network served as the empirical case for the research. I researched this by investigating the DHIS2 software ecosystem, and by creating and introducing resources to HISP India. This chapter details the methodology behind how this was done. The chapter begins by clarifying the epistemological basis of the research. Then, how the research was designed and how it evolved. Later, methods used during data collection and analysis are described. The chapter concludes with some ethical remarks.

## 4.1 Epistemology

Clearly defining one's philosophical assumptions when doing research is crucial, as it "affects every aspect of the research process, from how the evidence is collected to how the results are interpreted" (Atkins & Sampson, 2002, p. 102). Despite being important, it is often neglected (Checkland & Holwell, 1998, pp. 14, 17). Similarly, "we are all biased by our own background, knowledge and prejudices to see things in certain ways and not others" (Walsham, 2006, p. 321). For these reasons, I will discuss the epistemological stance this research project is based on. In other words, the assumptions of what knowledge is, and how to obtain it (Hirschheim, 1985, p. 10).

This research project is based on interpretive metaphysical assumptions. Interpretivism is a view where knowledge is socially constructed by human actors (Walsham, 1995, p. 376). Both researchers and subjects are affected by their subjective preconceptions. As a consequence, objective data cannot be collected. Knowledge is generated by understanding social interaction. Thus, to generate knowledge "researchers need to engage in the social setting investigated and learn how the interaction takes place from the participants' perspective" (Chen & Hirschheim, 2004, p. 201).

IS research has since the 90's been more accepting of interpretive literature, after a long dominance of positivism (Chen & Hirschheim, 2004; Walsham, 1995). A notable sign of this change was when MIS Quarterly call for papers based on interpretive methods in 1993 (DeSanctis, 1993, p. vii). Hirschheim (1985, p. 10, 32) argues that because IS are socio-technical systems, the IS literature benefits from epistemological pluralism. Interpretivism and studying

systems development in a socio-technical manner (Bansler, 1988) speaks to me as a researcher on a personal level.

## 4.2 Research Design

The aim of the research began broadly, by studying how HISP India implemented DHIS2 to fit local needs. The research design evolved through an iterative approach, which is common for qualitative research (Greenhalgh & Taylor, 1997, p. 740). The first field trip to India, lasting four weeks, was used to identify problems HISP India faced developing DHIS2 apps. The eight months between the first and second field trip was used to narrow down the research theme, formulate a research question, and plan for how data was to be collected and analyzed. An overview of the field trips to India and important events in 2019 is visualized in figure 4-1.

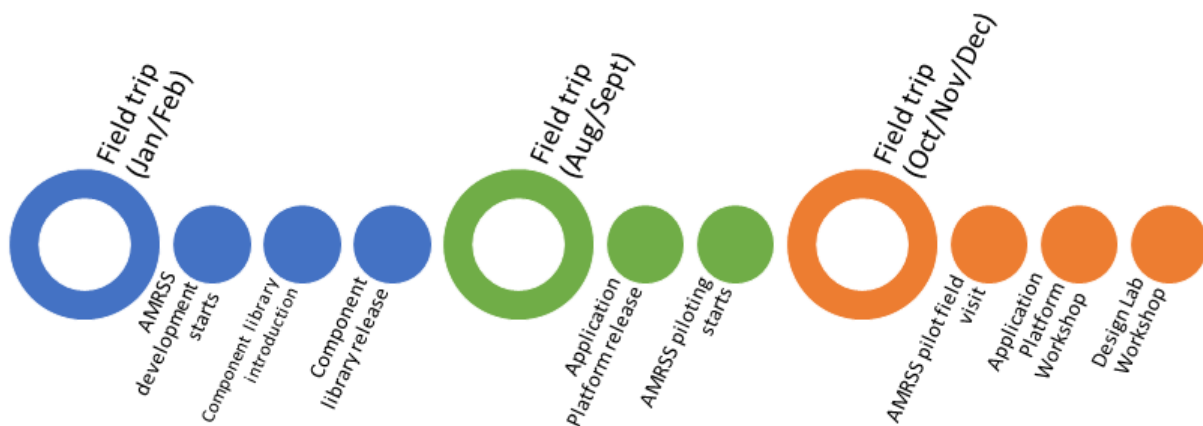


Figure 4-1: Field trips to India and important events in 2019

Prior to the first field trip, I had already agreed to participate to implement in one of HISP India’s projects. This made me a highly involved researcher. I also argue that interventions, where resources were introduced to the organization, was a better approach to answering the research question, rather than more passive methodologies. Thus, Action Research (AR) was chosen as the methodology. I was also inherently biased towards following AR by being part of the overarching HISP project, where many research projects are based on AR. The HISP project in itself can be viewed as a single AR project (Braa et al., 2004). Still, why would an inexperienced researcher attempt to use a methodology in an untested field? In addition to being appropriate for the research question, I also valued that AR has acceptance in several contexts relevant to my case. AR is widely used in the IS literature. Baskerville & Wood-Harper argues that “IS seems to be a very appropriate field for the use of action research methods” (1998, p.

90). Also, AR can be based on interpretivism (Goldkuhl, 2012b, p. 142; Klein & Myers, 1999, p. 69; Walsham, 2006, p. 321).

AR is practised in many different ways in IS (Baskerville & Wood-Harper, 1998; Elden & Chisholm, 1993). Therefore, it is crucial to specify how the AR methodology is used. This research was designed with the principles for canonical AR in mind (Davison, Martinsons, & Kock, 2004). The research methodology later evolved to be more similar to that of Action Case (AC) (Vidgen & Braa, 1997; Braa & Vidgen, 1999). It should be noted that Vidgen & Braa (1997) positions AC as a hybrid methodology based on AR and a case study, not as a type of AR. This is illustrated in figure 4-2. However, Goldkuhl (2012a) argues that “AR endeavours are intervention oriented cases studies” (p. 64) in reference to AC as an example of AR. Baskerville & Wood-Harper (1998) notes that “a substantial portion of the action research published in the IS field may have been mischaracterized as case research” (p. 105) while arguing for wider acceptance of variety within AR.

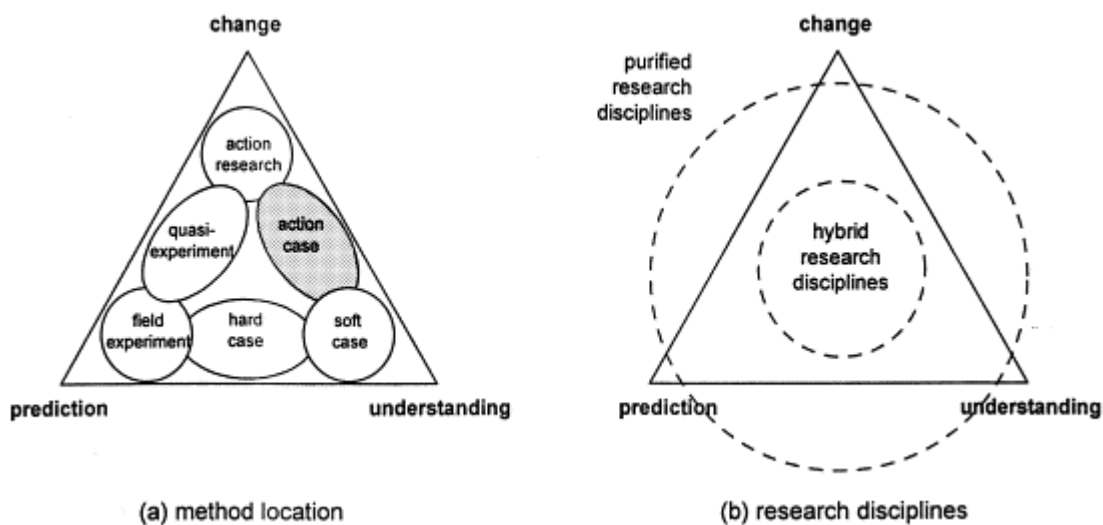


Figure 4-2: Action case as a hybrid methodology (from Braa & Vidgen, 1999)

The cause of what led to the methodology evolving to be more in line with AC was the postponement of the development of an app. Interventions were planned to involve a HISP India developer during the creation of this app. Due to this, the research project ended up comprising of interventions of smaller scale and to the intervention phases being shorter. Both are factors that differentiate AC from AR (Vidgen & Braa, 1997, p. 538).

### 4.3 Methods



This research project utilized methodological triangulation (Thurmond, 2001), where multiple methods were used for data collection and analysis. Also, data collection was done on multiple sites. Otherwise known as data triangulation (Thurmond, 2001). Triangulation types were used to increase the validity of the findings. While the main group of interest was the HISP India developer, I argue that the inclusion of HISP UiO developers and university students enriched the research. The university students added perspective to what developers with limited knowledge about certain technologies could produce. In contrast, the HISP UiO team provided data from an expert's point of view.

This section is divided into methods used during data collection and analysis. It should be noted that this research project cannot be stringently divided into phases of data collection and analysis. Instead, these phases intersected. This is also true for the presented methods. I would, for example, say that the interviews involved analysis.

### **4.3.1 Data Collection**

#### **Participant Observation**

Most of the time during the field trips were spent working together with HISP India employees involved with the AMRSS project at their main office. This involved writing code, attending meetings, analyzing existing solutions, gathering requirements, and collecting feedback. My involvement also continued remotely in-between field trips. Despite noticeable cultural differences, they were welcoming to the extent that I felt like I was part of their team. I joined HISP members to visits in the field relevant to the AMRSS project. A picture taken from a hospital piloting AMRSS can be seen in figure 4-3.



*Figure 4-3: Field visit to a hospital using DHIS2 apps*

I also made contributions to the globally used DHIS2 related software packages in the form of code and by creating issues. This gave me insight into how these could be used and how outside parties could make contributions. It also helped mature the software packages to be used in the UiO course and HISP India.

## **Interviews**

Countless unstructured interviews were held at the HISP India office. Some of these resembled “friendly conversation with no predetermined focus” (Crang & Cook, 2007, p. 60). Other times the interviewees were approached with the goal of data collection. These interviews were mostly held where others could listen in at, for instance at the lunch table. Additionally, unstructured interviews were also held at the HISP office in Oslo where the participants were developers from HISP India visiting developers from other HISP nodes, researchers, and other people involved in the HISP project.

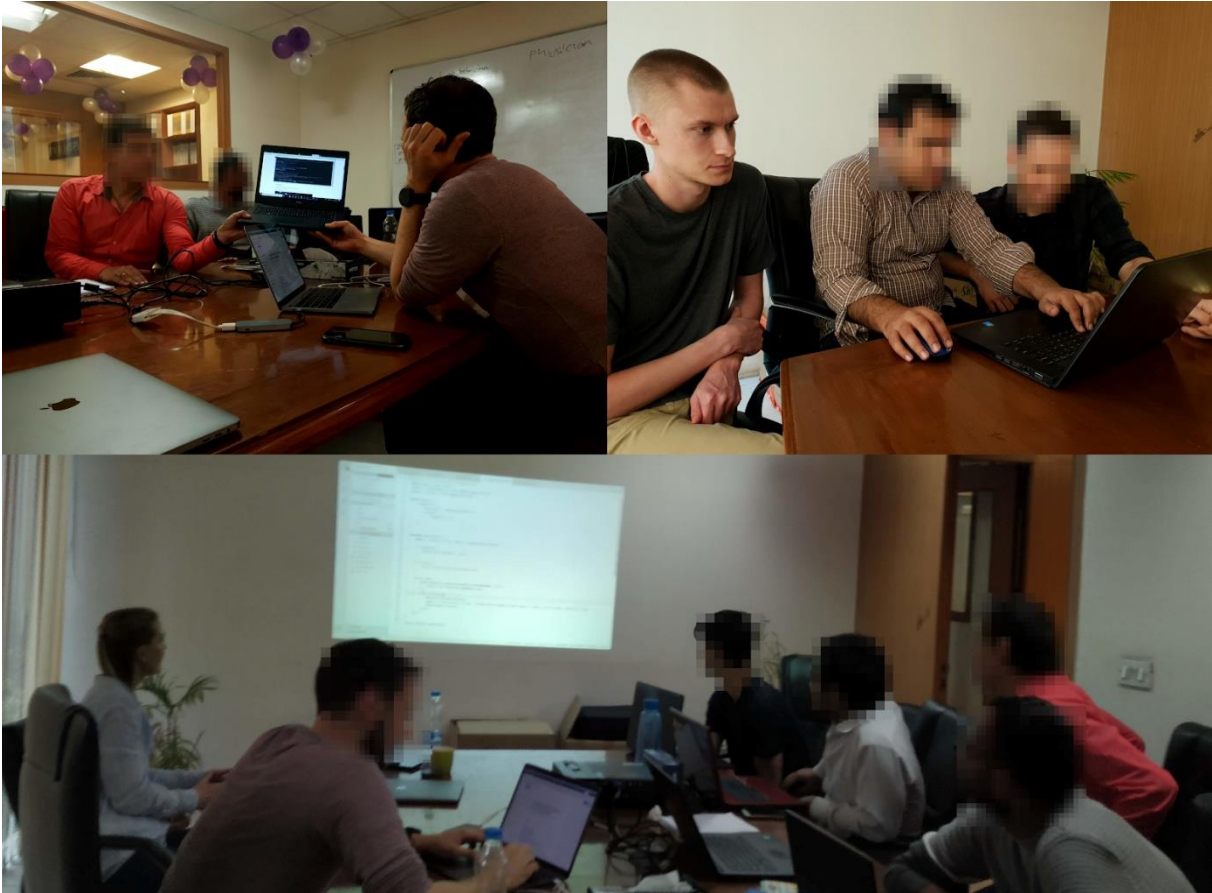
Two semi-structured interviews were held with HISP India developers. One with a senior developer with more than five years of experience in the organization, The other, a developer with less than a year of experience. The audio was not recorded in the first interview as the interviewee was unconformable with the recording. Detailed notes were taken during the interview, and it was transcribed from memory as soon as it was over. I was more familiar with

the second interviewee, and less concerned that the act of recording would affect the interview negatively, such as causing “the interviewee less open or less truthful” (Walsham, 2006, p. 232). A third semi-structured interview was also held with a senior developer from the HISP UiO team.

**Workshops**

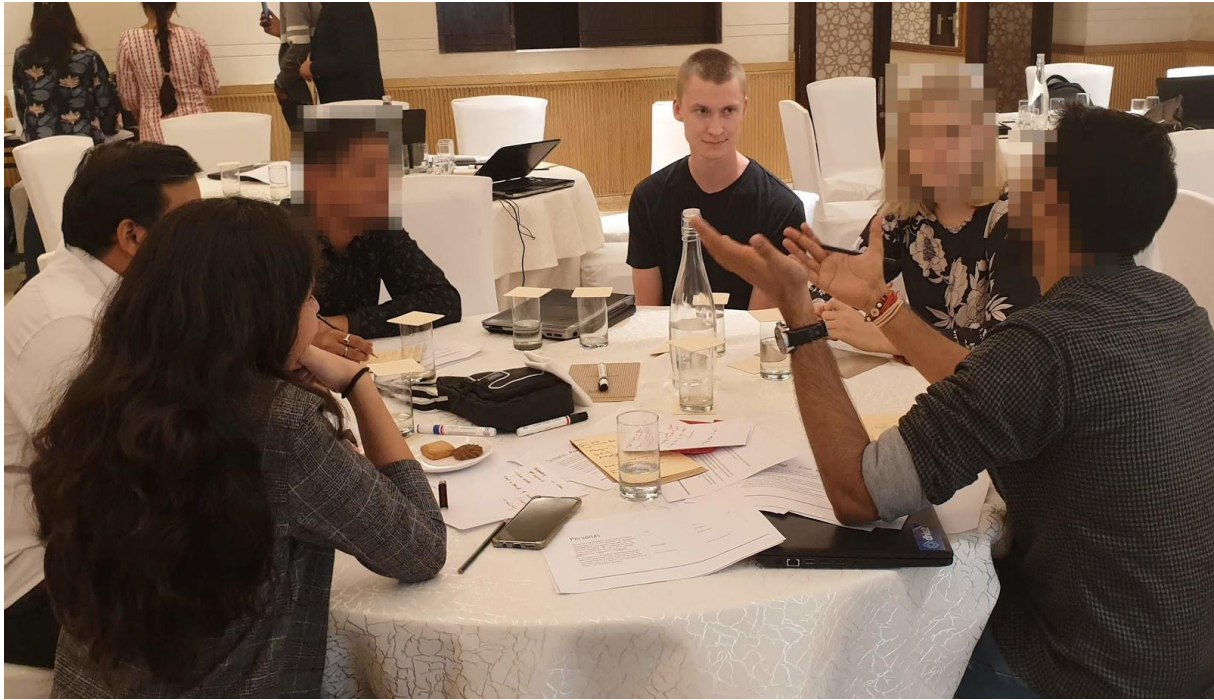
Several workshops have been part of the data collection. These workshops had sessions of discussions that were similar to what you would expect of group interviews.

The DHIS2 Application Platform Workshop was held by me, a HISP UiO developer, and a member from the DHIS2 Design Lab. All six of the local HISP India developers participated. The goal behind the workshop, which lasted two days, was to introduce new resources for developing DHIS2 apps and to get feedback from the HISP India developers. The workshop consisted of presentations, live coding sessions, and discussions. Pictures from the workshop can be seen in figure 4-4.



*Figure 4-4: The DHIS2 Application Platform Workshop*

As a member of the DHIS2 Design Lab, I was also involved in the DHIS2 Design Workshop held in New Delhi. The topic was implementation-level design. Specifically, different ways of implementing DHIS2 and techniques for doing so. A picture from a group discussion is displayed in figure 4-5.



*Figure 4-5: Group discussion during the Design Lab Workshop*

Another workshop was organized through the DHIS2 Design Lab, where the topic was Design Systems. Two architects behind a design system used in a multinational company and key contributors to a design system for DHIS2 from HISP UiO attended. Teacher's assistant from a UiO course, where students develop DHIS2 apps, also attended.

Finally, a workshop was held in Oslo related to the HISP India project I contributed to. Health domain experts and technical experts from around the world gathered for discussions. A prototype for a DHIS2 app created with the help of new DHIS2 software packages was presented.

## **Document Reviews**

The documentation of DHIS2 along with its related software packages and the software itself was studied. This was also done for software developed by HISP India. Both the HISP UiO and HISP India host their software on GitHub. Additionally, much of the HISP UiO team's

correspondence is public on GitHub. Emails from participants in a HISP India project and chat messages were also analyzed.

### **4.3.2 Analysis**

#### **Hermeneutics**

Hermeneutics was used as a method for analysis. The hermeneutic circle (Myers, 2004, pp. 107-109), one of the principles of the method, was heavily utilized in the research project. This principle refers to a constant alternation between understanding an object as a whole and understanding its parts. Understanding parts leads to understanding the whole, and vice-versa. Similarly, the method was used to understand the problem of the research project and its parts. Several times in the research project misunderstandings were clarified which later changed my holistic perspective of the research problem.

#### **Thematic Analysis**

Thematic analysis (Braun & Clarke, 2006) was used to identify and analyze themes in the data. Data was organized in a spreadsheet with codes assigned to it. Braun and Clarke (2006) describe the inductive approach to thematic analysis as being data-driven, where the themes are “not be driven by the researcher’s theoretical interest in the area or topic” (p. 83). The research question was already defined, and themes were related to the research question. Thus, thematic analysis was done through a theoretical approach, as described by Braun and Clarke (2006, p.84). The factors presented in chapter six and seven are derived from interpreting the themes.

## **4.4 Research Ethics**

### **4.4.1 Data Protection**

The Norwegian Centre for Research Data (NSD) was notified about the research project. NSD assessed it to be in accordance with the requirements of data protection legislation. Participants in the semi-structured interviews signed a consent form assessed by NSD. As required, the consent form included details of the research project and their data protection rights. As for other forms of participation, such as through the workshops, HISP India was aware of the research done through the DHIS2 Design Lab. HISP India has participated in several HISP

research projects, including several master theses. Some HISP India members are themselves, researchers. Thus, the organization was more familiar with the nature of research than what a researcher may find in other contexts.

#### **4.4.2 Research Outcomes**

I have aimed to identify a real-world problem impacting the HISP India developers, attempted to alleviate it through interventions. In words of Bang (2018, May 3): “Research *with* the people.” The participants of HISP have given me a considerable amount of their time. Some of it through participation in the AMRSS project where we worked together towards our goal. While some of it, like in interviews, mainly served to aid this research. Besides, I’ve received a considerable amount of help from HISP India for accommodation for the field trips.

Through my involvement as a software developer, I have given practical contributions to HISP India. I have also provided tools and knowledge through my participation and interventions related to reusable software for DHIS2 app development. Though I have certainly learned much from the HISP India developers.

It is challenging to assess to what degree the theoretical outcomes are helpful to HISP India. I do consider it plausible this research may have a positive impact on the DHIS software ecosystem, and thus be an indirect contribution to HISP India. Giving a voice to the external developers in the DHIS2 software ecosystem has been one of the goals of the research.

# 5 Results

This chapter presents the results of the research. It starts by describing the DHIS2 software ecosystem and how apps can be developed within it. Then the DHIS2 design infrastructure for software reuse by external app developers is outlined with experiences of how it has been used. Later, software reuse in AMRSS (a HISP project I participated in) is described. The chapter ends with software reuse practices in HISP India.

## 5.1 DHIS2 Software Ecosystem

DHIS2 has evolved from being a proprietary desktop-based software to be a modular app-based software platform. HISP UiO is putting increasingly more resources into creating resources which makes it easier to create apps for the platform for both internal and external developers. Over 30 apps are maintained by HISP UiO. These are the bundled apps, which come preinstalled with DHIS2. Because the four latest versions of DHIS2 are supported, four versions of each app are regularly maintained with updates. Some of the bundled apps have their origin from external developers in other HISP nodes. For example, the DHIS2 dashboard app displayed in figure 2-2 was inspired by a similar solution created by HISP India. The DHIS2 App Hub stores apps which can be installed to a local DHIS2 instance. However, there is no documented process for how an external developer can submit an app to be added to it.

### 5.1.1 DHIS2 Apps

Figure 5-1 illustrates the parts which make up a DHIS2 app, where the application category represents the parts that are unique to each app. DHIS2 apps have a lot in common such as, shared UI components, language handling, API calls, and authentication. Yet many of the common parts are handled by the apps themselves, often in different ways. For example, the apps do not all use the same software packages to build the apps. The team saw this as a major opportunity to reduce their workload and decided to develop several software packages and complementary resources to address this.



Figure 5-1: Parts which make up DHIS2 apps (from McGee, 2019)

HISP UiO also started actively promoting new software packages to external developers. The software packages were not only reasoned to improve productivity internally, but also improve app development productivity for external developers. One HISP UiO developer stated that instead of waiting for HISP UiO to add functionality to the DHIS2 core platform, external developers could instead implement it faster themselves. A HISP India employee explained that they had largely stopped making requests to HISP UiO for new features. The HISP India employee stated that: “It may not even be implemented. And even if it does, it may take three years to be implemented.”

By having external developers use the software packages, HISP UiO also hoped that it would improve the software and the documentation. Which in turn would make it easier for HISP UiO to onboard new developers. Finally, HISP UiO saw this as an opportunity to create a more connected community, where the external developers were not working in isolation. The new



software packages were also mandatory to use for students in a UiO course during the fall of 2019. The students found many issues with the documentation and the software packages, which help improve the software packages and the tools for using them.

### **5.1.2 Customization in DHIS2**

Implementing DHIS2 does not necessarily involve developing new apps. Existing apps are often customized and replaced by developers who are not the original author. HISP UiO hosts their DHIS2-related source code publicly on GitHub, with permissive FOSS licensing. HISP India also hosts its source code publicly on GitHub, although most repositories lack a license. This is also true for HISP Uganda and HISP Tanzania. Committing to “free and open source software and standards” is one of the guiding principles for the HISP network (Nielsen, 2019, April 27). However, without an explicit license, reusing source code is not legally possible.

As a result of the FOSS nature of platform DHIS2, external developers may alter, or “fork”, software assets in the ecosystem to fit their own needs. This is something that the HISP India developers often do. As of February 2020, 27 of the 171 repositories hosted in HISP India’s organization on GitHub are forks of repositories developed by the HISP UiO. HISP India is frequently forced to customize the platform core because government-mandated security audits find security issues. HISP India considers customizing the platform core as an undesired approach. The reason being that the code base is very complex. Also, over the years as they are doing fewer changes to the platform core they are losing more and more familiarity with the codebase. For small changes, HISP India often customizes the bundled apps. As with the core, this means that HISP India has to maintain the apps themselves if they want to use them with new releases of DHIS2. Another issue is that the apps are sometimes completely rebuilt by the HISP UiO, which you then may not be able to use. As one developer said: “We don’t want to spend time on something that will be absolute in the next version.”

As described in section 2.2, DHIS2 can also be configured by metadata and using the bundled apps. This implementation approach is often favoured in HISP India. Other approaches are usually only considered when the bundled apps do not have the necessary functionality.

## **5.2 DHIS2 App Design Infrastructure**

## 5.2.1 Reusable Software

### Legacy Software Packages

Over the years, the HISP UiO team has developed many software packages. While they were created primarily for internal use, they were published publicly on npm, a package manager for JavaScript. Two prominent software packages, D2 and D2-UI, have been deprecated and replaced by new software packages.

D2 was used for DHIS2 data management inside the app and to make it easier to exchange data through DHIS2 API network requests.

D2-UI was made to ensure visual consistency between the bundled apps. D2-UI is a UI component library. It includes small components such as buttons and larger components such as advanced tables. The components are based on Material-UI, a component library inspired by Google's design system, Material Design. The dependency on Material-UI created some challenges due to the lack of control. For example, the component used for picking the date and/or time was discontinued by Material-UI. This an important component in the DHIS2 ecosystem, and quite advanced as DHIS2 supports many other calendars than just the Gregorian calendar.

### Component Library

The DHIS2 component library is available through a software package. It has replaced D2-UI. Some components are simple, like buttons. Other components are complex and include functionality to exchange data through the DHIS2 API, like the DHIS2 header bar displayed in figure 5-2.



*Figure 5-2: The DHIS2 header bar*

The HISP India developers were introduced to the component library in January 2019. At this point, the component library was not officially released. However, it did include the most important components such as the DHIS2 header bar. The component library was also used in apps that I developed together with HISP India for the AMRSS project. Except for these apps,

the component library has seen little use in HISP India. The major concerns were the lack of support for their JavaScript framework of choice and the fact that it was not yet widely used by the HISP UiO developers in the bundled apps. The component library was released as production-ready March 2019.

## **Application Platform**

In 2019 HISP UiO began developing a series of software packages, which would later be known as the DHIS Application Platform. As illustrated in figure 5.2, a lot of the parts which make up a DHIS2 app are common. The DHIS2 Application Platform was designed a unified framework to develop DHIS2 apps by handling these parts and automate when it was possible.

A new software package to handle API calls using modern React features was made. This replaced the old D2 software package. Another software package was made to handle bootstrapping a new DHIS2 app, starting up the app for development, testing the app, and building the app. Other common features were also handled by software packages such as authentication, translation, and error handling.

The software packages, including the DHIS2 component library, were added as dependencies to the DHIS2 Application Platform. Thus, new apps bootstrapped with the DHIS2 Application Platform have the software packages as dependencies. It was released as production-ready in August 2019. The software packages are implemented using the React JavaScript framework. This means that developers who wish to use them are forced to use React. The HISP India developer uses the Angular JavaScript framework for the vast majority of their apps.

In October 2019 the DHIS2 Application Platform Workshop was held at the HISP India main office. The workshop was held by the DHIS2 Design Lab and one of the main HISP UiO developers behind the DHIS2 Application Platform, while the HISP India developers attended. The workshop included presentations, live coding sessions, discussion, and assignments where the HISP India developers spent two days developing small applications using the DHIS2 Application Platform.

The HISP India developers were largely positive about the new way of developing apps. One major reason was that it made tedious pain points trivial, such as bootstrapping an application and the header bar implementation. As one of the developers said: “The header bar itself is reason enough to use it [the DHIS2 Application Platform]”. Another developer who had made

a new app a month earlier had trouble with the header. He had accidentally used an older version of the header bar which was inconsistent with the header bar used in the other app used on the DHIS2 instance. When he realized this, he attempted to upgrade to a newer version. However, due to a lack of documentation and breaking changes, he eventually gave up on upgrading. The header bar has been changed at many points by HISP UiO, causing HISP India developers to have to upgrade to newer versions. A common practice is to contact HISP UiO when they have issues with the header bar and share pieces of code internally. HISP India has projects where several years old versions of DHIS2 are in production. Therefore, a universal implementation of the header bar is not currently possible for HISP India.

The component library also lacked some commonly used UI components such as complex data tables and inputs for date and time. D2-UI has commonly used components that are not yet implemented in the new component library.

The developers also liked how much of the UI was “taken care of”. Writing CSS is avoided as much as possible at HISP India. However, the team had concerns with the amount of spacing in UI components and that apps would not be able to fit as much information on the users’ screens.

As was the case at the earlier component library introduction, usage by the HISP UiO was a major talking point. Several of the HISP India developers stressed multiple times the importance of the HISP UiO team themselves to use the DHIS2 Application Platform for their apps, both because they liked having the UI of their apps being consistent with the bundled apps and because it would imply that the DHIS2 Application Platform would be stable and would continue to be maintained. At this time, the DHIS2 Application Platform was used for one of the bundled apps, though several were in the process of being ported over. Also, many apps used the component library at this point.

The incompatibility with Angular was also discussed. The majority of the DHIS2 Application Platform is built using React features and must be used with React. As a result, it is not technically feasible to use most of the features a rewrite of the software packages. HISP UiO uses the React JavaScript framework for the apps they develop, after switching from AngularJS. While the HISP UiO developers knew that the HISP UiO used React, they were not aware that HISP UiO had decided to stick with React for a long time. Still, the developers expressed that having to learn React would not be a dealbreaker. When discussing how this issue would affect

new developers, one developer noted: “I don’t think it’s a problem that new developers will have to use React when they use the app platform. They have to adapt.” The team expressed that they would use the DHIS2 Application Platform for all new DHIS2 apps.

The documentation was also found to be insufficient. The documentation for the UI components consisted solely of demos of the components. These demos are also used for tests, which caused the code to not always work outside of the test environment. This issue also occurred for students using the UI components in a course at UiO. New documentation has since then been added.

One developer was concerned with the number of dependencies. He had previous experience with one of the dependencies where it alone would cause the load time to be too large when he used it in an app. This is particularly important to keep in mind due to DHIS2 largely being used in developing countries where the infrastructure may be lacking.

## **5.2.2 Software Reuse Tools**

### **Design System**

In 2019 the DHIS2 Design System was made. It consists of two parts. First, principles for how DHIS2 apps should look and work. Examples of principles include how information should be presented to users, and which colours and icons should be used in various situations. The second part of the design system is components intended to ensure visual consistency and to allow developers to use more time “on building a positive user experience” (Cooper, 2019a). The components on the Brad Frost’s Atomic Design principles (Frost, 2016). Meaning that components are built using atoms, which are small components such as buttons. By combining atoms, you can create molecules, which are larger components such as menus. Finally, by combining atoms and molecules, you can create even larger organisms. The DHIS2 header bar displayed in figure 5-2, which the design system states is mandatory for all apps, is an example of an organism.

All components include instructions for how it should be used. The button, for example, has information when to use the various button types. A screenshot from some of the instructions for using the button can be seen in figure 5-3. Additionally, there is a section encouraging feedback and contribution, with instruction on how to do so.




Type	View	Usage
Basic		The most often used button that will suit the majority of actions. Should be the default choice. Several basic buttons can be in the same area.
Primary		Used to highlight the most important/main action on a page. A 'Save' button for a form page should be primary, for example. Use sparingly, rarely should there be more than a single primary button per page.
Secondary		Used for passive actions, often as an alternative to the primary action. If 'Save' is primary, 'Cancel' could be secondary. Not intended to draw user attention. Do not use for the only action on a page.

Figure 5-3: A portion from the instructions to use a button in the DHIS2 Design System

## Developer Portal

Late in 2018, HISP UiO made the DHIS2 Developer Portal as a centralized hub for DHIS2 app developers. It contains a list of upcoming events, a blog, links to documentation, and guides. The blog is mainly used to introduce new reusable software and announcing major updates to them. However, few articles have been posted in 2020. Developers in HISP India were unaware of the portal's existence.

## DHIS2 Developer Documentation

The official DHIS2 Developer guide consists of 348 pages. However, only 5 pages are dedicated to app development. The rest is documentation for the API and integration with R (a statistical computing environment and programming language). There is no information nor links to any of the DHIS2 software packages. The official DHIS2 web page does, however, have a link to the DHIS2 GitHub organization and to a repository, which is used as a discussion board for the HISP UiO. In the front page of this repository, there is a link to the DHIS2 Developer Portal.

## DHIS2 Design Lab Guides

The DHIS2 Design Lab Guides, a website with resources for getting started with React and DHIS2 app development was made by me for use by students in a UiO course. While it was designed to be used by both the students and external developers, it did not see much use by the HISP India developers.

## **Academies**

Multiple DHIS2 Academies are held every year to spread capacity globally. Attendees receive certifications and training in implementing DHIS2. There are academies of specialized topics, such as server administration and data quality. However, the academies have not focused much on topics related to the development of apps. Many of these are hosted by HISP UiO, but the goal is for the academies to be increasingly organized by other HISP nodes independent of HISP UiO (Sahay, 2019, p. 7). Additionally, there is the DHIS2 Online Academy which provides fundamentals in DHIS2. Due to the COVID-19 pandemic, many of the planned academies for 2020 have been postponed or cancelled. To combat this, more digital and online academies are in the works, including one for app development which can be completed individually like the DHIS2 Online Academy.

## **Annual Conference**

HISP UiO has arranged the DHIS2 Annual Conference (formerly known as the Expert Academy) seven times. The aim is to gather the community, share experiences, and announce new features. Participants include implementers, developers, partners, donors, politicians, and health experts. In 2019 the DHIS2 Apps Competition was introduced, where apps made by developers from around the world were promoted. The first public presentation about the DHIS2 Application Platform was also held at the 2019 DHIS2 Annual Conference.

## **5.3 AMR Surveillance System**

My primary involvement in the AMRSS project was in the data-entry module in AMRSS. DHIS2 already had a bundled app for this need. I argued, along with other members of the DHIS2 Design Lab, that while using the bundled app would reduce cost, it would be worth it to develop a new app tailored for AMRSS. It was argued that going from ICMR's tailored data-entry module to a generic app, would not be a good experience for the users. A screenshot of the app can be seen in appendix 1.

Several other apps were also developed for AMRSS. Following the literature stating that software reuse can increase productivity, two software packages were developed by me and published. First, a software package containing new UI components based on the DHIS2 Design System was made. Existing components in the DHIS2 UI component library were reused to

create new UI components. The software package requires using React. The component library from the DHIS2 Design System was used in the development of AMRDE, despite the component library lacking many components. This was done because the DHIS2 component library was not yet released as ready for production, and thus lacked some important components. Secondly, a software package to handle DHIS2 API network requests. The DHIS2 Application Platform was not yet released and did not provide this functionality. It does not require use with React. Most of the functionality of these software packages have since been covered DHIS2 reusable software.

Three apps made for the AMRSS were made without using the DHIS2 Design System. These were later remade using the DHIS2 Design System by the request of a project coordinator. The project coordinator stated that once the customer had seen apps made with the DHIS2 Design System, an app without a different UI was not acceptable for the users.

## **5.4 Software Reuse in HISP India**

The HISP India developers largely share code by hosting source code on GitHub and later reuse it in a copy/paste manner. Only one of the developers has published software packages to a package manager for reuse in the team. These software packages are only used by the publisher, although a previous member of the team used to use them.

D2-UI has not seen much use in HISP India. Developers from HISP India consider D2-UI to be difficult to use, due to poor documentation. They have mainly used it for advanced components which are time-consuming to develop. A senior developer stated that he had only ever used three of the components in D2-UI.

HISP India has decided to use the DHIS2 Application Platform for all new apps. However, the team often try to find existing apps and adapt them, rather than creating new apps. The decision on which approach to take when HISP India implements DHIS2 is based on what is least time-consuming. Concerning how the developers chose which UI component library to use, a developer commented: “It’s just based on familiarity, basically. Those choices are not fixed by the team. It’s up to you what you want to use.” The developers have to deliver products in a short timeframe. A developer that made two apps for a project in two weeks described: “If I said something would take several days, they would forget and come the next day to ask about it.”



A template app has also been made by the team. It includes some of the software packages included in the DHIS2 Application Platform. It also has the software packages created in the AMRSS as dependencies. The template app as a starting point for their new apps, instead of bootstrapping apps with the DHIS2 Application Platform. The DHIS2 UI component library has been used to some extent, but not exclusively. Other UI component libraries have been used in conjunction, including components which are included in the DHIS2 component library.

# 6 Analysis and Discussion

The DHIS2 software ecosystem has served as the case to answer the following research question: *What are some factors that characterize software reuse by external developers in a public good software ecosystem?*

What follows is an analysis and discussion based on field visits to HISP India and my participation in HISP UiO. First, the factors which address the research question is presented. Then, a discussion on app development by external developers in the software ecosystem. The chapter ends with reflections on the research process.

## 6.1 Factors Affecting External Software Reuse

HISP UiO has for many years developed and published software packages and other resources for internal use. While these were published with permissive licensing, they were not designed to be used by external developers. This shifted to some in 2019 when HISP UiO developers started developing and promoting new resources and sought feedback from external developers in the DHIS2 software ecosystem. From this, a design infrastructure for software reuse has emerged.

However, there has been little participation from external developers before the release of the resources. There are some mismatches from how HISP UiO designed the resources, and how external developers work in HISP India.

Table 6-1 presents a summary of the themes and key findings based on a thematic analysis of the results presented in chapter 5. These themes are described as factors which characterize software reuse by external developers in subsequent sections.

Themes	Key findings
Knowledge dependencies	The requirement of the React framework hampered software reuse

	Use of unfamiliar syntax in code examples has hampered software reuse
Software reuse tools	Poor or a lack of documentation has hampered software reuse
	Developers were not aware of key resources in the design infrastructure for software reuse
Internal to External	Reusable software was designed to mitigate problems in HISP UiO, which are not key problems in HISP India
	A large amount of padding in the DHIS2 Design System was not appropriate
	A large dependency in a software package had caused problems with long loading times
	Software package usage by internal developers gave external developers the impression that it was stable and would continue to be maintained
Self-reinforcement	The use of the DHIS2 Design System lead to the request of apps being remade using the DHIS2 Design System
	Highly reusable UI components enabled the development of new reusable UI components

Table 6-1: Themes and key findings

**6.1.1 Knowledge Dependencies**

Several of the key DHIS2 software packages require using the React JavaScript framework, which HISP India has limited experience with. While the HISP India developers do not see this as a dealbreaker, having to learn how to use a new JavaScript framework along with these new resources, does increase the complexity. HISP India has been unable to reuse software for the apps made with other JavaScript frameworks.

Similarly, the documentation for using the software packages has code with modern JavaScript, which was not appropriate for external developers. HISP India had issues understanding some of the examples encountered. Some popular software packages outside of the DHIS2 software ecosystem opt to have example code with older syntax in addition to modern syntax. Another option to deal with this may be through capacity building, such as with the planned DHIS2 Academies for app developers.

Mitigating cost of development is a major reason why developers reuse software (Haefliger et al., 2008). If the cost of learning to reuse the software is high, then third party developers may not perceive it to be worth it to use it. Reusable software is only an advantage when the gains outweigh the costs (Barns & Bollinger, 1991).

### **6.1.2 Software Reuse Tools**

The D2 (data manager and API wrapper) and D2-UI (a UI component library) were extensively used internally in HISP UiO before they were deprecated. Yet, they saw very little use in HISP India. D2-UI was rarely used for anything other than the DHIS2 header bar, which is deemed mandatory for all apps. The documentation was a major reason for this. The quality of documentation is an important factor for developers when they look to use software packages (de la Mora & Nadi, 2018).

It also is difficult for external DHIS2 developers to stay informed about the resources available for them. There is no information about any resources in the official DHIS2 documentation for developers. The DHIS2 Application Platform is a single software package, but it is also a framework for developing apps. It includes several software packages, which means developers must learn to use several software packages to fully take advantage of it. Fragmented documentation has been found to stagger developers learning how to use software packages (Robillard & DeLine, 2011). The DHIS2 Developer Portal can potentially mitigate this problem. However, it is already showing signs of a lack of updates, despite being relatively

new. Moreover, the HISP India developers were not aware of the portal's existence. The communication internally within the HISP UiO developer is naturally closer than the communication with external developers. External developers are more reliant on software reuse tools.

While HISP UiO has focused a lot on capacity building for DHIS2 overall, little has been done specifically for apps development. However, HISP UiO is showing signs that this will change in 2020. The Web App Development Academy and the Web App Development Workshop are both planned for the summer of 2020.

Resources in the design infrastructure for software reuse such as the DHIS2 Application Platform and the DHIS2 Design System require experience with modern JavaScript syntax and a specific JavaScript framework. However, little effort has been put into the barrier of entry that the knowledge dependencies cause. Furthermore, it has not been clear to the external developers that HISP UiO intends to stick with their choice of JavaScript framework for a long time.

The DHIS2 Design System allowed new components to be developed for the AMRSS. Its documentation for how UI components should be designed and the style guide principles enable the new components to respect the consistency of the UI of each app.

### **6.1.3 Internal to External**

The reusable software was created to mitigate challenges in HISP UiO. Namely that 1) they were overburdened by having to maintain many apps over several years, and 2) common problems were solved differently in the apps. This made it difficult for developers to move from working on one app to another. The situation is different in HISP India. They are a smaller team. It is not uncommon for a developer to work alone on an app. And they do not maintain the apps for as long as HISP UiO does. The challenges that the reusable software was designed to mitigate in HISP UiO, are not the same in HISP India.

The new software packages have issues in contexts with limited infrastructure. The component library uses a large amount of padding, making it less fitting for users with low-resolution screens. Some of the components have a "dense" option, which alleviates this. Secondly, the DHIS2 Application Platform has a large dependency which can cause a large amount of loading when the internet is slow.

The developers in HISP UiO are far away from most of the external developers and end-users in the DHIS2 software ecosystem. Without close participation, it is challenging to design reusable software for such different contexts. Designing reusable software for internal use, and later promoting it for external developers is problematic.

As the rate of use of the DHIS2 software packages increased internally in HISP UiO, so did the external developers' confidence in the software packages. This raises questions about how easy it would have been for the HISP UiO to include the external developers earlier in the development process. While it can be argued that early participation of external developers during the design of reusable software, it may be challenging for the orchestrator to accomplish.

#### **6.1.4 Self-reinforcement**

Not every app in AMRSS was initially made using the DHIS2 Design System. However, a project coordinator considered it necessary to remake apps using the DHIS2 Design System. Also, the Atomic Design principles (Frost, 2016) made the DHIS2 Design System components highly reusable. This made it possible to use existing components to develop new components, while still respected the DHIS2 Design System principles.

These findings imply that software reuse from design systems enable further software reuse. In other words, self-reinforced software reuse.

## **6.2 App Development by External Developers**

Developing new apps is not the only approach to implementing DHIS2. HISP India tends to opt for configuration whenever possible. Additionally, HISP India developers look for existing apps and customize them to fit their needs. New apps are generally only made when the bundled apps do not meet essential requirements and when there are no external apps which have similar functionality to what they need. The implementation approach is largely based on what is less time-consuming.

HISP UiO's goal of having more of the innovation in the DHIS software ecosystem coming from external developers is ambitious. Developing new apps require a lot of human capacity. Reusing software have shown to increase productivity (Barns & Bollinger, 1991; Basili et al., 1996; Lim, 1994; Mohagheghi et al., 2004, Schmid & Verlage, 2002, p. 50). The DHIS2

software packages thus have the potential to make the app development approach less time-consuming. However, these resources must be tailored to fit the need of the external developers and more should be done to enable the external developers to have the necessary tools and knowledge to use them. Additionally, HISP UiO must be mindful about the potential drawbacks from reusing software (see Eghan et al., 2019).

The goal of more innovation from the external developers also raises an ethical consideration for HISP UiO, being the orchestrator of the DHIS2 software ecosystem. The HISP nodes are closer to the end-users, allowing for more user participation during software development. The HISP nodes are also in a better position to develop tailored software. Both these factors are argued to be linked to higher-quality software. It can be argued that external app development is good for the platform, as innovative apps that are useful in many settings can be platformized by the orchestrator, absorbing the functionality into the platform. However, developing new apps is the most expensive approach to implement DHIS2. Is it in the best interest *for the external developers* to develop more apps?

## **6.3 Reflections on the Research Process**

As with all interpretive research, careful considerations should be made in trying to apply the results from this research to another setting. The factors which characterized software reuse in the public good software ecosystem which served as the research's empirical case, may not apply to software ecosystems where the aim is to generate revenue.

### **6.3.1 Challenges**

#### **Research Role**

While the HISP India employees knew I was there as a researcher on a field trip, their perception of my role change over time. Early during the first field trip, I was perceived as a HISP UiO internal developer from HISP UiO. On one occasion, a HISP India employee with more experience with DHIS2 than I did, asked me highly technical questions about one of the bundled apps developed by HISP UiO. Later, as I participated in the AMRSS project I was perceived more as one of them, an external developer in the software ecosystem.

My role as a researcher did change on some occasions, not just their perception. During the workshops involving software reuse, as took on the role of a contributor to reusable software developed by HISP UiO. It was also clear to me that I took on a role as a researcher looking in from the outside during the semi-structured interview. The signing of a consent form made the setting far more formal. I found my role as a highly involved researcher to a challenging.

### Air Pollution

The Air Quality Index (AQI) is used by the United States Environmental Protection Agency is used to monitor air quality (AirNow, n.d.). A value below 50 or below is considered good, where there is little or no risk of health problems. A value of over 300 is considered hazardous, where everyone is likely to be affected. New Delhi and the area surrounding area, where most of the field trips were spent, is one of the areas in the world with worst air pollution. The air pollution became particularly bad in November, during the last field trip. As seen in figure 6-1, most areas reached a maximum of 999 AQI. Schools and factories were shut down, planes were diverted, and traffic limitations were implemented. During this period, I stayed away from the HISP India office. This resulted in some research activities being cancelled.

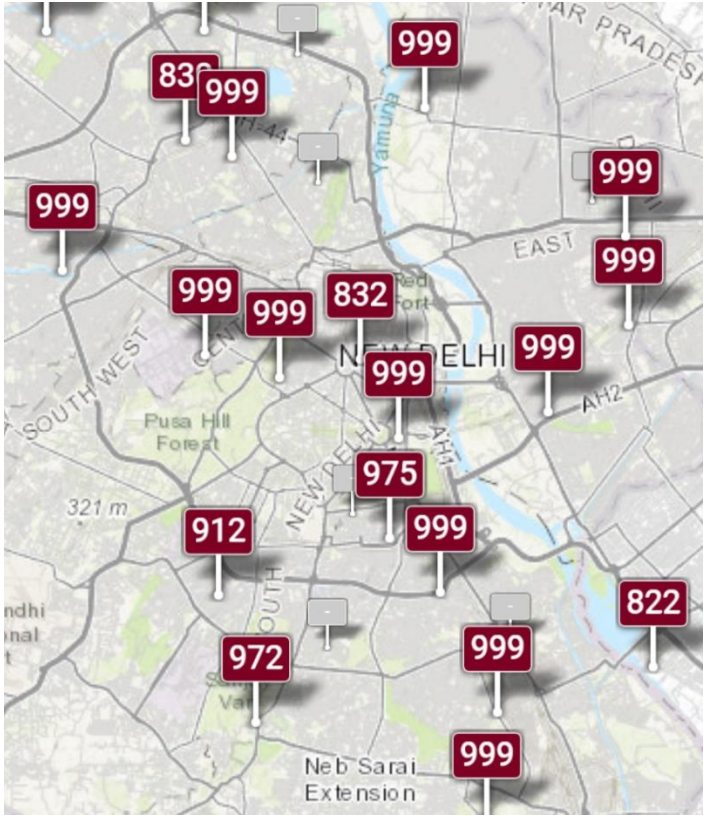


Figure 6-1: Air Quality Index during November 2019



## **COVID-19**

The COVID-19 epidemic did not reach India before well after the last field trip. It did however cause UiO to shut down. Thus, I no longer could rely on low threshold questions for members of HISP UiO. It also made it harder to reach HISP India employees, due to the shutdown of business in the country. However, as the epidemic started late in the research process, I only consider it to have had a moderate effect on the research.

# 7 Conclusion

This thesis is the result of an Action Case research project exploring software reuse by external developers in a public good software ecosystem. Through a software platform, software ecosystems can deliver products to large and heterogeneous groups of users. This is achieved by enabling external developers to create more innovations than what is feasible for a single organization. A prerequisite for a thriving software ecosystem is to attract and enable innovation by external developers.

The empirical case of the thesis is the DHIS2 software ecosystem. Reusable software, in the form of software packages, and tools for using them has been developed by me and HISP UiO (the orchestrator of the software ecosystem). Based on research showing that reusing software increases productivity, new software packages have been introduced to external developers in HISP India and experiences of using existing software packages among the external developers have been investigated. Furthermore, I have been involved in a HISP India project as a developer and I have been a teacher's assistant in a UiO course where students develop DHIS2 apps using software packages.

The theoretical contribution of this thesis is to the software ecosystem literature. By analyzing DHIS2 software packages and work practices of external developers in the software ecosystem, four factors which characterize software reuse by external developers in public good software ecosystems have been identified. These factors are displayed in table 7-1.

Practical contributions have been given to HISP India in the form of software packages and DHIS2 apps. Reusable software and tools for software reuse have been contributed to the DHIS2 software ecosystem.

## 7.1 External Software Reuse Factors

Knowledge dependencies	Software reuse by external developers is hampered by the knowledge needed to use it
------------------------	---

Software reuse tools	Resources are necessary to enable software reuse by external developers
Internal to External	Reusable software designed for internal use may not be fit for external developers
Self-reinforcement	Software reuse from design systems may encourage more software reuse

Table 7-1: Factors of software reuse by external developers in a public good software ecosystem

Software packages in the public good software ecosystem have *knowledge dependencies*. In other words, the knowledge that is needed to use the reusable software. Knowledge about the React JavaScript framework is a prerequisite to using the DHIS2 Application Platform and the component library of the DHIS2 Design System. This increases the cost of reuse by external developers. The gains of reusing software must be higher than the cost. *Software reuse tools*, such as documentation and capacity building, can decrease the cost of reusing software by alleviating knowledge dependencies. Reusable software should be designed together with external developers to ensure they fit with the external developers’ context. This can be challenging to do when reusable software evolves from being used by internal developers to be promoted to external developers in a software ecosystem.

## 7.2 Theoretical Implications and Future Research

The findings of this research support Fischer’s (1987) view that tools are important for software reuse. More research is needed on how to design these tools. More research is also needed to understand how to approach the move from internal reusable software to promoting them externally in software ecosystems. Other research could investigate if other reusable software, which is not derived from design systems, also encourages more software reuse.

As this research is based on the interpretive paradigm, no “objective truth“ can be claimed about the results and the application of the findings to other software ecosystems. Other research could explore how the findings relate to other software ecosystems and wider context.

As there are planned actions to continue to expand the infrastructure surrounding software reuse by external developers in the DHIS2 software ecosystem, future research could revisit the theme of this thesis. It would be especially interesting to follow how software reuse practices may change in HISP India.

The motivation behind this thesis is based on an assumption that software reuse by external developers in a public good software ecosystem is desirable. Other research could investigate if this is true, for whom it may be desirable, or what the positive and negative aspects are. Additionally, research could be done on what the implications are for HISP UiO's goal of more app development by external developers.

## **7.3 Practical Implications**

The factors presented in this thesis has implications for practitioners in software ecosystems, especially for orchestrators and designers of reusable software. Careful considerations should go into what knowledge dependencies the reusable software has. If external developers do not have the prerequisite knowledge, then this increases the cost of reusing software. External developers weigh the cost of reusing software against the benefits. Thus, a high cost can be balanced by designing highly useful reusable software. Alternatively, tools for software reuse can be used to close the knowledge gap. This research reaffirms the importance of documentation and capacity building for software reuse, and the negative consequences of a lack of user-participation.

# References

- Abdalkareem, R., Oda, V., Mujahid, S., & Shihab, E. (2020). On the impact of using trivial packages: An empirical case study on npm and PyPI. *Empirical Software Engineering*, 25(2), 1168-1204. doi:10.1007/s10664-019-09792-9
- AirNow. (n.d). AQI Basics. Retrieved from <https://www.airnow.gov/aqi/aqi-basics/>
- Atkins, C., & Sampson, J. (2002). Critical appraisal guidelines for single case study research. *ECIS 2002 Proceedings*, 15, 100-109. Retrieved from [https://www.researchgate.net/publication/221407453\\_Critical\\_Appraisal\\_Guidelines\\_for\\_Single\\_Case\\_Study\\_Research](https://www.researchgate.net/publication/221407453_Critical_Appraisal_Guidelines_for_Single_Case_Study_Research)
- Bang, A. (2018, May 3). Research, for whom? *India Development Review*. Retrieved from <https://idronline.org/putting-people-heart-research/>
- Banker, R., & Kauffman, R. (1991). Reuse and productivity in integrated computer-aided software engineering: An empirical study. *MIS Quarterly*, 15(3), 375-401. doi:10.2307/249649
- Bansler, J. (1988). Systems development in Scandinavia: Three theoretical schools. *Office Technology and People*, 4(2), 117-133. doi:10.1108/eb022657
- Basili, V., Briand, L., & Melo, W. (1996). How reuse influences productivity in object-oriented systems. *Communications of the ACM*, 39(10), 104-116. doi:10.1145/236156.236184
- Baskerville, R., & Wood-Harper, A. T. (1998). Diversity in information systems action research methods. *European Journal of information systems*, 7(2), 90-107. doi:10.1057/palgrave.ejis.3000298
- Barns, B., & Bollinger, T. (1991). Making reuse cost-effective. *IEEE Software*, 8(1), 13-24. doi:10.1109/52.62928
- van den Berk, I., Jansen, S., & Luinenburg, L. (2010). Software ecosystems: A software ecosystem strategy assessment model. *Proceedings of the Fourth European Conference on Software Architecture*, 127-134. doi:10.1145/1842752.1842781

- Bosch, J. (2009). From software product lines to software ecosystems. *Proceedings of the 13th International Software Product Line Conference*, 111–119.  
doi:10.5555/1753235.1753251
- Bosch, J., & Bosch-Sijtsema, P. (2010). From integration to composition: On the impact of software product lines, global development and ecosystems. *The Journal of Systems & Software*, 83(1), 67-76. doi:10.1016/j.jss.2009.06.051
- Braa, J., & Hedberg, C. (2002). The Struggle for District-Based Health Information Systems in South Africa. *The Information Society*, 18(2), 113-127.  
doi:10.1080/01972240290075048
- Braa, J., Monteiro, E., & Sahay, S. (2004). Networks of Action: Sustainable Health Information Systems across Developing Countries. *MIS Quarterly*, 28(3), 337-362.  
doi:10.2307/25148643
- Braa, J. & Sahay, S. (2017). The DHIS2 Open Source Software Platform: Evolution Over Time and Space. In Celi, L. A. G., Fraser, H. S., Osorio, J. S., Paik, K., & Nikore, V. (eds), *Global health informatics: principles of eHealth and mHealth to improve quality of care*. Cambridge: MIT Press. Retrieved from:  
[https://www.researchgate.net/publication/316619278\\_The\\_DHIS2\\_Open\\_Source\\_Software\\_Platform\\_Evolution\\_Over\\_Time\\_and\\_Space](https://www.researchgate.net/publication/316619278_The_DHIS2_Open_Source_Software_Platform_Evolution_Over_Time_and_Space)
- Braa, K., & Vidgen, R. (1999). Interpretation, intervention, and reduction in the organizational laboratory: a framework for in-context information system research. *Accounting, Management and Information Technologies*, 9(1), 25-47.  
doi:10.1016/S0959-8022(98)00018-6
- Braun, V., & Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative Research in Psychology*, 3(2), 77-101. doi:10.1191/1478088706qp063oa
- Carvalho, M., DeMott, J., Ford, R., & Wheeler, D. A. (2014). Heartbleed 101. *IEEE Security & Privacy*, 12(4), 63-67. doi:10.1109/msp.2014.66
- Checkland, P., & Holwell, S. (1998). Action research: its nature and validity. *Systemic practice and action research*, 11(1), 9-21. doi:0.1023/a:1022908820784

- Chen, W., & Hirschheim, R. (2004). A paradigmatic and methodological examination of information systems research from 1991 to 2001. *Information systems journal*, 14(3), 197-235. doi:10.1111/j.1365-2575.2004.00173.x
- Cooper, J. (2019a). DHIS2 Design System. Retrieved June 1, 2020, from <https://github.com/dhis2/design-system/blob/master/readme.md>
- Cooper, J. (2019b). Button. Retrieved June 1, 2020, from <https://github.com/dhis2/design-system/blob/master/atoms/button.md>
- Crang, M., & Cook, I. (2007). *Doing Ethnographies*. London: SAGE Publications. doi:10.4135/9781849208949
- Davison, R., Martinsons, M. G., & Kock, N. (2004). Principles of canonical action research. *Information systems journal*, 14(1), 65-86. doi:10.1111/j.1365-2575.2004.00162.x
- DHIS2. (n.d.-a). COVID-19 Surveillance Digital Data Package. Retrieved from <https://www.dhis2.org/covid-19>
- DHIS2. (n.d.-b). DHIS2 In Action. Retrieved from <https://www.dhis2.org/inaction>
- DHIS2. (n.d.-b). DHIS 2 play. Retrieved from <https://play.dhis2.org/>
- DeSanctis, G. (1993). Theory and research: Goals, priorities, and approaches. *MIS Quarterly*, 17(1), vi-viii.
- D'Mello, M. (2005). "Thinking Local, Acting Global": Issues of Identity and Related Tensions in Global Software Organizations in India. *Electronic Journal of Information Systems in Developing Countries*, 22(1), 1-20. doi:10.1002/j.1681-4835.2005.tb00140.x
- Edwards, R., & Holland, J. (2013). What are the practicalities involved in conducting qualitative interviews? In *What is Qualitative Interviewing?* (The 'What is?' Research Methods Series, pp. 65–76). London: Bloomsbury Academic. doi:10.5040/9781472545244

- Eghan, E., Alqahtani, E., Forbes, S., & Rilling, S. (2019). API trustworthiness: An ontological approach for software library adoption. *Software Quality Journal*, 27(3), 969-1014. doi:10.1007/s11219-018-9428-4
- Elden, M., & Chisholm, R. F. (1993). Emerging varieties of action research: Introduction to the special issue. *Human relations*, 46(2), 121-142. doi:10.1177/001872679304600201
- Fischer, G. (1987). Cognitive View of Reuse and Redesign. *IEEE Software*, 4(4), 60-72. doi:10.1109/ms.1987.231065
- Frost, B. (2016). *Atomic design*. Retrieved from <https://atomicdesign.bradfrost.com/table-of-contents/>
- Garg, K., & Varma, V. (2008). Software Engineering Education in India: Issues and Challenges. *2008 21st Conference on Software Engineering Education and Training*, 110-117. doi:10.1109/cseet.2008.36
- Goldkuhl, G. (2012a). From action research to practice research. *Australasian Journal of Information Systems*, 17(2), 57-78. doi:10.3127/ajis.v17i2.688
- Goldkuhl, G. (2012b). Pragmatism vs interpretivism in qualitative information systems research. *European journal of information systems*, 21(2), 135-146. doi:10.1057/ejis.2011.54
- Greenhalgh, T., & Taylor, R. (1997). How to read a paper: papers that go beyond numbers (qualitative research). *BMJ*, 315(7110), 740-743. doi:10.1136/bmj.315.7110.740
- Haefliger, S., von Krogh, G. & Spaeth, S. (2008). Code Reuse in Open Source Software. *Management Science*, 54(1), 180-193. doi:10.1287/mnsc.1070.0748
- Hejderup, J., van Deursen, A., & Gousios, G. (2018). Software ecosystem call graph for dependency management. *Proceedings of the 40th International Conference on Software Engineering*, 101-104. doi:10.1145/3183399.3183417
- Hirschheim, R. (1985). Information systems epistemology: An historical perspective. *Research methods in information systems*, 13-35. Retrieved from



[https://www.researchgate.net/publication/242501681\\_INFORMATION\\_SYSTEMS\\_EPISTEMOLOGY\\_AN\\_HISTORICAL\\_PERSPECTIVE](https://www.researchgate.net/publication/242501681_INFORMATION_SYSTEMS_EPISTEMOLOGY_AN_HISTORICAL_PERSPECTIVE)

Indian Council of Medical Research. (n.d.-a). About Us. Retrieved April 14, 2020, from <http://iamrsn.icmr.org.in/index.php/about>

Indian Council of Medical Research. (n.d.-b). AMRSN Network. Retrieved April 14, 2020, from <http://iamrsn.icmr.org.in/index.php/amrsn/amrsn-network>

International Monetary Fund. (2019). *India: 2019 Article IV Consultation-Press Release; Staff Report; Staff Statement and Statement by the Executive Director for India*. Retrieved from <https://www.imf.org/en/Publications/CR/Issues/2019/12/23/India-2019-Article-IV-Consultation-Press-Release-Staff-Report-Staff-Statement-and-Statement-48909>

Jansen, S., Finkelstein, A., & Brinkkemper, S. (2009). A sense of community: A research agenda for software ecosystems. *2009 31st International Conference on Software Engineering - Companion Volume*, 187-190. doi:10.1109/icse-companion.2009.5070978

Klein, H. K., & Myers, M. D. (1999). A set of principles for conducting and evaluating interpretive field studies in information systems. *MIS Quarterly*, 23(1), 67-93. doi:10.2307/249410

Li, M. (2019a). An Approach to Addressing the Usability and Local Relevance of Generic Enterprise Software. *Selected Papers of the IRIS, Issue Nr 10 (2019)*. Retrieved from <https://aisel.aisnet.org/iris2019/3>

Li, M. (2019b). Making Usable Generic Software - The Platform Appliances Approach. Workshop on "Platformization in the Public Sector". *Twenty-Seventh European Conference on Information Systems (ECIS2019), Stockholm-Uppsala, Sweden*. doi:10.13140/rg.2.2.11381.83687

Li, M. (forthcoming). *Facilitating Design During Generic Enterprise Software Implementation*. Research paper in review.

- Li, M., Nielsen, P. (2019). Making Usable Generic Software. A Matter of Global or Local Design? *Tenth Scandinavian Conference on Information Systems (SCIS2019)*, Nokia, Finland. Retrieved from <https://aisel.aisnet.org/scis2019/8/>
- Lim, W. (1994). Effects of reuse on quality, productivity, and economics. *IEEE Software*, 11(5), 23-30. doi:10.1109/52.311048
- Manikas, K. (2016). Revisiting software ecosystems Research: A longitudinal literature study. *The Journal of Systems & Software*, 117, 84-103. doi:10.1016/j.jss.2016.02.003
- Manikas, K., & Hansen, K. (2013). Software ecosystems – A systematic literature review. *The Journal of Systems & Software*, 86(5), 1294-1306. doi:10.1016/j.jss.2012.12.026
- McGee, A. (2019). *DHIS2 Application Platform* [presentation]. Retrieved from <https://developers.dhis2.org/2019/07/what-is-this-app-platform/>
- Mili, H, Mili, F, & Mili, A. (1995). Reusing software: Issues and research directions. *IEEE Transactions on Software Engineering*, 21(6), 528-562. doi:10.1109/32.391379
- Mohagheghi, P., Conradi, R., Killi, O., & Schwarz, H. (2004). An empirical study of software reuse vs. defect-density and stability. *Proceedings. 26th International Conference on Software Engineering*, 282-291. doi:10.5555/998675.999433
- de la Mora, F., & Nadi, S. (2018). An Empirical Study of Metric-based Comparisons of Software Libraries. *Proceedings of the 14th International Conference on Predictive Models and Data Analytics in Software Engineering*, 22-31. doi:10.1145/3273934.3273937
- Msiska, B., & Nielsen, P. (2018). Innovation in the fringes of software ecosystems: The role of socio-technical generativity. *Information Technology for Development*, 24(2), 398-421. doi:10.1080/02681102.2017.1400939
- Myers, M. D. (2004). Hermeneutics in information systems research. In Mingers, J., Willocks, L. P., (eds), *Social Theory and Philosophy for Information Systems* (pp. 103-128). Chichester, Wiley.

- Nicholson, B., Sahay, S., & Heeks, R. (2018). Global sourcing and development: New drivers, models, and impacts. *Information Systems Journal*, 28(3), 532-537. doi:10.1111/isj.12188
- Nielsen, P. (2017, April 27). HISP Groups. Retrieved from <https://www.mn.uio.no/ifi/english/research/networks/hisp/hisp-groups.html>
- O'Neill, J. (2014). *Antimicrobial Resistance: Tackling a crisis for the health and wealth of nations*. Retrieved from [https://amr-review.org/sites/default/files/AMR%20Review%20Paper%20-%20Tackling%20a%20crisis%20for%20the%20health%20and%20wealth%20of%20nations\\_1.pdf](https://amr-review.org/sites/default/files/AMR%20Review%20Paper%20-%20Tackling%20a%20crisis%20for%20the%20health%20and%20wealth%20of%20nations_1.pdf)
- Robillard, M., & DeLine, P. (2011). A field study of API learning obstacles. *Empirical Software Engineering*, 16(6), 703-732. doi:10.1007/s10664-010-9150-8
- Sahay, S. (2019). Free and open source software as global public goods? What are the distortions and how do we address them? *Electronic Journal of Information Systems in Developing Countries*, 85(4), n/a. doi:10.1002/isd2.12080
- Schmid, K., & Verlage, M. (2002). The economic impact of product line adoption and evolution. *IEEE Software*, 19(4), 50-57. doi:10.1109/ms.2002.1020287
- Subramanian, R. (2006). India and Information Technology: A Historical & Critical Perspective. *Journal of Global Information Technology Management*, 9(4), 28-46. doi:10.1080/1097198X.2006.10856431
- Thurmond, V. A. (2001). The point of triangulation. *Journal of nursing scholarship*, 33(3), 253-258. doi:10.1111/j.1547-5069.2001.00253.x
- Titlestad, O., Staring, K., & Braa, J. (2009). Distributed development to enable user participation; multilevel design in the HISP network. *Scandinavian Journal of Information Systems*, 21, 27-50. Retrieved from [https://www.researchgate.net/publication/315741117\\_Distributed\\_development\\_to\\_enable\\_user\\_participation\\_Multilevel\\_design\\_in\\_the\\_HISP\\_network](https://www.researchgate.net/publication/315741117_Distributed_development_to_enable_user_participation_Multilevel_design_in_the_HISP_network)

- United Nations. (2019). *World Population Prospects 2019: Highlights*. Retrieved from [https://population.un.org/wpp/Publications/Files/WPP2019\\_Highlights.pdf](https://population.un.org/wpp/Publications/Files/WPP2019_Highlights.pdf)
- Vesselov, S., & Davis, T. (2019). *Building Design Systems*. Berkeley: Apress.  
doi:10.1007/978-1-4842-4514-9
- Vidgen R., Braa K. (1997) Balancing Interpretation and Intervention in Information System Research: The Action Case Approach. In: Lee A.S., Liebenau J., DeGross J.I. (eds) *Information Systems and Qualitative Research* (pp. 524-541). Boston: Springer.
- Walsham, G. (1995). The emergence of interpretivism in IS research. *Information systems research*, 6(4), 376-394. doi:10.1287/isre.6.4.376
- Walsham, G. (2006). Doing interpretive research. *European journal of information systems*, 15(3), 320-330. doi:10.1057/palgrave.ejis.3000589
- World Health Organization. (2020, January 13). *Urgent health challenges for the next decade*. [Press release]. Retrieved from <https://www.who.int/news-room/photo-story/photo-story-detail/urgent-health-challenges-for-the-next-decade>
- World Health Organization. (n.d.). WHO Configuration Packages for DHIS2. Retrieved from <https://who.dhis2.org/documentation/index.html>

# Appendix 1 AMR Surveillance System

The AMR Surveillance System uses the DHIS2 platform. Several of the bundled apps are used, primarily to visual data. Several apps were developed by reusing software by me and HISP India for other purposes. A screenshot of one of these apps, the AMR Data Entry app, can be seen below.

The screenshot displays the 'AMR surveillance system - Data Entry' interface. The form is titled 'Record' and is divided into four main sections:

- Person:** Includes fields for CR number (filled with 'banana'), Name, Age/DOB (Years: 1, Months: 0, Days: 19), Date of Birth (2018-12-02), Gender (radio buttons for Male, Female, Transgender; Male is selected), State (Arunachal Pradesh), and Consultant.
- Panel:** Includes Organism group (Enterobacteriaceae), Organism (Citrobacter freundii), Type (radio buttons for Urine sample testing, Non-urine sample testing; Urine sample testing is selected), and Date of Sample (2019-12-02).
- Institute / Hospital Information:** Includes Hospital department, Location (radio buttons for OPD, ICU, Ward (Non ICU); OPD is selected), Infection type (radio buttons for Community acquired infection, Health care associated infection, Not known; Community acquired infection is selected), Prior admission in last 3 months (radio buttons for ICU, Ward, No; No is selected), and Clinical diagnosis.
- Sample information:** Includes Lab Sample ID, Identification Method (radio buttons for Biochemical test, 16s rRNA Sequencing, Whole Genome Sequencing, Proteomics; Biochemical test is selected), Type of sample (radio buttons for Urine; Urine is selected), and Antibiotics/Antifungals (Taken for 3 days in last 1 month).

*Screenshot of a form in the AMR Data Entry app*