Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Eirik Kaldahl
Martin Havlor Kostveit

Health Information Systems in Developing Countries

Designing for an effective evolutionary platform

Master's thesis in Informatics
Supervisor: Eric Monteiro
June 2020

NTNU
Norwegian University of Science and Technology

Eirik Kaldahl
Martin Havlor Kostveit

Health Information Systems in Developing Countries

Designing for an effective evolutionary platform

Master's thesis in Informatics
Supervisor: Eric Monteiro
June 2020

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU

Kunnskap for ei betre verd

Abstract

This thesis focus on scalable platform ecosystems, and how to design for an effective evolution of these platforms. Scale often comes at the price of specialisation, so we explore methods for making platform development more agile, to accommodate for the eminent lack of specialisation in large, generic platform ecosystems. Our data collection is the result of four weeks of fieldwork in Zomba, Malawi, were we participated in an ongoing development process to make a unified health information system for Malawi. Using the District Health Information System 2 platform, we made a prototype, while conducting research on how the development process were coordinated. The research method used is based on an action research approach, which entails simultaneous observation and participation of a process. Our findings tells a story about poor infrastructure and computer knowledge among the end users, influencing design decisions. This ensued systems with a difficulty to identify patients and store their records. We discuss tools that can be used to help the distribution of decision making, and partition the system into smaller ecosystem, which can evolve separately and in accordance with its' environment. The suggested approach is designing for metadata, by utilising an Adaptive-Object Model pattern. This allows for rapid changes to business needs, by altering variables that changes the domain at run-time rather then code. We argue that this will encourage more agile processes in an environment otherwise characterised by rigid implementations, and shifts power to the end users.

Sammendrag

Denne avhandlingen setter søkelys på skalerbare plattform økosystemer og hvordan design kan påvirke plattform evolusjon. Et systems skalerbarhet kan være med på å redusere evnene systemet har til å tilpasse seg. Vi utforsker derfor metoder for å øke smidigheten til en plattforms utviklingsprosesser. Datakolleksjonen vår er et resultat av fire uker feltarbeid i Zomba, Malawi der vi deltok i en pågående utviklingsprosess med hensikt å utvikle et enhetlig helsesystem for Malawi. Ved å bruke plattformen District Health Information System 2 lagde vi en prototype. Samtidig, forsket vi på hvordan utviklingsprosessen ble koordinert. Forskningsmetoden vår er basert på aksjonsforskning-metoden, noe som innebærer å observere samtidig som man deltar i en prosess. Våre funn kan fortelle om dårlig infrastruktur og lave teknologiske kunnskaper blant sluttbrukerne. Systemet som blir brukt som et resultat av dette har vanskeligheter med å identifisere pasienter, logge og lagre data om dem. Vi foreslår verktøy som kan brukes for å fordele beslutningstaking og partisjonere systemet slik at hver del kan utvikle seg separat og i samtid med miljøet rundt. Vår tilnærming er å designe for metadata, ved å bruke et programvaremønster kalt Adaptive-Object Model. Dette tillater raske endringer av organisasjonens behov ved å endre variabler som definerer domenet istedenfor kode. Vi hevder at dette vil oppmuntre til mer smidige prosesser i et utviklingsmiljø som ellers er preget av rigide implementeringer, og flytter makt til sluttbrukerne.

Table of Contents

Acknowledgements

We would like to take the opportunity to thank everyone that has helped us with this thesis. First, we want to thank our supervisor, Eric Monteiro, for his his insights and useful guidance. Without you, this wouldn't be possible. We also want to thank Magnus Li, who helped us with local knowledge of DHIS2 development and assistants in finding the location of our fieldwork, in addition to introducing us to relevant literature. Lastly, we want to thank Tiwonge Manda, and the entire team in Malawi, who helped us with accommodations in Malawi and field expertise.

List of Abbreviations

AOM	=	Adaptive-Object Model
API	=	Application Programming Interface
ART	=	Antiretroviral Therapy
CSIRO	=	Commonwealth Scientific and Industrial Research Organisation
DHIS2	=	District Health Information System 2
EMR	=	Electronic Medical Record
EU	=	European Commission
GIS	=	Geographic Information System
GSMA	=	Global System for Mobile Communications Association
HIS	=	Health Information System
HISP	=	Health Information System Program
HSA	=	Health Surveillance Assistant
IIMC	=	International Information Management Corporation Ltd
ISP	=	Internet Service Provider
IT	=	Information Technology
LAN	=	Local Area Network
MVP	=	Minimum Viable Product
OPR	=	Outpatient Registry
OSS	=	Open source software
PD	=	Participatory Design
TB	=	Tuberculosis
UDP	=	User-Defined Product
UiO	=	University in Oslo
WHO	=	World Health Organisation

List of Figures

Chapter 1

Introduction

1.1 Motivation

Economies of scale is often obtained in information systems using platform ecosystems as they possess scalable traits. While scalability is desired by the organisation, users want specialised implementations for their needs and situation. These desires are contradicting, as scalability can only be achieved through generification. The systems' relevance is also a very important consideration as the environment is never constant and in turn, systems often gets outdated. Thus the scale has a positive correlation with the challenge of keeping its relevance. Considering systems' evolutionary traits is therefore of increasing importance with scale, because boundaries can mean that systems have no room to evolve. Scalability and relevance is not something that happens on its own, it needs to be carefully considered by the developers. We would suggest an agile development process, but acknowledge that it does not share platform ecosystems' ability to scale.

This thesis focuses on scalable platform ecosystems and how platform organisation influence it's ability to change to its environment, which we call evolutionary dynamics. We encourage a discussion regarding economies of scale, where we argue that scalability often comes at a price. Because platform ecosystems need to be inherently generic, implementing user specific solutions appear contradicting. By encouraging the platform to easily evolve however, we can keep the modules relevant even in a changing environment. We discuss how we can encourage platform evolution by decentralisation of governance and facilitating for decision rights partitioning within organisations. We use the District Health Information System 2 (DHIS2) platform as an example and recommend some changes to the platform. The thesis is written in collaboration with the University in Oslo (UiO) and is a contribution to the Health Information System Program (HISP) (2.3.3).

Our case study involves fieldwork in Zomba, Malawi, where we were part of a development process of a new HIS project. The project researched the possibility of a collective HIS for Malawi. Developing countries suffer from scarce resources which mean that development decisions have greater impact. A failed attempt or wrong focus can mean make or

brake in many situations as resources are not available to compensate or try again. Africa have long been in this developing stage, and with all the new technology available they will evolve faster then earlier nations have (Rosling et al. (2018)). The importance of a systems' evolutionary dynamics complimenting the evolutionary speeds of the environment is therefore even more relevant in the developing world. The evolutionary speed is evident from how they have adopted mobile technology before ever adopting computer technology (Lule (2018)), and also true in regard to health information systems (HIS) where they now want to implement their systems on smartphones. Yet, the change is not constant in all contexts and the speed differ greatly from situation to situation. Systems need to be able to develop individually from one another. We encourage different implementations by designing for incompleteness, giving decision makers power to change systems through an Adaptive-Object Model (2.2.1).

The original assignment text sounded as follows;

Health Information systems in developing countries *"Developing countries have limited resources for healthcare delivery hence need to make the most of resources available. This project / thesis is linked to the HISP / DHIS2 (www.dhis2.org/) initiative. Based on open source software, HISP aims at increasing the efficiency and quality of health services by enhancing the necessary reporting of health status. Mobile technologies are crucial as, even when roads and electricity is patchy, there are mobile phones. The approach of HISP is pragmatic: rather than elaborate, complex 'perfect' solutions, HISP provides simple and robust ones that have a realistic chance of uptake. HISP, across Africa and Asia, is implemented in about 50 countries in varying degree of completion. It is one of the world's largest systems serving patients in the Global South, measured by size of the caption population. The project/ thesis involves empirical fieldwork in Africa or Asia on selected services of the HISP portfolio. The purpose of the work is to identify requirements and subsequently help implement these as part of the evolving portfolio of HISP software. The HISP project is managed by the University of Oslo (UiO). This project / thesis will be in collaboration with the UiO team."*

1.2 Research questions

1. Designing for an effective evolutionary platform

   (a) *Economies of scale: Scalability vs. specialisation.* Scale comes at the price of specialisation and developers need to make considerations of global or local development. How does the focus on scalability effect the evolutionary dynamics of the platform?

   (b) *How to make platform development agile?* Usually platform development follow a waterfall development process and choices are made at the beginning of the development process rather than on the go. We acknowledge that this impair evolutionary dynamics and see how certain techniques mitigate the issue through;

   - Making partitions?
   - Splitting design into implementation-level and generic-level?

   (c) *How the internal organisation can distribute governance?* Often decision making distribution resemble the hierarchical organisation which impair evolutionary dynamics. We look at actions the organisation can do to counter these effects.

1.3 Structure of the thesis

Our thesis is made up of five main sections. Firstly we examine related topics through a literature review, where we focus on user centred design principles and generic development processes. We also look into health information system and review the HISP organisation and work processes. The topics we discuss in literature review are meant to explain concepts we use in discussion and which should be explained. After the literature review we go through the case methodology by showing how we gathered our information. The chapter is meant so that anyone can recreate our findings and to know how we got the the research contribution we are presenting in this thesis and how we interpreted the data available to us. The case section is where we present all findings from our field work in Malawi. It is solely our experiences and how we interpreted the findings. It introduces some experiences and prerequisites for development in Africa and Malawi and detailed description of the prototype we made in collaboration with the locals in Malawi.

The discussion chapter is where we contribute to the field of research being health information systems, platform development and specific recommendations to the DHIS2 system. We try to have a general focus but use examples from our experiences to support the discussion topics. Finally we conclude our discussions and recommend further research topics and improvements which could be done to our process.

Chapter 2

Literature Review

2.1 User centred development

As part of the recent excitement for IT systems and the realisation of how it can solve user specific problems there has been a trend of involving users in the developing process to solve more user specific problems and increase the usability of the system. Usability in the context of computer science have become a new science on its own. It refers to how well an artefact works in regard to a specific set of users within its context. It is often measured through *learnability*, *memorability* and *user satisfaction*. Users' needs often change and are not fully categorised before development start. The process is therefore in need of constant reevaluation and many businesses adopt *agile methods* for this purpose.

2.1.1 Agile methods

Agile methods are development process principles that aim to reduce cycle time of developing new technologies. Today developers have reduced the cycle time of developing new technology due to an increase in computer power and utilisation of new processes. This means that developers have a lower threshold for creating new software and more user specific experiences. Especially when talking about agile processes compared to old methods like waterfall, we see developers reach out to their end users and create issues based on their response to the product. For instance, creating a minimum viable product (MVP) and testing it on the presumed end users has become an industry standard in many startups where the end user is in focus (Ries (2011)).

The main idea of agile developing is that development is evaluated continuously and split up into small periods often referred to as sprints.

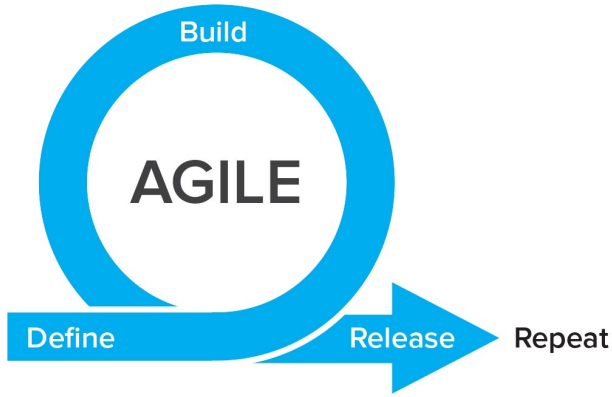

Figure 2.1: Agile development process

As shown above in (Figure 2.1) each sprint starts with defining what is to be , then the build and release. When this is done the implemented build is evaluated and a new sprint starts with the same approach. The evaluation often include the customer and user. The main advantage of agile development is that the system is continuously evaluated and therefore ensured to be developed in the intended direction. Working close with the customer and including them in the process ensures that changes can be made without using to much time down the wrong path.

These processes usually follow a loosely defined set of steps. Firstly you have an introductory sprint, often called sprint zero, where you plan how to realise the project at hand. You will have to define the problems to be solved, the context, form and goals of the problem and with this set some requirements for the system. These take form of functional and non-functional requirements. Functional requirements are requirements the system specifically need to do while non-functional requirements are goals to be reached in regard to the operation of the system. With the requirements set it is helpful to make personas which represent some users of the system. The developers task is then to solve the requirements set in regard to each personas. It is most often very helpful to make a minimum viable product which can be tested before the system goes into production as production requires a lot of resources and changes are harder and more *expensive* the later into production they are made. This is in very rough points how user centred development can be achieved effectively and with high quality. It however require some very important knowledge about the system. To solve the user specific requirements the users have to be visible, the context which the users are in must be conceptualised and the goals of the system must be formalised. This make up the motivation to start the development. In small systems this can happen organically as the project is not initialised before the motivation is defined and the motivation is user specific. In larger systems however the motivation is often not as specific and the end users hard to reveal.

As Garud et al. (2008) discusses, the system can be decomposed into smaller *entities* which collectively solve the purpose intended and can individually solve smaller and

preferably user specific problems. While the collective system focus on one general goal each decomposition can focus on specific tasks. This makes it easier to organise a development team into developing large scale, dynamically influenced systems.

2.1.2 Participatory design

A central part of succeeding in information system development is through involvement of the end user (Berg (2001)). The principles of participatory design (PD) sheds light on this by giving the end users more presence in the development process. Traditionally, projects using PD were aimed at empowering workers at their workplaces (Braa & S. (2012)), and today the core principles are much the same. They are based on changing the end users' role in the process, from being an informant to being an active participant and decision maker. Bodker et al. (2004) argues that the participation becomes more real when the end user is in on the process, i.e drawing sketches, rather then just answering questions. According to *Routledge International Handbook of Participatory Design*, by Simonsen & Robertson (2012), the PD participants usually takes on the role of either a *user* or a *designer*. The users tries to communicate their desires while learning about the technological reach of the system. The idea is that the users have to understand the system to be able to know what is feasible to create in terms of design and functionality. The designers try to help the users learn, while simultaneously trying to learn about the users' real needs.

Different PD approaches also depends on setting. While the above explain how to preform PD in a perfect vacuum, this is never the case. In developing countries, PD can be a challenge since the ground work is not laid down. One of the prerequisites for PD to work is that the participants is familiar with the technology in question, since they are supposed to engage and contribute with useful input. In developing countries however, there is a general lack on technological skills among the people, mainly due to lack of computers, infrastructure, internet and education in these areas. For PD to work on this environment, one must first address this issue, and train the users in the appropriate technologies before starting PD work. A suggested approach, by Kimaro & Titlestad (2008), is to use participatory customisation, which is similar to PD, but is based on users and developers collaborating on adapting a known system to their needs. It was shown that when users had training and knowledge about the system, their contributing also grew.

End user involvement

When relying as much on the end users contributions as you do in PD, it is important to be aware of what type of contributions they are making. As mentioned earlier, the user has to understand both the system and its limits when participating, to be useful. In addition to this, the developers must remember that one users thoughts and ideas might not reflect the majority of the user bases thoughts and ideas. Kushnriuk & Nøhr (2016) addresses this issue, stating that for large information systems where the user base contains thousands of people, it may be difficult to figure out how representative a given user is. Other sources suggest opposite opinions, however, as Kujala (2003) agrees that finding representative users is difficult, but follows up by stating that with too many user voices it may be an

issue to find consensus. They also mention that users may need to be educated in certain design aspects before engaging with the design team.

2.2 Platforms and generic development

The majority of developing teams today adopt some form of an agile work methodology were they involve the end users in one way or another. Especially when talking about development of large systems, where the process is iterative and the system requirements and preferences is expected to change, it is important for the developers to adopt techniques that reduces the resources needed to functionality and design.

2.2.1 Generic development

A known technique for helping resource management for large systems is developing for generic use. This is generally split up into two fields. The first field is creating a generic code base for high modifiability and reusability of components. The second is creating generic software, which can be used for a broad amount of use-cases and instances of a system. We will go more thoroughly over these in the next sections.

Generic code

As the systems' preferences is expected to change developers use agile processes. Developers evaluate their progress in each cycle and make changes in a relatively short period of time. It is therefore important that the code is easily editable. This is where generic code comes in.

The idea of generic code is that code should be developed in a way that it can be described in different ways to be reused for many purposes. This is a concept that makes a developers task more effective as less code is needed to describe a complete system. Generic programming is a well known concept when developing systems but recently it has gotten a new meaning. When exercising generic programming it is hard to also solve the problem of user centred development to achieve high usability. They are contradicting concepts that are very difficult to perform symbiotically. While generic software is a process of generalising code to be reused, user centred development focus on specifying a specific user demand. One solution is to separate the implementation phase from the development phase and focus on developing a generic core system so you can implement it in many use cases. The idea is that local designers should use the core code base to implement different solutions that solves the use case at hand, without writing any new high level code into the system. Some systems that utilise this concept is described as low-code software. Low-code comprises a part of the development process where the designers implement solutions with very little or no actual programming code.

Component based development A form of generic coding is component based development. When developing UI elements and following the generic software approach it is often *best practice* to develop individual components for reuse. Components are described

individually but the content is supplied when the component is implemented in the system. Component based development is an old term, but with development platforms as React, Angular and others the term have gotten a new meaning being an important one in current and future development. The term describes a way of developing independent generic components that can be manipulated in different ways for reuse. This makes for a reduced cycle time as you only define a component once and reuse it for any purpose. Following the standards of React development, for instance, is a very good example of how these element are created, where you usually split up the code in different components. A component could be any segment or part of a segment of the system. For instance, a component could be a *card*, like in Figure 2.2. Instead of coding multiple of these card, one card component can be made and then called upon where needed with the right parameters.

Figure 2.2: A card, from W3School

Big projects controlling big systems demands a well organised code structure and high reusability. By creating independent generic components this can be achieved.

Generic software

Issues may arise when developing a so called generic core system. Pollock et al. (2007) brings up some of these issues when talking about how to accumulate functionality in the ”birth stage” of a system. They describe the perfect situation as making generic templates and alter these to the local needs. In the empirical study they did, they found that with each new customer to the system, new modifications to the template was required. The reason these templates were made in the first place, was to reduce development time and costs. Since requirements continued to increase, they found that something needed to be done about the method of obtaining them. What they found was that by shifting the level of implementation away from the individual and to the community, the requirements could become more generic. In short, by staying close to the different costumers when creating generic software will in most cases make the software non-generic. The different costumers all got their different requirements, and when the software tries to fit all the different requirements it will end up becoming a semi tailored solution with a couple of features some of the costumers won’t even use. By creating requirements in a community

on the other hand, the community of costumers can find the highest level requirements that fits everyone.

While small, idiosyncratic functionalities can wait for a local implementation when creating a generic software, there could arise a split in the community defining the requirements. Pollock et al. (2007) also touches on this with an example from their empirical study. When a generic software for student management was implemented at universities in both Europe and America, they realised that the universities had different rules for how to progress students from one year to another. Since this was such a big feature, it implied that the two different institutions should have two different versions of the software. The solution they came up with was to make two generic templates, instead of just one.

Local changes to generic software We have discussed the different issues that a completely generic software can lead to. It can become too generic and not fit in anywhere. It can even have to many specific functionalities and be regarded as non-generic. For generic software to be effective, especially regarding systems like DHIS2 which is made to be able to support multiple health institutions, it needs to be able to implement local needs. When DHIS2 was rolled out nationwide in Kenya by the Ministry of Health, there were uncertainties regarding the internet coverage in parts of the country. It was then decided to implement the system on one online central server supplemented by standalone installations for offline use. This local implementation made it possible for the workers with poor to no internet connection to use the system. Offline data would be transferred as soon as the device got connection, together with offline data entry features from HTML5 to manage unpredictable connection issues Manya et al. (2012). Changes like this does not only help the institutions that are creating it, but also everyone adopting it. It is even fair to say that local changes to the generic software help with the innovation of the product. Soon after Kenya rolled out its version of DHIS2, countries like Ghana, Uganda, and Rwanda followed and adopted the same approach Poppe et al. (2013).

Adaptive Object-Model

Adaptive Object-Model (AOM) is an architectural pattern that renders components defined by data interpreted at run-time (run-time meaning from when the program is opened/executed to when it is closed/terminated). It is often called a *reflective* or *meta* architecture, because it renders these components based on the information it receives from some source of metadata. The main aim of AOM is to make the system more configurable and dynamic, to suit the systems' needs. In a business setting, this could mean managing existing products or extending a system to add new products. With AOM, tools can be made so inexperienced users can modify the system and its design based on simple variables.

So how does AOM work? AOM tries to represent data like classes, attributes, relationships and behaviour as metadata, often by storing it's object models in the database. When the system starts up, it interprets all the metadata stored, and renders it as system components. We will explain this with an example, creating a survey with AOM. By using AOM, we would have a lot of metadata about the survey, e.g name, gender and age. These data points would also have attributes attached to them, such as string, boolean and integer. When the system is started, the metadata would be sent to the system, and the system would interpret it and sort out how to show it as components. For example, the name field

would be interpreted as a string input field. The whole survey is then rendered like this. If someone added, modified or removed something from the metadata, these changes would then be reflected on the system the next time it is ran. This also shows how anyone can modify the system without using code. AOM is a great asset in any generic system, as it can work as a way to fit these systems into more specific contexts without having to redefine the whole system code. An example of this is the User-Defined Product framework (UDP). The UDP framework was developed as a system used to represent insurance policies, but in reality it can make any complex business objects. It introduces an easy way to modify these different objects and their attributes. For the initial purpose, it could produce insurance policies, with a type (car, house, etc), a price and a duration, among other things. But as Yoder & Johnson (2002) points out, it can be used in a variety of scenarios such as a bicycle manufacture that needs to describe the different bike models it sells. This means that whether you are a bike manufacturer or an insurance manager, you can build these components so that others, such as a salesperson, can customise a bike or an insurance policy based on the models created. This highlights the key aspect of AOM, which is to be able to quickly adapt to a business needs such as adding new attributes and components on demand.

2.2.2 Application Platform Development

Platform-based ecosystems are in many markets the dominant model for software development. The theory behind such ecosystems consist of a generic core which provides a shared interface for which modules can solve specific problems. They depend on a diverse developer community to add on functionality to the generic core. This has been argued to inspire for a much faster evolution of a system as everything is potentially replaced and improved by individually creative thinkers developing specialised features for their use. Examples of systems like this is iOS with its 140 000+ apps and Firefox with its 8 000 add-on extensions. Many of the relatively small applications or add-ons of the aforementioned systems have added functionality that the ”owners” themselves hardly could have imagined. The key being that the developers that contribute to the platforms make systems that solve a problem they themselves have discovered and realised could be solved with the building blocks provided by the platform. They add specialised, user specific functionality which in a way is a shortcut for the creators of the systems but also a way of ensuring good user specific content. We are very familiar with some large platforms that we use everyday (e.g iOS, Android, Google Chrome). They have created platforms that are inherently governed in the users/developers and their output is not very controlled, almost anything can be created at any time. How these systems are run and controlled is very much up to the owners of the systems.

Creating a platform based system is an exhaustive activity and should be considered carefully before starting the development process. The system is expected to have a long active life and priorities and focus elements often evolve in time. The choices made at the beginning of the platforms life should therefore accommodate for changes unforeseen. This may sound impossible but following some architectural principles the problem can at least be mitigated. The system need to permit changes to individual modules without compromising their ability to function together again (Tiwana et al. (2010)).

The platform architecture describes how the systems partitioned code base interoperate

with other partitions and modules in the system (Tiwana et al. (2010)). When considering how the ecosystem of partitions and modules should work together it is ideal to ensure that the core exhibits low variety and high reusability while the *rules* for adding modules ensure the modules produced exhibit high variety. The partitioned ecosystem are also more adjustable as they are smaller in size and work on the specific partition features is easier then it would have been had it been one single entity.

2.2.3 Open source and community driven development

Open source software (OSS) is software that is publicly accessible (opensource.com" (n.d.)). The idea is that anyone can access the source code and modify it. The counterpart to open source is called *proprietary* or *closed source*, and means that only the organisation or person that created the software, controls it. Weber (2004) describes the creation of open source software as an experiment in social organisation around a distinctive notion of property. Property, in this statement, is not like the conventional meaning of property, where it means something that is yours, and that you can exclude others from using. It is more centred around the right to distribute something, rather than exclude.

Being a part of a OSS team often means working without any sort of remuneration. However, Hann et al. (2004) argues for other motivations for contributing to OSS projects. They list five different reasons, which we will briefly go over here.

- The first one is for *normative* reasons, which means to become esteemed within a certain circles, whether it's friends, online communities or other circles.
- The next one is called *values*, which means contributing because you strongly believe in the core values of the OSS. An example could be wanting software to be free, hence helping OSS software to compete with the proprietary ones.
- Motivation for participating in OSS development also comes from wanting to learn. The idea of *understanding*, means wanting to become familiar with different types of technology, and therefore entering a community with knowledge in these technologies.
- Another motivation can be for the purpose of acquiring knowledge about something career-oriented. This is similar to the understanding motivation, but the technologies in question might not be as freely chosen. As an example, someone could be involved in OSS development to acquire skills to use on their resume for job searching.
- The last one is called *Ego Enhancement* and is related to participation to boost your own self-esteem or personal growth. Also similar to understanding, but centred around the thought of accomplishment.

OSS is made by a community of people, all with their own motivations. But how are OSS projects typically governed? O'Mahony (2007) refers to open source governance as having direction, control, and coordination of a community of individuals and organisations in regards to the project they contribute to. While there are many ways of govern a given system, when talking about OSS, the most used model is community-managed.

By gathering data from four large OSS communities (the Apache, Debian, GNOME, and Linux Standards Base), O'Mahony identified five key features of community managed governance. We will briefly go over each one.

- Independence - The software is not dependent on any one sponsor, but diverse in where it gets its funds and other support from.
- Pluralism - The software management is maintaining multiple ideas in regards to approaches, methods and individuals point of view.
- Representation - All members of the community have the right to be represented in decision-making of the whole project.
- Decentralised - No organisation has the sole decision-making rights. The contributors have access to all different decision-making levels, from code level (new features, bugs, etc) to community-wide decisions (e.g change in process or structure of the leadership).
- Autonomous participation - Any member of the community can contribute in their own way, This means that the governance model should promote participation based on a members' own motivations and skills.

2.3 Health Information Systems

Health Information Systems (HIS) are systems that process data, information and knowledge in health care environments. HIS are large and often integrated into the public records where the system is implemented. This makes the systems complex and often hard to develop to be generically modified. Additionally the systems have been implemented and developed for years, and being the field of medicine the *try and fail* methodology is not a popular approach as the consequences can be dire. The systems are often a collection of many smaller systems that each solve a problem faced in the field of medicine. They have in the past been institution based systems that are specific for each field of medicine and specific treatments. Recently, however, there has been a trend shift to a more patient-centred approach Haux (2006). As the aim of HIS is to contribute to a high-quality, efficient patient care and following the idea of user-centred design (2.1) focusing on the patients rather then the institutions.

Health Information Systems rarely have the same prerequisites. Some health systems are private, some are public and countries often have a shared amount of both. This means that no one system can solve the problems faced in the health industry. A HIS needs to consider patients interactions, nurses interaction, doctors interaction and process all data connected to all different technologies you can find within different institutions. A lot of the data is analysed and used in health care planning and clinical research, which makes up the basis for further medical advances and help doctors diagnosing their patients.

2.3.1 Open source health systems

As mentioned above, a major part of all HIS are used in public institutions, and when talking about public institutions we have to talk about budgets. A public institution does

not always have the funding to go for the best and most expensive equipment or software. These institutions needs to find other ways of acquiring this, that doesn't involve spending a big amount of money. An example of this is the Centro Médico Nacional la Raza, a major hospital complex in Mexico Ciy, funded by the government. Instead of using a huge chunk of their budget on a paid software system, they chose an open source software (Webster (2011)). Webster also talks about how open-source is becoming more popular by mentioning how software like OSCAR in Canada, Open Medical Record System in many parts of Africa, SIGA Saúde Health Information System in Brazil and DHIS in big parts of Africa and Asia, gaining more attraction.

While commercial companies argue that open-source systems has more potential for bugs and security breaches than commercial software does, a recent analysis from Reynolds & Wyatt (2011) state that this is simply not true and in fact open-source systems are more repellent of external attacks. The argument is that because open-source systems allows for an "independent assessment of the security of a system" it "makes bug patching easier and more likely, and forces developers to spend more effort on the quality of their code".

A general understanding of why open source software works and thrives, especially within health, has something to do with who is involved. The resources invested often comes from people with knowledge and understanding of which problems exists. They are also often the ones with these problems, hence becoming a part of the open source community in order to fix the problem. Both knowledge and passion about the software is what fuels these open source systems, which a lot of the times are more valuable than e.g. consultants.

2.3.2 Development in developing countries

The need for offline systems

According to a report from 2019 (GSMA (2019), on mobile internet connectivity in Sub-Saharan Africa, there are around 3,5 billion mobile internet subscribers globally, which accumulates to about 47% of the population. In Sub-Saharan Africa, however, only 24% have access, which reveal that they account for about 40% of the population not connected by mobile network. Furthermore, studies like Ajuwon & Rhine (2008) and Hilbert (2014) suggests that mere access isn't even sufficient, but must be backed up by effective usage of the technology. According to the GSMAs digital inclusion report from 2014, this is an issue for communities all over Africa (GSMA (2014). Countries with the highest African mobile internet connectivity statistics, like South Africa, experience that internet access is very slow and expensive, unreliable and sometimes unavailable. These problems also indirectly impacts transnational projects and organisations because of the mismatch in responsiveness between different locations (Wyche et al. (2010).

A strategy proposition for e-education in South Africa, made by Walls et al. (2015), presented a conjecture claiming that offline platforms are more suited to the requirements of most rural or peri-urban communities. They argue that offline decentralisation of hubs and mesh networks will make resources more available, and that scaling up to online platforms and cloud based solutions in the future would be a more viable strategy.

Implementation of offline systems

Today there are countless projects that utilise offline options for their systems. In China, around 72% of the population with access to the internet experiences low-quality connection, often due to low access to bandwidth, unstable data connections and the ISP barrier (commonly know problem in China that refers to the poor inter-connectivity between ISPs). Because of this problem, hundreds of millions of chinese people have started using technologies supporting offline downloading of large files Li et al. (2015). In this case, the offline downloads happens by users requesting large downloads from a proxy, typically with fast and reliable network. The proxy downloads the file, and the user can fetch the file at any time, when their connectivity is stable.

CSIRO is an Australian federal government agency, responsible for scientific research apps. In a paper on how well their field data collection was gathered from two of their apps, *the CSIRO Surveyor (Post Bushfire House Surveyor)* in Australia and *DroidFarmer* in parts of Africa, a review of their offline capabilities was conducted. Both systems have some sort of offline functionality to prevent data loss where internet is unstable or non existent. While the CSIRO Surveyor uses a cache to backup data in case of a mobile internet breakdown, DroidFarmer uses a offline storage, and later synchronises the data with the server. Both of the systems was well received by their target audience, and their offline data collection methods have a huge impact for gathering data where internet is not accessible Lane et al. (2015).

There is also an extensive use of offline functionality in health information systems. The health sector often work with data that is critical then and there. If HIS have to rely on internet connection to be able to fill out forms it could lead to detrimental consequences. When implementing DHIS2 in Ghana, their offline solution was based on storing data from forms in the browser cache, and upload it when connections where stable. They found, however, that there was not sufficient storage space in the browser cache to store all the forms. This lead them to disregarding the idea all together. As mentioned in 2.2, DHIS2 in Kenya implemented offline support in order to meet the challenges regarding poor internet coverage in the country. Later, this was adopted by other countries, such as Ghana, by copying what Kenya had done (Poppe (2012)).

2.3.3 HISP and DHIS

Background

Health Information System Program (HISP) is a global Health Information organisation aiming to strengthen Health Information Systems in developing countries. The organisation started up in South Africa in the 1990's, and has since then spread to a number of African and Asian countries. The University of Oslo (UiO) is one of the leading organisations within HISP, contributing in various projects including in-country capacity building and implementation support and research. This master's thesis will be included as work UiO does in contribution to HISP.

District Health Information System (DHIS2) is an open source, web based health management Information system controlled and developed by HISP. It is a platform for collecting, managing, analysing and using health data in developing countries and has become a

global standard in international development. With it being the national health information system in 67 countries it has a huge impact on how developing countries manage their information systems. It is also utilised by organisations like Doctors Without Borders, International Medical Corps to manage their routine data (*DHIS2 Fact sheet* (n.d.)).

Levels of design

The DHIS2 software development operates on two levels of design, one with a global objective and one for local implementation and customisation. Li & Nielsen (2019) refers to these levels as *generic-level design* and *implementation-level design*, which we have described in the sections below.

Generic-level design In a general context, generic-level design refers to design and development of the generic parts of a system. The DHIS2 platform core is the generic part of DHIS2, and is developed by a group of *core developers*, mainly situated in Oslo, Norway. The software they are creating aims to support a vast amount of requirements and use-cases, so it can fit implementation purposes in as many contexts as possible. The idea is to have a configurable software core that lays the foundation for a local fitting and implementation of the software.

Implementation-level design Implementation-level design refers to design and development during implementation of the generic software and how the it is built and modified to support local needs. This is often done on site, by anyone that wanna implement DHIS2 into their software. The design and development processes can be similar to in-house development, but with a 'core' system to support the development from the start.

Figure 2.3 illustrates how the two levels of design complement each other, where the 'core developers' make a generic system to support an implementation by HISP India.

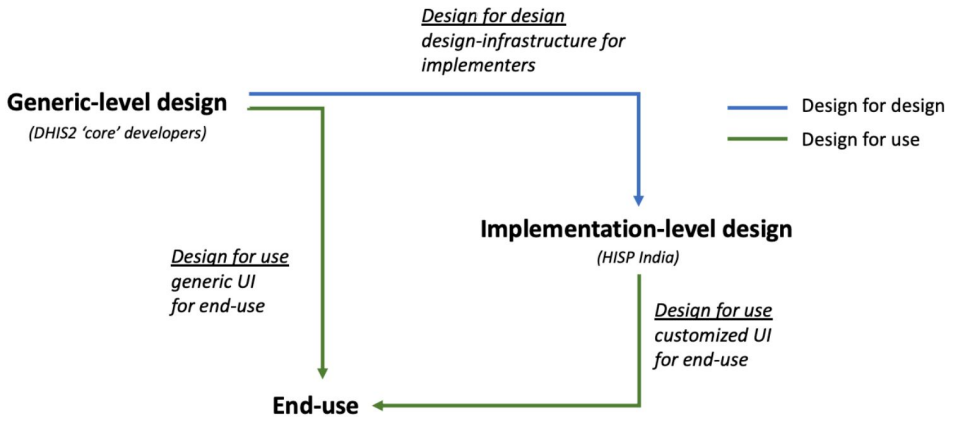

Figure 2.3: Multi-level design

Data modules

DHIS2 provides a variety of data modules for different ways to create and change the DHIS2 systems. You can create customised data elements to be used where ever you want. And by using your meta-data model you can generate data entry forms based on the model automatically or modify them to fit specifically to your needs. The gathered data can later be aggregated and visualised in various ways. The most notable data modules are the Tracker, the Event Capture and aggregation.

Tracker The DHIS2 Tracker is an extensions of the DHIS2 platform. It is made up of the same design principles as DHIS2, which means it is based on a generic data model, with the possibility of metadata configuration via a user interface. The Tracker's main functionality is to capture and store information about individuals, track them over time using unique identifiers and produce statics on single entities or aggregate data on a larger data collection. DHIS2 Tracker was made as a way to share clinical data across different health facilities in a simple manner. It's purpose is to be a basic and easy to set up transactional system, rather than being a fully functional EMR system. Instead of being this advanced EMR system on it's own, it wants to take on a complementary role in the system.

Event Capture The Event Capture lets the user register an event in a given time and place. What is referred to as an event in DHIS2 is a single data enquiry, e.g. a survey. The Event Capture does not track an entity, like Tracker does. The events are associated with a program (e.g ART treatment), an organisation unit and a specific date and time for when the event occurred. It consists of a set of data points like, e.g name, age and some measurements. The Event Capture also works offline, that is to say, if the user loses internet connection, the event can still be captured. The data is then stored locally and uploaded and stored on the server once connection is achieved.

Aggregate Besides from gathering data, through enrolment and monitoring of patients, mapping of diseases, etc, DHIS2 is used to aggregate data. It uses raw data sets, gathered from DHIS2 data entries, or from external sources that has been entered into the system. The aggregated data can be displayed in a large variety of ways. Thematically, by grouping the data based on topics, such as HIV/AIDS, vaccine etc. Timelines, visualising trends over time. Geographical, grouping data based on location, e.g region, district or national level. Lastly DHIS2 provides a number of different presentations of the data, like maps and GIS, tables, reports and graphs.

Chapter 3

Method

3.1 Access to case

The case study we conducted in Malawi is a contribution to the HISP design lab at the University of Oslo (UiO). The design lab is the headquarters of HISP and responsible for development and research in regard to the DHIS2 platform. The lab houses PhD candidates in the field of computer science and have tight affiliations with institutions and organisations in the developing world, with which they frequently exchange sabbatical eligibles and students' abroad studies. The work that is done by these connections account for research toward further development of the DHIS2 platform.

In our case we established contact with a PhD student at the lab which had a good understanding of which institutions that were available. These included an organisation in Delhi, India, a HISP team established in Dar Es Salaam, Tanzania and the Chancellor college in Zomba, Malawi. At that time the team leader in Malawi had planned a new project which focused on subjects interesting to our field of study and so we planned prerequisites for a case study at the college.

The HISP team in Malawi is centred around the Chancellor College in Zomba. This is also where we were stationed during our stay. Through a facilitator (later known as FA-01) at the campus, we got access to local developers and their workflows, to learn about the different ways they work and what they put emphasis on. For our thesis he placed us in the team working on digitisation of the countries health passports (See Figure 4.2. Through the college we also got to meet the end users both on campus and in the field. Usually, one of the facilitators (FA-01 was in charge of workshops and meetings with end users on campus. Another facilitator (FA-02) arranged the field trips to go to the health facilities and look at how they use, or don't use, computer systems to help patients and store data. So through these facilitators we unlocked a sea of information we could use in our thesis.

3.2 Action Research

Action research is a research method that centres around collaboration between researchers and practitioners. Bryman & Bell (2011) describes it as ”an approach in which the action researcher and a client collaborate in the diagnosis of the problem and in the development of a solution based on the diagnosis”. The purpose of this type of research is to use people involved in a social situation and try to change the existing situation to the better (Meyer (2001)). Reason & Bradbury (2008) also points out that action research is about ”creating new forms of understanding, since action without reflection and understanding is blind, just as theory without action is meaningless”.

3.2.1 Action research in our study

Meyer (2001) states that the strength of action research comes from its focus on creating solutions to practical problems as well as its ability to empower practitioners, getting them to engage with research and the subsequent development or implementation activities. This is the main reason why we find action research to be especially applicable to our study. In a given HISP project, local empowerment is a key factor for success and by involving in their work we can help with just that. By conducting action research, the researcher can also benefit from personal relations with the other actors. Since the local actors are most likely well known with the local context, a close collaboration will essentially help validate the research.

3.3 Data Collection

3.3.1 Data source categorisation

While collecting data, we have been in contact with numerous people. For convenience of the readers and ourselves we have made codes to help recognise different data sources. The codes are listed below:

- LD-xx: Local developer
- FA-xx: Facilitator
- NU-xx: Nurse
- HSA-xx: Health Surveillance Assistant
- HISPD-xx: HISP developer in Norway.

xx will be replaced by a number when used. I.e LD-01.

These categories are made up by how we view the different actors and it is based on what type of role they had when interacting with them. This means that their role in our study might not be the role they play in their work, but rather our perception of it. We mention this because we observed that the projects in Malawi had very diffusely organised themselves. This was most likely due to the fact that the majority of the project members

consisted of students at the university. Students tend to have an unstable schedule which can result in high turnover for the projects. Roles in the projects will be affected by this.

A facilitator refers to an authority figure in HISP Malawi, which is responsible for decisions regarding what is to be done, with who and when. I.e when to conduct a workshop.

A health surveillance assistant is anyone working with the patient or the HIS treating the patient. A nurse is also a HSA, but we have a different code for nurse since these were the ones we meet in the workshops described later.

3.3.2 Documents

As part of the HISP lab and earlier development processes there have been written a number of articles and master thesis' on the subject of Health Information Systems and related topics of interest to the DHIS2 platform. All this information is available free of charge for anyone to use. We used these articles and papers to get a brief overview of what the HISP lab and HISP team in Oslo is working on and which topics are interesting for further development. We quickly understood that making usable generic software was a big issue and consideration in every update of the platform. The core team is trying to create a platform that can be implemented everywhere and into any situation. The definite majority of the articles written about the platform had the assumption of using the platform in a HIS capacity but some of the more recent articles discussed the possibility of implementing the system into other arenas. One master thesis based on field work in Ghana did research on the possibility of utilising the system i the educational sector. The motivation of creating a more generic platform has thus only increased the past years, but yet facilitating for implementation-level design (see 5.3 from Li & Nielsen (2019)).

In this process we where introduced to different questions regarding large platform development and the issue of planning the platform governance.

3.3.3 Observations

Workshops in Oslo

Prior to our field work in Malawi we where invited to participate in two workshops at UiO with the HISP team, lead by HISPD-01. At these workshops we where introduced to prior and current research done for the DHIS2 platform. Students that had already been on their field work, shared their experiences and students and PhD candidates discussed their field of interest and possible research problems. During the workshops we discussed in plenum the potential research problems introduced and tried to formulate our own interests and expectations about our research. It was meant as a open discussion for everyone's benefit.

Meetings

Much of our data was gathered through unstructured meetings. These meetings included different actors based on it's agendas. We had the meetings either alone with the developers or together with both developers and a facilitator.

Even though the meetings we had throughout our stay had different agendas, they were mostly planned the same way. We often got informed one day ahead of time with when

and where to meet, and what the purpose of the meeting was. The meetings was often very loosely planned and we frequently experienced change of time, place and agenda. The time aspect was the hardest to get used to, since it is so different from the way time is handled in Norway. It became apparent after a while that the team members, and the culture as a whole for that matter, had a very casual relationship to time. A typical situation was getting told to meet at a certain time to start working, while in reality, the meeting did not start before one hour past the scheduled time. The meetings was often about planning. We planned how to work, i.e who does what, but before this we often planned metadata and design decisions so everyone was on board. In DHIS2, metadata is a important aspect of the system and has to be designed carefully before work on the system can be started. This was the theme in the majority of the meetings with the other developers. We discussed how to set up the metadata properly, which data points we needed and distributed work assignments. The dynamic was often that the most experienced developer knew how to go forward, and would assign different tasks to the others. When design decisions was concerned, the facilitator was often there to distribute assignments.

Throughout our stay, we participated in around 20 of these types of meetings. The facilitator would be present in about half of them, while the rest was developers only. The meetings usually lasted three to four hours, and gradually transitioned into work sessions.

Workshops in Malawi

As mentioned in Section 3.1, our facilitators had contacts in the health sector which they frequently used for design purposes. One of the things they would do was to invite nurses to the campus to participate in design workshops. We decided to call them casual workshops because of the seemingly lack of agenda. Despite the fact that the agendas were vague, the workshops were surprisingly effective. Their method was simple; let the nurses control every design decision being made, because they are the expert users. Since they know what they like and dislike with the current systems, they should know how the best possible system will look like.

The workshops included several activities. First, FA-01 brought up different systems with similar or same aims of our system and we discussed and pointed out strengths and weaknesses in these systems. This was mostly to get inspired for the next phase of the workshop. We proceeded to draw individual prototypes of different parts of the system, chosen by the facilitator. This was quick sessions, to get main ideas on the table. The nurses was also active in the prototyping, drawing their own version of the system. After the prototyping, we gathered around to compare drawings. In this phase, the nurse was the main actor. They had the power to tell us what they liked and disliked about the different designs and to pick out what they wanted to work with. Even though they had the power, everyone was trying to explain the different advantages of their designs. This was because the nurses did not have too much experience and might miss important details about the design that made it good. As an example, when designing the front page the nurse wanted a small list view of all the different pages you could choose from while a lot of important and heavy functionality was going to be on the front page. At first she didn't understand our design decision (this was the design we eventually chose, and can be viewed in Figure 4.6), and why it lacked functionality. After explaining why we wanted to separate functionality and have a simple dashboard with easy access to everything, the nurse agreed and went for

our approach.

During our stay we participated in three of these workshops and they lasted the whole day, mainly because the nurses took the day of work to participate.

Interviews with nurses

Often in combination with the workshops, we had sessions of conversations with the nurses. We call these interviews mostly because we were supposed to ask them questions and they answer, even though they felt closer to casual conversations than interviews. While the design decisions were mostly done in the workshops, the interviews was more about them and how they worked on a daily basis. Even though we asked questions together with the developers for the sake of the system, we also asked questions for the sake of our thesis. This was also encouraged by our facilitator during the workshops and the interviews. We asked them how they approached and enrolled their patients, and how the sent them to the different wards, to try and understand how the data flow worked on the different clinics. The conversations were usually after the workshops and lasted no more than 30 minutes.

Interview with facilitator

After getting home from the field trip, we processed the data we had gathered in Malawi. While doing this, we wrote down data we felt we lacked, so that we could have an interview with the facilitator over Zoom (a video and voice chat application). This interview lasted about one and a half hours, where the facilitator filled us in on the missing data. The interview was transcribed and later used in various parts of findings and discussion.

Field trip

Arranging workshops and interviews with the nurses was a way to get concrete feedback and make prototypes based on this. Moving on, the facilitator wanted us to go on a field trip with the local developers to a health facility. Before we could go however, we had to get approval from the facility which took nearly a week.

On the day of the field trip, we had a meeting with NU-02, LD-01 and FA-02 on campus before all five headed to the facility. At the meeting we discussed what we expected from the field trip and what we were there to observe. The field trip took us to a health facility near the college. While we were there we had two agendas, following the action research approach. As a part of the local development team we were there to observe, ask questions and take notes and pictures concerning the system we were creating. In addition to this, we made notes and observations on how the facilitators and local developers wanted to use these field trips to gather information.

When we arrived we got a tour of the facility, and NU-02 introduced us to HSA-01. He showed us the different wards, and we were able to ask him questions. In the different wards, he introduced us to the people working there, and let us sit down with them to observe. Here we got to talk to the nurses and HSAs operating the systems as well as the patients getting enrolled in them. Some nurses and patients didn't speak English, so a lot of what was said had to be translated by FA-02. Most of the wards didn't have

functioning computers, resulting in having to use paper based solutions. Other wards, such as the antiretroviral therapy (ART) ward, had all the computer equipment the facility could support because of the high demand of this ward. Since they used computers the most in ART, we conducted most of our observations in there, by sitting next to HSA-02 and HSA-03 while they took care of their patients.

3.3.4 Prototyping

Our main objective on behalf of the HISP team in Malawi, was to participate in the creation of a prototype for a health information system. Prior to this, the team had two researchers work on an information accumulation project, mHealth4Afrika (4.1.5), and learnt how the DHIS2 web API worked. Our system was supposed to replace the mHealth4Afrika system as a prototype for the coming DHIS2 system in Malawi.

Action research development

Working on the prototype, we only used the development methods preferred by the HISP Malawi. This included how we defined the requirements and how we planned the implementation. While we tried to give as much input as possible on the prototype itself and it's design, we tried not to interfere with which work methodology we used.

Methodology and implementation

The methodology we practised was similar to agile development in the way that they decided requirements and changes to the requirements. Requirements was frequently changed depending on input from end users, but there were never any sprints or other kinds of time frames for when functionality should be done. The way it was handled felt more casual. When implementing we had a lot of freedom on the design and code structure. After some implementation we were supposed to show the work we had done to the team and end users, which refined the requirements ones more.

3.4 Data analysis

This thesis is primarily made up of two cornerstones, the literature review in Chapter 2 and the case in Chapter 4. Before the field trip to Malawi, we had already done research on topics we found interesting and added it to the literature review. While we were in Malawi, we looked for data on the subjects we had written about to get own data to compare it with. As an example, one of the subjects we wrote a lot about before the field trip was generic development, and how to do local changes on a global generic software. Since we went down to Malawi, which were a local branch of the global organisation HISP, we tried to observe similarities and differences in what we wrote and what we saw.

This also worked the other way around. In Malawi we gathered unforeseen data, which we never had thought about writing in the literature review. Since we have nothing to compare our findings with, we had to go back and find papers on these subjects. As an example, we found that the Malawian HISP team would like to use low code principles

when coding the system. This in turn meant that we had to go back to the literature review and add information about it.

There were also a few times we found data on the subjects we had written about already. Even though it was comparable data, it was never one-to-one. This meant we had to go back to both sources, pick out similarities and discuss it. These topics include agile development. In the literature we had written some very general sections on agile development, and in Malawi we got to see some very unique practises of it.

We wrote notes about what we experienced every day in Malawi. These notes were scribbled down in the field and later reviewed and refined. They were used for various things, since we follow the action research method. Some was used to gather requirements for the application, and other notes were written as observations on how the team in Malawi worked.

When collecting qualitative data as we do in this thesis, through meetings, workshops, field trips, etc. one must be aware that personal biases exist. As mentioned by Walsham (2006), the background, experience and prejudices we have, will influence how we make sense of the data we gather through our research. When gathering data in a country so different from our own, it is important to understand this. In 4.1.1 we clarify how the situation as-is in Malawi, and try to make the setting of our field trip as transparent as possible.

Much of the data analysis was done after the field work, back in Norway. We found it easier to make sense of the bigger picture doing it this way, contrary to analysing the data as we got it. You can make an argument for analysing data when it's fresh in memory, but we rather trusted in our ability to review and refine it for later use.

3.4.1 Data validation

To help us evaluate our own data collecting we decided to adopt some of the principles for interpretive field studies described by Klein and Myers (1999).

The fundamental principle of the hermeneutic circle suggests, according to Klein and Myers, "that we come to understand a complex whole from preconceptions about the meanings of its parts and their interrelationships". Furthermore, in a interpretive study contexts, *parts* can be viewed as the researcher's and participant's preliminary understandings. *The whole*, in turn, can be understood as the shared meanings emerging from interactions between us as researchers and everyone we have interacted with. This means everyone ranging from HISP Norway, to HISP Malawi and the health workers there.

The principle of interaction between the researcher(s) and the subjects concerns how researchers recognise the participants as interpreters and analysts. As Klein and Myers puts it, "Participants are interpreters as they alter their horizons by the appropriation of concepts used by IS researchers, consultants, vendors, and other parties interacting with them, and they are analysts in so far as their actions are altered by their changed horizons". In our study we have practised action research, and by that unconsciously interfered with the participants horizons regarding the software development process and design choices.

The principle of dialogical reasoning involves understanding of how preconceptions from preliminary research can influence the data that emerges from the field study. As Klein and Meyers points out, "the intellectual basis of the research design provides the

lenses through which field data are construed, documented, and organised". As we discussed above, one of our data collection strategies was to look for data we already covered prior to the field trip. It is important to understand that in a hermeneutic circle the point is to iterate between the theoretical preconceptions and the findings to try and modify the original research design to fit with the findings. The data we collected helped us refine our preliminary research by adding and removing different topics. As an example, we wrote about user centred development in the preliminary research. We used information about other field trips where findings revealed a lack of user centred development. Then we wrote theory about what we thought contrasts how they do it in Malawi. As you will read in Chapter 4, this was not the case, which meant we had to go back and refine what we wanted to use the user centred development theory to say.

Chapter 4

Case

Our case study is situated in Zomba, Malawi. We were invited to gather data on how a local HISP team's work methodology is, identify requirements needed to evolve the HISP software portfolio and subsequently help implement these.

In a research capacity we were invited to be a part of a development team that had a new system delivery aimed at improving the local HIS in Zomba. We were expected to take part in the development process, but not intervene with the plan as we aimed to improve on the process on a higher level. We were given the main responsibility of the design choices for the system, so the local HISP team could benefit from our experience in the interaction design field.

4.1 Background

4.1.1 State of the art, Malawi

Africa is in general considered less developed than many other parts of the world. It has had it's issues over the years stretching from colonisation and resource exploitation by European countries, to civil wars and bad management. Malawi is no exception. Malawi is a relatively small country in central Africa (Figure 4.1). It has a population of around 18 million, where only about one million live in the capital Lilongwe, and 700 000 in the next biggest city, Blantyre. This means that the effective population density at any location is very low. The majority of the population live in smaller villages all over the country, mainly involved in agriculture. Health clinics are often associated with bigger cities, meaning that people usually have to travel far for medical treatment. Malawi has only recently lost their status as the poorest country in the world, now the fourth poorest, where 70.3% of the population live under poverty line of 1.9$ a day, according to World Bank Data. The low income means that the vast majority can't afford any transportation which they can use to get to hospitals or clinics, thus health visits happen rarely. People are seldom registered in the system, checked and are uneducated when it comes to health. Because of the lack of sexual education in the country, as of 2016, approximately 1 million

people (9,2% of the population) are diagnosed with HIV/AIDS. Being among the poorest countries in the world Malawi gets a lot of aid from other countries and organisations such as US AID, UNICEF, The Global Fund, Bill & Melinda gates foundation. These resources are used to give food to the poor, build hospitals, schools and fund research and development of HIS.

Figure 4.1: Malawi location

Zomba is the former Capital of Malawi, but is today considered a smaller city with just above 100 000 citizens. The city holds the biggest educational institution in the country, the Chancellor College, which is also the biggest college ground of the University of Malawi, having more then 4 500 students. In addition to regular education activity, the College acts as a hub for HISP in Malawi. A local team is sanctioned to research and develop HIS demanded by the Health ministry and health organisations present in the region. The local HISP team has tight bonds with the University of Oslo (UiO), where the current HISP Malawi team leader has written his PhD and helped other Malawian students with exchange programs. We will refer to the local development team as the *HISP team* throughout our thesis, to make it easier for the reader. It is however, important to mention that the local team only utilises HISP products where it is needed, often complemented with other open source software, like OpenMRS.

Health system The health system is comprised of several *programs* which focus on specific health issues, e.g. *TB program* monitor and treat patients suffering from tuberculosis. Different programs have different kinds of Health Surveillance Assistants (HSAs), which conduct patient visits to the clinic, monitoring and treating of the health issue associated with each program. A visit is logged using generalised forms which the HSAs fill out per visit. Small clinics function as local help to their immediate surroundings and the programs offered vary from clinic to clinic. Bigger institutions, like general hospitals, are localised in the cities and offer all the programs and specialists for different health issues.

One program which is common for all the clinics is the outpatient registry (OPR). The OPR program is designed to diagnose and treat patients that don't need to be admitted into the hospital for overnight care. This is generally the most used program any clinic offers.

4.1.2 Personal identity

An important aspect of life in Malawi, and in general in Africa, is that many people are not familiar with the concept of personal identification. Malawi don't have any formal identification papers, passports are rare so the use of identification papers are small. The health system is based upon *Health Passports* (Figure 4.2), which briefly identifies a persons medical history. They are purchased at the clinics by patients not carrying one already and is meant to store all the patients visits and prior treatments. It is however, only a book, often lost and rarely updated properly by the Health Surveillance Assistants (HSA), so for a health worker to meet a patient without a health passport or any other identification paper is not uncommon.

Figure 4.2: Health Passport

The clinics keeps files on site of their patients, but as mentioned above, getting to clinics can be tough. Occasionally, babies are born outside of the clinics, leaving no history if not registered later on. The clinics don't share their files with other clinics either, so getting a positive ID and the patients' history is a rare luxury for HSAs. All in all it is a chaotic and complex system. Identifying the patient you are seeing as a HSA is very important as diagnoses vary greatly dependent on family history, previous visits, age, etc. Additionally, the experience of nurses vary and are generally low, which mean that they often diagnose on the basis of simple signs. They usually avoid thorough examination as well, because of lack of resources and capacity. To illustrate how big an issue this is we will give an example from the visit to one of the clinics (3.3.3).

One patient came in for consultation for the ART program, going through the baobab system and process described in Section 4.1.3. The patient was not registered in the system and had to be enrolled to proceed. It is worth mentioning that HSA-01 later told us that she most likely had been enrolled before, but since they had no identification or record it

was just as easy to enrol her once more. To be able to enrol her, HSA-01 needed her name, place of birth, date of birth and some family history. Her last name was one of the most common ones in Malawi, many share the last name of different old tribes or important state figures. Her place of birth she descried as a small village with no name. HSA-01 had to ask some follow-up questions and try to guess which village it was. After some time he managed to do so, and asked for her date of birth. She did not know how old she was, only that she was born in *the dry season*, which indicates the period between mid-May and mid-August.

Being able to get any identification from a patient with those prerequisites is an almost impossible task. The name and place of birth was not too important at this point, but she would have to be correctly identified for her next visit to the clinic. Having a common name and no place of birth is not a very easy identifier. The more immediate problem was her age, because treatment at the ART program takes her age into account, not to mention being important for doctors to diagnose her later. To solve the immediate problem however, HSA-01 asked about important events during her life time, such as presidents, elections, football world cups, etc. Say that she remembered the 2010 football world cup and began school the following term she must have been around five or six which means she was born in 2004 or 2005.

Resolution The government is currently working with a solution to this problem. Citizens are encouraged to get unique ID papers, but it will take time. Currently, 80% of the population have registered and have their unique ID papers yet the clinics have not taken it into use because it is not effective if only 80% of the population has it. The process is ongoing and anyone can register at centres in the bigger cities. HSAs need to expect everyone to be a part of the system before we can use it to develop solutions that depend on it.

4.1.3 Baobab HIS

Another system currently in use is developed by the Baobab Health Organisation. This system is divided into programs as mentioned above and aims to manage and improve the health system on a national level. One of Baobabs' program projects is an ART program, for diagnosis and treatment of HIV/AIDS. LD-02 from our team had been part of this project and had hands on experience with the system. The system is integrated onto touchscreens (Figure 4.3), which trained HSAs use to record visits to a clinic. It is a fairly new system, which aims to replace the old ART patient cards (see Appendix A).

Figure 4.3: Baobab system

The ART program consisted of a number of stages which effectively digitised one ART patient card entry. Instead of having to fill out paper forms, store them and having to physically find the forms each time the patient revisited, the system did many of these things for HSAs.

To depict how the system works, we will go through how one ART visit was conducted. One room in the clinic was allocated to ART related visits. In this room, there were three nurses and two other HSAs, trained in the Baobab system. Five patients at a time entered the ART examination room and presented their Health Passports (Figure 4.2) to one of the nurses together with a registration number unique for each patient. While the first nurse logged the entry in a large ledger by hand, the other found a plastic folder in a large cupboard (Figure 4.5), which had pages with stickers representing earlier visits. The plastic folder and passport was then placed in a pile working as a queue for the HSAs working with the touchscreens. The HSAs then used the unique identifier to find the patient in the system, and started logging the visit. The system resembled one column in the ART patient card (Figure 4.4) and asked for the information in each row in separate stages. Each stage had its own page with a next button at the bottom. When the HSA was finished with all the information a machine printed out a sticker containing all the relevant information of the visit which was glued to a free space on the paper in the plastic folder. After this the patient was directed to a nurse and received their medication. The plastic

folder was then stored back in the cupboard and the HSA was free to help a new patient.

Visit Date *day month year*	Hgt *cm*	Wt *kg*	Adverse Outcome	Outcome Date	ART Regimen *Adult Formulations*	Side Effects (Current) Specify 'Other' in Notes	TB Status (Curr.)* Suspected No | Yes	Confirmed noRx | Rx	Pill Count	Doses Missed	ARVs Given No. of tablets To	CPT No. of tablets	Family Plan Depo given	No. of condom	Months on ART	Viral Load Sample taken	Result	Next Appointment
Jan			D Def Stop TO		1A 2A 3A 4A 5A 6A 7A 8A Oth	No Pn Hp Sk Lip Oth	N Y	C Rx			P G					Bled		
Feb			D Def Stop TO		1A 2A 3A 4A 5A 6A 7A 8A Oth	No Pn Hp Sk Lip Oth	N Y	C Rx			P G					Bled		
Mar			D Def Stop TO		1A 2A 3A 4A 5A 6A 7A 8A Oth	No Pn Hp Sk Lip Oth	N Y	C Rx			P G					Bled		
Apr			D Def Stop TO		1A 2A 3A 4A 5A 6A 7A 8A Oth	No Pn Hp Sk Lip Oth	N Y	C Rx			P G					Bled		

Figure 4.4: ART patient card entry

The program resembles heavily the paper based ART patient cards in that it asks for the same information in each entry point on the cards. As mentioned above, each stage in the system resembles one or several rows in the table of entries in the patient cards. The program in its whole represent one whole column (Figure 4.4).

Figure 4.5: ART patient card records

As you can see from the figure above (Figure 4.5) the system is definitely not fully digitised yet. The system stores patient information but doesn't contain all the previous visits. As we understood the system, reports of aggregated statistics are generated on demand. Since this takes a lot of time, it typically takes months between each demand. At this point technicians and nurses gather the ledgers and records stored in the cupboard and enters this into the DHIS2 system to create reports which say something about the HIV and AIDS situation. Thus the potential of the system is not fully utilised.

Interviews with the technicians showed a positive excitement for the system with some obvious points of improvement. The system had no *back* function which meant that if an error was made on the last stage the technician would have to restart the system by pulling out the power cord and putting it back in again. This was a time consuming affair. The hardware also raised issues. The touchscreens where robust and fast, which also meant expensive. The technicians told us that for every touchscreen functioning, there where two out of order, because they didn't have the money to repair them when they went out of order. While strolling through the different wards we saw this first hand, touchscreens just piled in corners, not in use.

4.1.4 HISP organisation and governance

The local team in Malawi is structured in an informal manner, where the role is often more dependent on experience and seniority of each individual setting rather then clear predefined roles in an overall capacity. The development team is also subject to structural changes where people are brought in based on specific knowledge and expertise. During our time in Malawi, there was little way of knowing which members where participating in each activity. Below, we will discuss the different roles involved in the project.

Product contractor The team works as implementers of the DHIS2 platform. Each implementation have a contractor which require HISP Malawi's local experience and expertise. This contractor is often large health organisations that require data tracking software to help the local population. The system we were replacing, mHealth4Afrika which we will discuss further down in Section 4.1.5, was contracted and funded by European Commission (EU) under Horizon 2020. They where a big stakeholder in the development process, and had substantial governance over the design. At the preliminary design and concept meetings for that system, they expressed important aspects which then was conceptualised to form the system they finished.

In interviews with FA-01 and LD-01, which participated on the mHealth4Afrika project, we discovered that there are often many contractors and funders of the system. This often leads to negative political complications as all the funders have their own take on how the system should look and behave. In the particular example of mHealth4Afrika, the contractors had so many different opinions and requirements that the system became too broad for Malawi to use. Interviews also revealed an unbalance of money contributions between the contractors which had a positive correlation to the power they had over the system. Politics regarding each contractors' level of governance contributed to difficult design planning.

Partnering entities Some projects are based on a collaboration between several HISP entities across countries. The mHealth4Afrika for instance, is a collaboration between the University of Malawi, Nelson Mandela University in South Africa, Strathmore University in Kenya, University of Gondar in Ethiopia and the University of Oslo in Norway. Additionally, developers from Ireland and Turkey was involved from International Information Management Corporation Ltd (IIMC) and Software Research, Development and Consultancy Corp. This organisational outline is a very difficult environment for development. Many people are involved and the governance distribution is unclear. In addition as the

main development team was in Ireland, but all the end-users are in Africa, design choices was often made difficult and on wrong assumptions.

Team leader The Malawi HISP team has a team leader (FA-01), which organises meetings and activities for the rest of the development team. FA-01 is a a senior member of the HISP organisation and a high ranking faculty member which act as the main organiser for local development. The local HISP meetings mentioned above (3.3.3) was always organised by FA-01, weather he was attending or not, and he made all arrangements for field trips, development and interviews together with FA-02. The team leader had great respect in the meetings, and had a higher level of governance when it came to the process and design. Being a high ranking faculty member he also had governance in regard to the College. This helped our processes as we got to use more of the college resources available.

One aspect to consider is that this team leader is an academic. From interviews with FA-01, we understood that this is normal as the HISP teams are often affiliated with local Universities. Our experience is that the recent academic background contributes to good processes through field knowledge.

Developers As the team we worked with is stationed at Chancellor College, a university in Zomba, it did not come as a surprise that most of the developers were students from the university. The developers were involved with HISP and DHIS2, either because of own interests in the system, or because it was a part of their thesis. Nevertheless, one of the problems with having students as developers is the large turnover. A students free time is quite unpredictable and varies from year to year, depending on how many classes they enrol in, etc. In addition to this, students eventually graduate, and most likely move on to pursue other work. As far as governance is concerned, the local developers is granted a lot of freedom in their development. Everything from language and frameworks, to dependencies and initial design proposals, is completely in their hands. The moment where the difference in governance is evident, is on the organising level. The developers has little to do with organising meetings, interviews with users or what is to be changed after these meetings and interviews.

Expert users One of the main sources of functionality and design decisions came from something the team called an *expert user*. Seemingly, the expert users were the end users of the system being made. In our case, implementing a health system for use at health centres, our end users were nurses and other health personnel, and therefore our expert users. Different nurses would get invited to participate in workshop sessions and informal interviews. With the title ”expert user”, came the power to influence and even control what were to be added, modified and removed from the system requirements. The development team had programmed parts of the system and explained different design decisions, but it all came down to whether the nurses agreed or not.

The HISP Malawi organisation raises many important question related to governance distribution in large platform systems. In light of new additions to the platform, specifi-

cally the Web API (4.1.5), how is the system governance affected? This is something we will take a closer look at in the discussion section.

4.1.5 DHIS2 web API

Shortly before we where introduced to the DHIS2 system, the core team had worked on implementing a Web API for DHIS2 implementations. Prior to the Web API you had to base all implementations on the DHIS2 dashboard functionality which is very limited. It solves specific problems for statistical report purposes and data management. In interviews with the local developers, they said that they ”often had to *hack* the intended structure to meet the system requirements”. If it couldn't be done, they had to request new applications from the core team. For instance, the developers wanted to use the Adaptive-Object Model (AOM) (2.2.1) to auto fill the different input fields in the forms. This meant that the system had to talk to the web API and ask for the fields for different form inputs like personals or ART treatment. However, when the meta data the forms were based on were made, they were sorted alphabetically and not by the order they were made. This meant that when asking for the data from the API, there was no way to automatically sort the data so that the forms were correctly sorted in the system. To *hack* this, we had to create the meta data with numbers in front of their names to sort them (i.e. ”1 - First name”, ”2 - Gender”, etc), and then strip the names of their numbers in the code. The easy fix to this would be to add a field in the meta data creation, linking each item to a number, and when confronting the local developers with this they told us it was not possible to request such small changes to the core team at UiO. In the local developers experience, this was often not prioritised in favour to more critical functionality. The requested application would therefore take too long to develop for the team to use in the current development process. Additionally, the application made by the core team needed to be developed with a broader intention so it could be fitted elsewhere. The application was in many cases not specific enough and couldn't solve the requested needs. To mitigate this huge bureaucratic problem, the local developers requested a Web API. Developers can use the API to create systems with a much better user experience and without having to wait as everything can be developed locally with the use of React and Node. It is still a new addition to the DHIS2 platform but so far it seems that the local developers are pleased with new addition. A test project was initialised by the University of Malawi in collaboration with the local HISP team which showed good results. Our field work is an addition to research which aim to improve the platform, and specifically the API.

Two years prior to our arrival the local HISP team developed a HIS using the API. This was called mHealth4Afrika.

mHealth4Afrika

mHealth4Afrika is a collaborative research and innovation project, aimed at ensuring correct data tracking among other things. Before this project, the health clinics in many remote villages have kept their records by physical logs which challenges the validity and relevance of information. The problem also being that there are many different programs that each patient is a part of and need to be registered for. At a national level, this sometimes mean that patients are duplicated in different programs and the statistics get inflated.

The project is a collaboration between HISP teams in Malawi, South Africa, Kenya and Ethiopia. The main goal of the project was to make the data collection process digitised. This would mean that health clinics all over the partnering countries could utilise the same tool to track and update data and be sure that the information at hand was up to date with recent events. The application is meant to track events that follow a clinical visit and keep track of the information gathered. When the developers initially made the use cases and evaluated the user stories, the DHIS2 Tracker application was considered, but concluded to be inadequate for the application as the users required a custom outlook of the forms. The developers needed a *path sensitive* form that did not require to much writing, but rather a series of multiple choice questions followed up by questions sensitive to earlier answers. This could not be done in DHIS2 Tracker without *tricking* and *hacks*, so the team decided on making a new application from scratch using the DHIS2 API. This way they had more freedom to design the page as they wanted. The application was targeted at Health Surveillance Assistants (HSAs) that work at these clinics, logging events every day. They needed an easy to use, fast application that could be used locally at any clinic. They also needed it to work offline as the system is expected to encounter bad internet services at times. The application was developed in Angular, tested and released.

The local developers experienced a problem in regard to the funding of this project. The contractors of the project had not anticipated upkeep costs of the system after implementation. As the contractors didn't have a computer science background or sufficient experience during development processes, the upkeep costs was not taken into account. The whole of the budget was spent on development, which led to a big problem when it was implemented and expected to function in the field. In interviews with the developers, they had the same experience from a number of other projects they had worked on. It most often led to the system being outdated or even never useful as the funding ran out before the whole system could be finished.

4.2 The problem

The expectations of what the problems could be, we got from our guidance teacher and HISPD-01. These were of course only parts of the problem, as the two teams have very different motivations. The HISP team in Norway are the ones developing the platform and wanted us to do research on how the platform could be improved upon, while the local team in Malawi wanted research on a local implementation using the platform.

HISP Norway The workshops (3.3.3) introduced us to a number of potential research problems. One of the research problems presented read ”What aspects determine whether something generic could be usable and relevant across implementation (i.e., what can be global, and what needs to be local?)”. This question is no new consideration when developing large generic software as DHIS2 and Google, Apple and Amazon have all their strategies on how this is best done within their domain. The question in regard to DHIS2 however, raises some important questions and problems that are not as easily solved as many of the issues of the platforms mentioned before. DHIS2 is a health based service, which has some sensitive and specific problems. The question of ”what can be global” helped us realise that it don't need to be. Global, in any case, is way to generic for any HIS in our opinion. Local systems narrows the perspective down, but can be too closed minded when considering the benefit of sharing information and systems with other *local* implementations. We wondered then if it needed to be as black and white as global or local implementations. The ”D” in DHIS2 stands for district, and can really mean anything considering who the system is meant for, which countries, region or continent. Our question in regard to this was not what aspects determined weather something could be cross implemented, but who had the knowledge and connections to find out if it was? Who had governance? Who had interest in implementing across other implementations? What was their incentive to do so? At this point we only made more questions for ourselves, but it helped us get a grasp of what we were going to write about, what problems we could face, what questions we could get the answers to. One of the PhD candidates at UiO was heavily invested the question of governance, with regard to HISP teams worldwide, though he focused mainly on projects and organisations in India, where they do things very differently. This also interested us at this point and we wanted to find out more specific to Malawi.

The experiences from other parallel master thesis was helpful in making some expectations about the work we were set out to do. Prior to these workshops, we had tried reading *state of the art* and getting equated with large platform systems, HIS and generic programming. The preliminary study period (3.3.2) before these workshops, we experienced as a tedious process, but necessary to get an overview before conducting the field work. As the field work period was restricted by many factors, we were dependent on making the most of the time we had, and therefore dependent on preliminary knowledge to guide our data collection process. The method itself of doing action research development was new to us, but having it presented to us beforehand helped us understand how we should work when we arrived in Malawi. Specifically the exercise of noting and logging the process seemed difficult. When participating in the development process of a project you are not used to note how the work is being done, rather just focusing on doing the work. FA-01 however, had experience with this method and reminded us on how to work.

HISP Malawi Following the action research development process and observing the local team's needs and wants regarding the new system, we found the largest problem to be with regard to the collection of HIS in Malawi. The Malawian HIS is divided into different *programs*, which is typically a sort of clinic which intend to solve specific health problems, e.g ART programs intends to monitor and treat HIV and AIDS patients while TB programs intends to monitor and treat patients suffering from tuberculosis. Until now, the HIS consisted of many separate systems that intended to help specific programs, e.g Baobab system acted as a system for the ART program. Optimally, the local developers wanted a system which incorporated all the programs into one. An underlying problem to this was the identity issue Malawian citizens face. What to associate each program's entry to is not very obvious when you don't have a system containing any associations. For data to be relevant it needs to be associated to time and relevance, which in the case of HIS is a patient that needs to be associated with a unique identity. This was something that wasn't immediately obvious to us as it is an unknown problem in our world but something that needed to be addressed before we could continue in Malawi. Another big concern regarded the availability of the system. The old DHIS2 system did not support responsive data downloads, you had to request the whole data set with no possibility of limiting search, which is of course a huge problem in a country where internet is scarce and unreliable. This was only the functional problems however, because funding for these systems came from separate organisations acting as *product contractors* to the local developers. The organisations had their interests often specific to one of the programs and would therefore have greater governance when it came to decision making in regard to the system specifications.

So Malawi wanted a system to digitise the health passports, incorporating all programs, keeping patient records and history up to date and available to all. This was not a task which could be solved with the old DHIS2 system as it was. Design governance was almost completely limited to the HISP Norway core team. The new Web API did, however, present new possibilities for developers, distributing governance, making it possible to personalise local implementations.

4.3 Prototype work

4.3.1 Development process

This section is meant to give an overview of our development process experience in Malawi and give an overview of the workflow of HISP Malawi.

1. Preliminary phase

As we were invited to participate in the development process we, as well as the local team, needed to understand what we were going to implement. The development process started from scratch which was initiated by a preliminary phase intended to uncover system preferences.

Activity The team started by reviewing mHealth4Afrikas' preferences and from there arranged a workshop with the new development team where we would discuss what we wanted to keep and what we wanted to add. We got a very brief overview of how the system was going to look and what work we needed to do to uncover the rest of the user preferences.

Result The system should optimally replace the old health passport system in Malawi. This is an old system where each citizen visiting a health clinic gets a health passport which they use to keep their health records. At the first visit the patient fills out a general section which adds some general family history and personal information and at each visit the patient updates the notes section of the passport. These health passport is kept by each user and brought to the next return for a clinical visit.

A frequent problem is that these passports are lost and not regularly updated. Additionally they only gather general visit information and do not carry any specific program information. The specific program information is done in separate forms (e.g ART form from Appendix) that are kept at each clinic and very rarely shared between other clinics.

2. Design research

After concluding the preliminary phase and getting a general overview of the functionality of the system, we needed to review some design choices. To help us with this, the local team arranged a workshop together with a nurse from one of the nearby clinics.

Activity Over the course of one afternoon, a nurse participated in our design workshop. We started with a simple interview about her daily workflow at the clinic and continued with a brainstorm workshop. We included the nurse in the whole workshop and encouraged her to give feedback on all our design discussions. Each team member, including the nurse, made a small paper prototype which we presented in plenum and commented on.

Result Having the nurse participate in our activities helped us swiftly come to a set of design concepts. All ideas where instantly tested and reviewed by a user and the time used was very productive. In other projects we have experienced that even a lean approach to the design phase is somewhat tedious. The development team involve the end-user in the development process by interviews, being a tedious process on its own, reviewing the interview, discussing design approaches only within the team, testing their theories on a new set of interviewees and doing the same process all over again until the team has a viable approach. HISP Malawi have adopted a much more casual approach to this process, by inviting the user to participate in the actual process, not only partially through interviews.

3. Design prototype

After the activities of the second phase we had a very rough sketch of how the system should look. The local team still wanted to make a prototype which we could use to show users and ask for input.

Activity We started directly with a partial prototype created in React. This prototype was created over the course of the next three weeks. We only had the sketches for a couple of main features, but tried to make the components generic and create new features as we went along. The prototype was built on the Adaptive Object-Model (2.2.1) pattern without any real metadata to be based on. The metadata was in parallel built by other team members.

Result We created a somewhat complicated minimum viable product (MVP), without any real connections to the metadata. We created our own dummy data, which could be replaced with real data queries. The MVP was then showed and discussed with a local nurse which gave us some pointers on what she liked and disliked.

4.4 Findings

4.4.1 Ease of use

Opting for click functionality In interviews and workshops the expert users, NU-01 expressed her main concern to be that systems often require to much typing, and are therefore not user friendly. Problem being that the health personnel have a hectic workload and ”are lazy” in NU-01s’ words. Local Health Surveillance Assistants’ (HSA) experience with computers are on average very low, which suggests favouring clickable interactions. During our stay in Malawi we scheduled several interviews with developers and nurses, which all revealed the same conclusion. LD-02, that have taken part in earlier projects, such as the Baobab HIS (4.1.3) and mHealth4Afrika (4.1.5), commented that ”at the request from expert users we opted for clickable functionality” mainly rationalising it by speeding up processes that earlier had to be manually typed. The nurses also saw the benefit of ”restricting functionality” by restricting use and letting the system validate the information the HSA typed into the program.

In our system, we had a request to hide functionality from the users. Our system meant to mitigate the problem of connecting data to the right patient and therefore we needed to have the patient registered in the system. At launch, the databases for the system would not have registered users and the nurses would have to populate the database. This, however, revealed an interesting issue we would not have foreseen. The nurses wanted to hide the ”Register Patient” functionality from the main page.

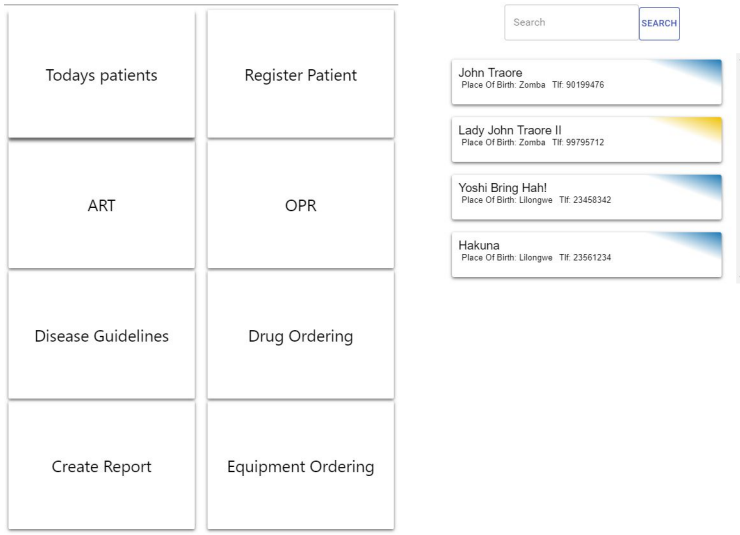

Figure 4.6: System main page

As shown in Figure 4.6, we instinctively placed the button in the main view so it would be easily accessible, being a very frequent functionality just after launch. NU-01 disagreed with this and meant that ”the HSAs would just register new patients every time they where

helping patients" disregarding the search functionality as the system intended. NU-01 argued that "the HSAs are lazy and will just register new patients", whilst we argued that the true lazy thing to do is to search for the patient in the database and not have to fill in a whole form every time. Most likely, the HSAs felt safer by just registering a new patient each time, as they have been trained to do so. Introduction to a search field can be stressful, especially with the ongoing identification issue.

Opting for click functionality can also be viewed as *not opting for typing*. This can create problems when implementing search for patients, or typing in a medical comment. It can be mitigated by trying to anticipate what the user is wanting to do. For instance, the search functionality in our system, located to the right in Figure 4.6, should try to further fill out the patients' name or restrict the search in the background by location or common clinic databases. This is a subtle functionality, and helps increase the usability and speed of the system. In our short time span, we did not get to implement any resembling feature, but tried to solve it in other ways. By clicking the *Today's patients* button from Figure 4.7, the user gets a list of all enrolled patients at the clinic. We understood that during a patient's visit the patient sees many different HSAs, nurses and doctors which all add on visit information. Therefore it should be easier to find enrolled patients than random patients in the system, to reduce the pool of searchable patients.

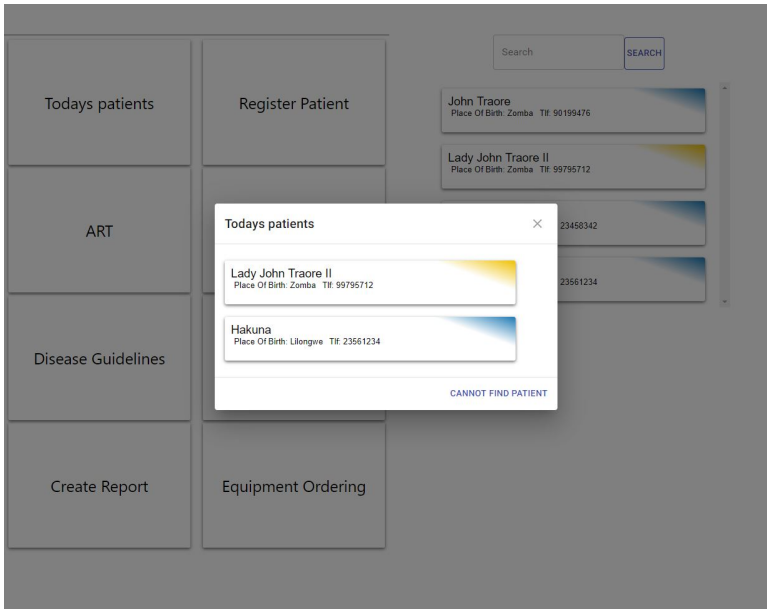

Figure 4.7: Active patients

Familiar design One experience from the prototype work (3.3.4) was that all development opted to copy the already known systems that were in place. Our work aimed to digitise the forms and health passports (Figure 4.2). The team already had experience from the

Baobab project, where they more or less directly copied the ART forms (see Appendix). From this experience, and interviews with the HSAs using the system at the clinics, we concluded that this was an important design choice to consider and maintain in our work. It is not hard to imagine the benefits of the system resembling the already in place solution, but there is of course some concerns by blindly copying old systems as well. As we understood from the developers, projects usually have those kind of premises. They are rarely asked to create new and innovative systems, which using cutting edge design. The systems opt for familiarity, which in the case of Malawi are some basic forms like the ART form (see Appendix). This also became apparent in the workshops, when the expert users always chose the designs resembling the system they used. We often interfered, asking them ”well, how about this” pointing at new design proposals. Their response was often ”this is similar to what we use now”, while pointing at the resembling design.

Though the time constraint hindered finishing the prototype, we got to implement a basic Adaptive Object-Model which populated forms based on metadata. Creating the logic for accumulative data is one of the main features of the DHIS2 application today, and based on the structure set we tried letting the system interpret and populate the forms associated with the different programs. You can read more about this in Section 4.4.2. The colour coding of each patient (top right of each patient card in Figure 4.7) is also meant to give a familiar feel to the HSA at each clinic. They represent the different colours of the individual Health Passports (Figure 4.2), yellow for women, blue for men and orange for children.

Multiple paths Another finding was that there should be multiple ways of doing a given task, as the interviews revealed that workflow for each nurse was unspecified. We wanted the nurses to be able to choose how they wanted to solve a given task. This also aimed to get the nurses more involved in the system and increase ownership over it. A positive bi product when implementing this is shallow paths to all points. A challenge when developing these different workflows is that all paths lead to a point where data need to be relevant to the situation in the workflow chain. In our prototype you could access a patients' ART program information either by going to ART from the home page or the patients page.

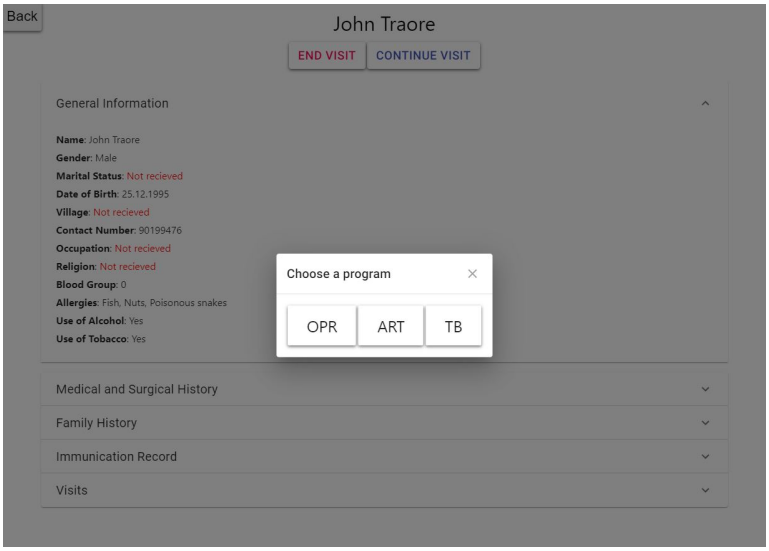

Figure 4.8: Continue visit

When the patient has been registered as an active patient the health workers can access the continue visit feature (Figure 4.8). This feature saves the patients current data, e.g from check in, and makes it easy for any ward to find the patient and continue to register information for the visit.

At either of these points the situation calls for the same relevant information regarding the patient as the data system need to link to the relevant patient. A session is what we called the time from when a patient enrols in to a visit, to the time he/she ends the visit. From the patient page this information can easily be transferred through to the session, but from the home page a patient is not selected and therefore need to be chosen before the health worker can continue. In the process of creating this feature we struggled with making it clear for the health workers the idea of a session. We experimented with selecting a active patient for the current session, but feedback from interviews and simple testing discarded the feature. The experience only increased the lessons learned from opting for familiar design.

Responsive design As we understood from the interviews with the expert users and developers there was no obvious technology platform for which the system was best suited

for. Clinics share no equipment standards and the staffs' technical experience differed greatly. Additionally, the system needed to be implemented onto a platform which was easily available. The Baobab system used very expensive touch screens, which were hard to fix and update to keep working. Computers were not broadly available and internet connection unreliable. We had to make the system responsive to fit *any* platform, but opted for mobile first design. This decision was based on interviews with the NU-01, which worked at the clinic we visited on our field trip (3.3.3), and assumptions made about the general health clinic situation in Malawi. NU-01 wanted to use her smart phone to interact with the system as the clinic in question didn't have access to computers. She also argued that this is the situation in many of the clinics and many nurses own a smartphone. It is also a good idea for making it easily accessible and mobile for all users.

4.4.2 Designing for metadata

When starting the development process we, assumed the system preferences to be much of the same as what was defined on its previous iteration. This lead to a very shortened preliminary phase and the team decided after a short period to start development, just after the first couple of workshops. Also, rushed by our time constraint, we eagerly started development without looking over to much of the developing standards. We made the decision of defining the fields required on each page within the code. After understanding that the system would be presented in different ways based on its' location, we had to rethink how the system was being rendered. Following the principles of Adaptive-Object Model (2.2.1), we had to define metadata to be interpreted at run-time. The metadata wasn't defined yet, but creating a mock data structure for illustration and development was sufficient for our purposes.

Figure 4.9: Register patient

In Figure 4.9 and 4.10, we have used the same elements and render functions to populate the forms based on the relevant metadata. The metadata is structured based on *programs*, associated with *tracked entities*, which can accumulate *data entries* for different purposes. We realise this is not a very detailed description, but for this chapter we consider it sufficient to understand the concepts. The register patient form (Figure 4.9) creates a tracked entity associated with programs, like the ART program. The ART program is made up of initial information and a recurring data entry consisting of a number of stages.

These represent the entries shown below (Figure 4.10). We were only partially introduced to the concepts specific to DHIS2, but we managed to make logic of it in a short manner of time. It was from this experience we realised the power of metadata design and the huge advantage of creating an API which can implement these ideas, not only to make the system design flexible but also also the governance distribution flexible. We will go into further detail in the discussion below.

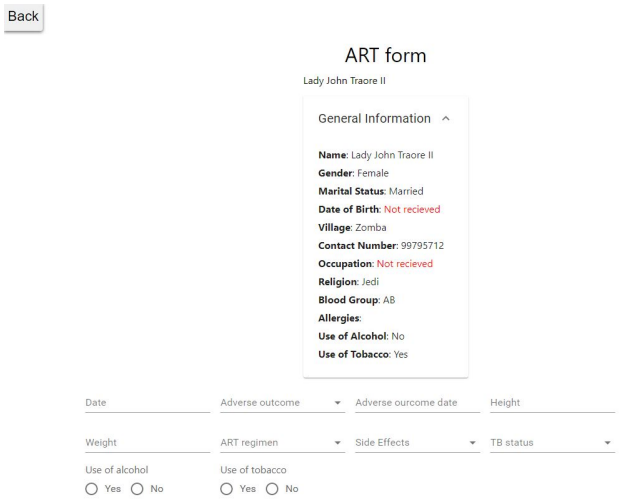

Figure 4.10: ART patient form

Chapter 5

Discussion

5.1 Research questions

1. Designing for an effective evolutionary platform

   (a) *Economies of scale: Scalability vs. specialisation.* Scale comes at the price of specialisation and developers need to make considerations of global or local development. How does the focus on scalability effect the evolutionary dynamics of the platform?

   (b) *How to make platform development agile?* Usually platform development follow a waterfall development process and choices are made at the beginning of the development process rather than on the go. We acknowledge that this impair evolutionary dynamics and see how certain techniques mitigate the issue through;

   - Making partitions?
   - Splitting design into implementation-level and generic-level?

   (c) *How governance effect evolutionary dynamics?* Distributing the decision-making authority to a broader set of stakeholders often encourage specialisation.

The concept of economies of scale describes cost advantages, enterprises obtain due to their scale of operation. As their operation grows the minimal cost of producing one more unit is lowered due to the already developed infrastructure etc. This is also true for generic software development, as the initial cost of development is often the most significant one, and maintenance cost would be relatively lower per unit the bigger the reach of the system is. Making a system fit numerous purposes means more sales and more users, leading to decreased development cost per user. However, this require that the software is inherently generic, so it can fit into as many use cases as possible. In user centred design (2.1) developers consider the end-user to influence design choices. It makes systems more relevant for the intended audience and increase usability traits such as efficiency,

effectiveness and user-experience. The process of user centred design means specialised solutions for specific users. These two concepts, generically and user centred, are both very important for enterprises, but they also appear contradicting. It is paradoxical to have a solution that fits all, and at the same time fit for specific use. Platform-based software ecosystems are often sought after for this purpose. They exhibit an inherently generic core for scalability and individual modules for specific cases. It is emerging as a dominant model for these purposes and the success stories are many, including Apple's App store and Mozilla Firefox's extensions. Platform development divides the development and design processes into two separate aspects. One for maintaining the generic traits and one concerning the specialised modules. Li & Nielsen (2019) have developed a framework which name this separation appropriately. They argue that there are different levels of design, concerning local and global aspects, which they have named *Implementation-level design* and *Generic-level design* accordingly. In this chapter we will discuss our findings with regards to implementation- and generic-level design.

5.2 Implementation-level design

5.2.1 Patient Centred Design

Health Information Systems (HIS) are often centred around a patient entity. When the users of HIS, which are health personnel, interact with patients, they usually need access to the patient's records and manage to survey information about a diagnosis. To make these systems function like intended, information need to be associated with a given patient. This require extensive planning when designing the system, to correctly identify and assign the right information to the right patient. The HIS we experienced in Malawi disregarded this consideration. Their HIS consisted of a number of very separate entities that solved specific and very local problems. For instance, the Baobab system (4.1.3) functioned as a digitised form, printing out a copy of the examination, which then was stored locally at the clinics. It didn't consider making data accessible for other systems or incorporate other systems' data. It's purpose seemed to be making the current forms slightly more effective to process through. In our opinion the designers had not considered the impact the system had on the environment. Wever et al. (2008) states that "the way users interact with a product may strongly influence the environmental impact of a product", and argues that developers have a responsibility to keep in mind sustainable outcomes of the use of their systems. He makes a good point about the responsibility the designers over users of a system. This way of handling the examinations was not a sustainable solution. So why did the systems we encounter ignore the apparent immense potential of creating a connected, collective of HIS?

The health system in Malawi is made up of a number of programs which help monitor and treat distinct health problems. The ART program mentioned earlier is one of them, and intends to monitor and treat HIV and AIDS patients. The TB program is another, which intends to monitor and treat patients suffering from tuberculosis. The programs gather data at different clinics, about specific observations related to the program. For every program the patients are registered the same way, but with no link between the programs. As an example, the same patient can be a part of the ART program and the maternity

program, which means that they are both HIV positive and pregnant. The connection between the two programs indicates that precautions should to be taken. However, because of the program separation in the Baobab system, this conclusions cannot be drawn. When designing this system the developers have paid little to no attention to how the programs they create can act together in a pool of programs. We label this approach as *Program Centred Design*, because the separate programs is in focus. Following generic design principles (2.2), we tried to find generic traits in different components of the system. We identified that the programs should be more coherent in regards to the patients they serve, and suggest an alternative approach. We call this approach *Patient Centred Design*, with an aim to associate all data entries to a patient instead of a program. This approach was explored while developing our prototype (3.3.4), and found that the core generic trait of the HIS was the patients. All programs need to be associated with a patient, which means a patient package or module can be developed generically to fit all programs.

During our prototype work our task was to create a system that could act as a collective HIS. With limited time resources we focused on the design principles for the system using the new Web API. As mentioned above we put the patient in the centre and associated all data entries to a patient utilising the Tracker model provided by the DHIS2 platform (2.3.3). Therefore, one of the first things we developed was the patient page (Figure 5.1).

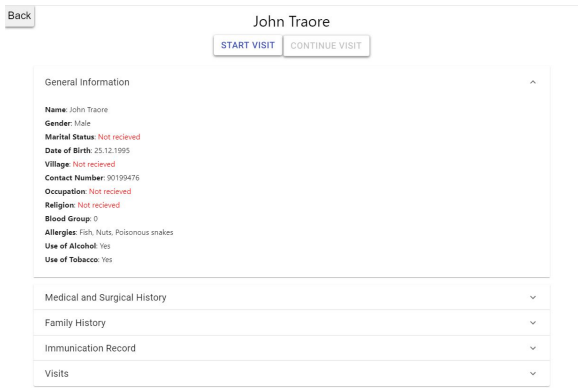

Figure 5.1: Patient page

This page represent a unique patient and is meant to be viewed whenever a patient visits a clinic. Our idea was to replicate the health passport and associate all visits with a patient. This is a simple idea, which is why we had a problem understanding why this approach weren't thought of before. Why didn't Baobab create a system that benefited the collective system and made it easy to associate data with patients? Similar issues arose when prototyping in Malawi. In interviews with nurses they told us that "HSAs will not bother with searching for a patient, if they can just register a new one". They argued that the reason was HSAs were "too lazy" to search for the patients. One could argue what is more lazy, but this was their perception. As we mentioned in Section 4.4.1, we initially had to hide the register patient button because of this, but we still wanted to find out why it seemed like none wanted to bother with associating a patient to different programs and

visits.

Unique associations A reason this might be the case is linked to a problem we had with data associations. New data entries could hardly be associated with any relevant data entities as there didn't exist any. A prerequisite for data to be associated with a data entity is that the data entity is unique. Data can be linked to a unique clinic, hospital, village or in this case a patient. When the underlying information describing the data entity is not unique this is not possible to do. From the personal identity situation in Malawi we described a scenario where a patient tried to explain who she were and where she came from (4.1.2). She gave a name, but in Malawi there are often many with the exact same name from the same village. She did not know her day of birth, but only knew how old she were. Her village was very small and didn't have a name, they more or less just called it home. For the HSA enrolling this patient, it was an impossible task to uniquely identify her. In our system we wanted to create patient records associated with the patient, but how could we identify the right patient? How could they know if a person named "John Traore" was the same as the patient "John Traore" (from Figure 5.1) found by searching his name?

Lack of time meant that we didn't get all the answers we where after. We got to prototyping and used it to understand how they developed systems in Malawi using DHIS2. The overall experience is very positive and we are confident that the developers at Chancellor College are well suited to develop systems using textbook methods and theories. It was their idea to interview HSAs, include them in workshops and visit them at the clinics, all of which reflects a thought out user centred development process. The issues we encountered was with the ground work for identification which hinders the process of creating a unified system. Work on solving the identity problem have already began however, and according to the local developers, 80% of the population have already been registered into a new identification system. The goal is to supply every citizen with a unique ID numbers and identification papers. This will have huge impact on the current HIS, as registered citizens will be easy to identify using the ID. One of the main issues will now be to create a relevant system suited for the information available today, but at the same time make the system modifiable for the future identification situation.

What we argue to be the best solution will be difficult to implement for several reasons, one being that it takes time before it *can* be implemented, because of the ID roll out. Another important thing to consider is that the technicians at the clinics are not used to the idea of tracking data entities or creating data associations. The technicians have been trained in doing a repetitive task, like filling out forms when presented with a patient. It is important that the end users are on board with the changes to the system. As mentioned, today's system are *program centred* and doesn't concern about the value of the information it gatherers. And as Wever et al. (2008) was quoted earlier, "the way users interact with a product may strongly influence the environmental impact of a product".

5.2.2 HSA centred development

User centred design (2.1) aim to improve user experience of the system by making sure the system fits intended use. This means specialising the system for specific users and specific

situations. If the system has additional goals of achieving economies of scale, it must cover a broad set of users and use cases. Commonly, user centred design is complemented by an agile development process that continuously evaluate the direction of the development process. In practice this means that the development team ask the users what they need, then develops, tests, evaluates it, before starting on a new iteration of the same process. Generic software however, is only used if implementations of the same system will share the same traits, i.e parts of the system are common in all implementations. Therefore, implementing generic software need to be considered before the development process can start. If the generic traits was discovered at a later stage of development, the likelihood of it effecting already implemented features would be high. Pollock et al. (2007) refers to *the birth of a package*, as he argues that "there are many choices influencing the extent to which the package will become 'generic'". What organisational practices are common and what the organisation want to generalise needs to be considered at "the birth" of the packages. The challenge is making a plan in an environment that is constantly changing.

Even though we know that a new system is on its way to ensure that everyone can be uniquely identified, Malawi cannot wait for a HIS. It needs one which works in the current situation, for the current users. The users of the system are the Health Surveillance Assistants (HSA). They are using Health Passports (4.2), which represent the patient's entire medical history in a small booklet, both as a record storage, and as a encyclopedia to look up potential risk groups and prior illnesses the patients may have. Our prototype work (3.3.4) consisted of digitising the booklet as a mobile friendly system with the same functionality, such as recording and looking up information about the patient. This task was assigned due to a combination of the local teams research on user needs prior to our arrival, and our own research and findings. On the same note as the current system, our prototype had to find a solution to the problem of identification, for the current environment situation. After interviewing nurses and experiencing the situation first hand at the clinics, we came up with a temporary solution that could work until the national ID was fully enforced. The idea was to use the amount of information that existed and combine it into unique identifiers. The patients would hand over as much personal information they have to the HSAs enrolling them, typically by handing them their health passport. The HSAs would keep feeding information into the system until it had enough to uniquely identify the person. For example, name and date of birth might not be enough, because two or more people could have the same name and date of birth. The HSA would proceed by filling in parents name or village, until only one person was left in the search results. This approach would also mitigate the problems with citizens not knowing their date of birth, as the HSAs could just work with the information the patient actually knew.

The system we imagine would base its entirety on the Health Passports, since the information of one that has been duly updated is actually very comprehensive. It contains general history, family history, past medical and surgical history, family planning for women, growth chart for children, vitamin supplementation and more. The main body of the booklet however, is a set of blank pages where summaries of previous visits are logged. These logs are written by the HSAs at the different wards at the clinic, and finalised at the outpatient registry (OPR) with a diagnosis or treatment. OPR is, same as the general health information, a generic part of all the systems, while the programs vary from

clinic to clinic depending on what services they provide. Being able to add programs on demand is what enables a platform to scale in its environment which represents the next stage in our suggested development process (Figure 5.2). This includes adding individual modules to complement the system, like App Store does for the Apple iOS or extensions does for Mozilla Firefox to name a few.

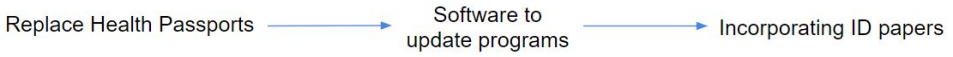

Figure 5.2: Process

Having the system dynamically add programs that are needed means that the system can stay relevant even in a changing environment. The environment the system is to be implemented in is expected to change, and new programs are needed periodically. At the time of writing this thesis, we are experiencing the outbreak of Covid-19 which represent a new program that needs to be implemented into the system. Adding this today would mean a development team would have to create a whole new system which would take time to develop. Handling outbreaks like Covid-19 and Ebola requires swift action, and the programs needed are most likely very similar to one of the other forms that already exists. We suggest following the principles of designing for metadata as discussed in 4.4.2. The programs should follow principles of *continuity* to make the system feel complete and familiar resembling each other. By basing the data on metadata editable in the DHIS2 dashboard a new program can swiftly be added to the system without to much work. In our prototype work (4.4.2) we tried implementing this idea, and it works as a good example of how it can be done.

Availability As we mentioned in 4.4, we opted for a mobile first, responsive design. This was on request from the local developers and had come up as a user preference before our arrival. We confirmed this whilst observing the current system and concluded that the current machines reduced availability. The machines running Baobab (Figure 4.3) often broke down and was expensive to repair. They were also expensive to buy in the first place, which had led to a lack of machines on the clinics. Mobiles and tablets on the other hand, are cheaper and much more available to the HSAs. Therefore, in the process of HSA centred development this was in our opinion the best option to increase availability.

In one of the first workshops we attended, we did a walk through of the mHealth4Afrika system where the developers pointed out what was lacking and could be improved upon. This system had to download the whole data set before it rendered, making it slow to load in. It also meant that the system was heavily internet depended. First hand experience, and research on the topic (2.3.2), told us that internet was expensive and lacked proper availability and speed. In our prototype work we had no time to consider this aspect of the system, but it is an important requirement developing further. In our prototype work we had no time to consider this aspect of the system, but it is an important requirement developing further. If availability should by improven and then maintained, the system need to function with an infrequent access to internet connection. Offline capabilities are therefore important to consider.

Offline capabilities As the individual systems now function, data is accumulated on paper or local data sources, which are on a later time fed into the DHIS2 platform (see **??**). This is a very time consuming process, because the HSAs need to type in the information a second time after gathering it at the clinics. One of the reasons that this is not done at the initial time of data gathering, is the internet problem discussed above. Since internet infrastructure is not expected to change drastically, any future system need to acknowledge this issue. Offline capabilities need to be implemented so the system can gather information as normal, but store the data in the background, and post it when an internet connection is established. Another issue with the scarcity of internet is that store data wouldn't be accessible, like searching for patients and viewing patient's prior records.

Today, they gather information and store them in hard disks or paper forms, but this information is not accessible to the HSAs phone. Today, they gather information and store them in hard disks or paper forms, but this information is not accessible to the HSAs phone. What can be accessed however, is wireless signals. Creating a Local Area Network (LAN)- connected to hard disks on-site which holds all patient information could make the HSAs jobs much easier. They would not need internet at work, and they would not need to access a limited number of expensive machines to register visits. Rather, they could connect to the LAN and update the local hard drives. The hard drives would then need to be connected to the internet occasionally, and feed the DHIS2 database.

Informal participation in development processes

Earlier we have touched in on the subject of user involvement in Malawi. We experienced their involvement to be very informal in nature with no clear tasks when asked to participate in development activities. In our experience the process is usually very distant to the user following formal interviews and careful analysis of the data. In Malawi however, HSAs were asked to meet at the college in the morning and participate in a whole day of planning activities. The experience from Malawi was positive, as the nurses had time to formulate and explain their ideas more carefully by participating in the design process directly.

A question was raised while working in this fashion, about how much inclusion of HSAs was healthy for the system and how to balance the efficiency and desires they bring, with common design standards. Interviews revealed that the developers usually use the HSAs that are most vocal and the ones concerned with particular aspects. Kushniruk points out that in large scale health information systems it can be hard assessing how representative the users aiding the design process are of the general user base. In Malawi alone, there will be thousands of users if the system are to be rolled out nation wide. Take the example where the nurse aiding us told us to hide the "Add new patient" button until after the nurses had used the search function. Was this really true for the majority of the HSAs? In the voice interview with FA-01, we were told that the user group for the workshops were usually broader than we experienced in Malawi, but it might not be as large as desired.

As stated by Pollock et al. (2007) generic traits must be considered at the birth of the package, which mean setting a plan for development. Planning before the development process means setting a field of focus and making boundaries. Boundaries come in forms

of less freedom of development and higher costs of changing system features later in the development process. User centred design is meant to increase users' governance but developing for generic traits contradictory reduce it by encouraging planning and complete solutions.

5.2.3 Designing for relevance

When developing user-centred modules it is important to consider the users' context. For the sake of this thesis, we have called this the systems' "relevance". Common approaches to system development evaluate the environment, finds a purpose and defines system boundaries. These boundaries are subject to a specific time and place, a context where it can be relevant. This approach is based on the assumption that the environment is stable and the contexts stay relevant. Garud et al. (2008) highlight a pragmatic approach to design, trying to mitigate the issue, in *Incomplete by Design and Designing for Incompleteness*. The article discuss the disadvantage or the pitfalls of a traditional scientific approach which "extols the virtues of completeness". They argue that the *conventional approach* is likely to run into problems in environments characterised by continual change, which is often the case in large platforms. "In such contexts, system boundaries are often unclear and user preferences are both heterogeneous and evolving". This is especially important to consider when the scope of the system is large and considering many environments for the system to stay relevant.

The development process we recommend is meant to ensure that the HIS is kept relevant in a period where the ID system is changing. How far the process has come is not only time dependent, but geographical as well. More urban areas, near the ID registration centres will most likely adopt the system faster. The system need to change concurrently with the degree of ID implementation in the immediate surroundings. While implementations in rolled out, there is also room to test how the system is working in the new environment. We are in a unique position in this project as we know for a fact that the environment is going to change and we know in what direction it is changing. This make "designing for relevance" an easier task if following a set of techniques that encourages dynamically modifiable systems.

Adaptive tools

Following principles of Adaptive Object-Models (AOM) from 2.2.1, moving the systems' data structures and design logic into a database results in a system that can quickly adapt to changing situations and can keep its relevance. The database can be accessed through tools specifically developed for the purpose of changing the metadata, and as mentioned by Yoder et al. (2001), "it also encourages the development of tools that allow decision-makers and administrators to introduce new products without programming and to make changes to their business model at runtime". Imagine the process recommended in Figure 5.2. At which point should the decision to change the system be made? As we have mention in 4.1.1, communication and situation awareness is limited to immediate surroundings and many communities might seem isolated. Who knows best how the HIS should function in these places? Surely the people who live in the situation knows best how to tackle it

and when to make changes. Therefore, having them make decisions on how their system should look like would be beneficial for everyone. It can result in less work for the local HISP teams and ”the power to customise the system is placed in the hands of those who have the business knowledge to do it effectively”, (Yoder et al. (2001)). In this scenario, the local HSAs are the decision makers. If they have the power to change the system when and how they please, they don't need to waste time waiting on the domain experts. In Africa, this might sound like an impossible task, but our experience from workshops with the nurses showed us that they are willing, and understanding of development processes. They showed us that they wanted to participate in the design choices and that their ideas was relevant to the systems' needs. This also brings up a point from Kujala, stating that users may need to be educated in certain design aspects before engaging with the design team. Lets recall the problem about nurses wanting to hide the registration button, so HSA workers that used the system wouldn't just add a new patient every time. The tools would then allow certified nurses to implement that function, and try it out for a period of time. Then they could evaluate and make a decision about what works best, completely without any help from programmers or domain experts. Maybe it was a right decision in the first phases of the new system, but not when everyone had been using it for a while?

The system governance would have been distributed to the last end of the chain, the users of the system. An important question however, is how much they should be able to control and change. Though the systems may look different from clinic to clinic, they are still gathering the same information which needs to fit into the common data scheme. The patient page still need to represent a patient, and so on. The example above is only one alteration of the system. Should everyone be able to alter the system to their pleasing? One could then argue that the HSA workers wanting the registration button, would just change the system to show it. Of course, there would be a need for some kind of ”expert users”, a panel or a group of people who represents the users preferences, chosen by either the community of workers, or by the leadership of the HIS.

Evolution of a system, in comparison to natural selection, happens when an alteration become favourable and later iterations, generations, adopt it. Having many different alterations, mutations, means that evolution can happen faster. It means that users can test solutions faster and do minor or major changes quickly. The system wont immediately be adopted by all users and it wouldn't work in all situations, as we have discussed above. Nevertheless, with the power of rapid alterations it could evolve without the need for specialists. When we discuss testing in this context we don't mean to compare it to the exhaustive testing often associated with development processes, but rather a very informal approach. Let the users that are going to use the system evolve it as they go. In this process, simple sharing of ideas and different implementations can be a very powerful tool. The users that is going to use the tools, regardless of how they are organised, will not be familiar with the new system. Getting input from other solutions would be very beneficial for each clinic to make their system.

Aiming for economies of scale means having a broad, *global* perspective. It requires an ecosystem which can facilitate many use cases and exhibit generic traits. However, a systems' *relevance* is in danger when planning for platform-based ecosystems. Implementations need to be specific for each use-case and be relevant *locally*. This means involving

users more extensively and one solution, following an agile approach, is giving them tools that closes the otherwise apparent technological knowledge gap. Designing for incompleteness (Garud et al. (2008)) in environments that are in constant rapid change.

In this chapter have discussed all the considerations in regard to the *local* aspect of an otherwise *global* platform ecosystem by focusing on implementation-level design. Next we are considering the global perspective, the generic-level design.

5.3 Generic-level design

Platform-based architecture is a choice made by the development team of a system. It should be carefully planned by the platform owners and architects before starting the process. They are responsible for designing the platform in such a way that users can create *modules* effectively for the end-users at present and future. Tiwana et al. (2010) present an idea that ”the evolutionary dynamics of platform-based ecosystems and their modules is influenced by the co-evolution of the choices of the platforms owners endogenous to the ecosystem and the environmental dynamics exogenous to the ecosystem”. They present what they call a ”tripartite co-evolution perspective” which considers platform design, platform governance and their environment separately. The previous section considers the environment and discuss a situation where the developers of each *module* distribute the governance. In this section we focus on how the platform itself can encourage distributed governance. The platform is in theory infinitely scalable as it can be incorporated into any major global, regional and domestic arena which need the accumulation of personal data, not only specific to the health domain. The platform have already started research on incorporation in education (the school system in Ghana), as well as agriculture. Additionally, research into other arenas (e.g police recording) is on its way. In this potentially unlimited ecosystem, how should the individual modules act? What unites the modules and can be conceptualised into rules for everyone? What can be generic? Who can use the modules?

First we need to evaluate how the platform should fit into the current HIS environment.

5.3.1 Platform fit

The Baobab system (4.1.3) and our prototype (3.3.4) exist within the ecosystem of the HIS in Malawi. They are used in individual situations, but share the accumulation of personal data and act in the same domain. DHIS2 is an open source platform and the packages created by the HISP core team is easily available for everyone through their websites. The individual modules of Baobab however, is not easily accessible. Development teams using DHIS2 is responsible for how the module act in the ecosystem and for making their solution available. There is no common directory to search for solutions or share ideas with other teams wanting to utilise the DHIS2 platform. It is therefore very hard to know, for a team wanting to start a project, if the solution already exist or if they can use modules and ideas from other projects.

The systems that we encountered often used DHIS2 as a form of report making system. They used the aggregate data model (2.3.3) to periodically collect and report health data. Each system collected their own separate data associated with each program and fed the DHIS2 platform individually with that data. Not all the programs had been digitised, but most of them reported some data to the platform. Using the aggregate model (2.3.3) to monitor an entire population was a time consuming process, as all entries needed to be summarised outside the system before it could be entered into the system. This solution stems from the same problem discussed in 5.2, that they lack the unique associations to use the Tracker model (2.3.3). For the same reason, it would be difficult to collect the data together, compare them and cross match the results. As an example, let’s say the World Health Organisation (WHO) wanted a cross match of how many have died in the

hands of tuberculous and COVID-19. Two individual solutions could not generate those associations. So, each module function individually, but how could this be improved?

Each module function individually in a domain where they collect their own data. For that purpose the modules have their own system to identify the patient they are monitoring. Remember the Baobab system (4.1.3), which prints out data on stickers and keep the data in huge ledgers. This system is in no way perfect, but it is a system. That system is not available for any other developers to use. It is created and used only for the ART program. The system could be implemented into the TB program or the OPR program using the same means to identify a patient. This was in large what the prototype (3.3.4) we were developing was going to solve, but our solution started from scratch. We, as well as the developers before us, made the assumption that it would be easier just to create our own system following a patient in addition to making systems for each program. But could it be solved with what was already available? As the systems use the same standards for data collection they fit into the same data scheme in DHIS2. This could be exploited by making tools that can fit into the current situation. Either by adopting one of the major modules and implementing the same solution into all the others, or by making one united solution and altering the existing systems to utilise the new instead of the old. With the new identification system on its way the latter might seem like the better solution, opting and planning for the new.

Creating a community Making a community, or a *store* where development teams can search, use and share ideas would mitigate the immediate problem before a generic solution could be implemented. Not only to solve the identification problem, but for design examples, form solutions, metadata interpretation to name a few. With the new Web API they could follow a package based development process, like Node packages, and import packages they needed for their purposes. Additionally, if the community followed open source principles they could even improve on solutions made by others.

5.3.2 Making the platform fit

Until now, we have discussed a top-down approach trying to fit the modules into the platform to create a collective generic HIS for Malawi using only the platform. Developers in Malawi however, are not exclusively developing HIS using DHIS2. There are a number of other services available such, as OpenMRS, which have many of the same features as DHIS2. Developers of HIS in developing countries, in similarity to developers in other fields, are contracted to solve a need and try doing so as effective as possible. They try finding services that fit into their platform. Sometimes this involves using DHIS2 services, but rarely it involves the whole of the system. Rather, developers pick and choose what they need and discard whatever does not fit or work in their situation. Sometimes DHIS2 solutions are the best and sometimes they are not, and in many cases is it not sustainable to only use DHIS2. A bottom-up approach, considering how the platform can fit into the modules is therefore an important aspect of the platform. Separate functions can be made available through an API functioning individually from one another. The concept of *black box* functions is one that is used when describing functions you feed information and get something else in return. The name let you imagine the function as a black box

you don't look into, but just get what you need from. In the same way, HISP can make their platform's functions work for the developers to use when creating modules, such as the Tracker data (2.3.3) functionality can be available through an API exploiting some of the strong sides with DHIS2 software.

HISP have already started working on an API for this function encouraging the use of *Docker* where their functions can work even using different code languages. What the HISP core team effectively does is move governance over to implementation-level design, from generic-level design, which moves governance closer to the users aware of the situation.

5.3.3 Adaptive abstraction-levels

When developing platform ecosystems it is important to consider the connecting solution. The focus on economies of scale can sometimes work against the system generalising to broadly making the solution vaguely fit all solutions, but not specific enough to be relevant to intended use. It can quickly function as a number of individual services rather then a connected platform which aim for a collective goal. Platform ecosystems are often very complex and planning how all future modules are going to fit is not possible. Developing countries are often in rapid change and in different stages of these changes based on location, meaning that generalising a solution is especially difficult. Some specialisation is therefore needed at higher levels then just implementation. Having logic that have more defining impact over the system means specialising solutions and indirectly keeping the systems' intentions clear and on point, not making the mistake of generalising too much.

In organisations there are hierarchies, and because of the lack of highly educated people in developing countries, one can argue that hierarchies can be strong on the higher levels. These structures are important for the business models to work, especially in large organisations or fields such as a health system. Where expertise is scarce, well established bureaucracies are important to make good and coherent decisions regarding the whole of the a system. For example, some decisions should be made by the chief of medicine to ensure coherent medicine practice, while other decisions should be made much closer to the patients to ensure effectiveness locally. This type of hierarchy also exist in a platform ecosystem. The core team have knowledge about the whole system and how it's focuses on a global scale. Then there are some with continental and cultural knowledge which have specific suggestions that should be heard in regard to their expert field. After this you can imagine someone having specific knowledge based on countries that can influence the system, continuing narrowing the abstraction level to regions or institutions and finally having someone that are familiar with the local aspects. Yoder et al. (2001) discuss an Adaptive-Object model example where he have distributed tools on different levels. These levels of abstraction represent levels of governance. At each point in the *chain*, specialisation is made to make the solution better fit the situation it is intended for.

It is also important to reevaluate the purpose every now and then as developers frequently do in agile development (2.1). In a grand ecosystem bound by choices made at the *birth of the system* (Pollock et al. (2007)) this might seem hard, but through Adaptive-Object models (2.2.1) and clever planning you can design for incompleteness and reevaluate the platform as you go. Having the platform logic be dependent on easily changeable variables at each level, reevaluation of an abstraction-level means change downward in the

system. And change follows on lower abstraction levels when decisions are made higher up.

5.3.4 Unified platform tools

Creating tools that distributes the governance have one significant downside which we did not discuss in full above. For a development team to create tools that can alter design and metadata takes time, time that developers don't really have as resources are limited. Therefore, developing such tools, even with the necessary knowledge and guidelines, is not very likely to be prioritised.

The goal of DHIS2 have always been to make a generic solution to HIS. What all these systems have in common is that they aim to help locals and that they are used by less educated users. Also, the situation they need to stay relevant in change constantly and very individually. Situations changing individually is a key phrase here. As we have seen with the health system in Malawi, the situation differ from district to district. What we have been getting at is that it is to broad for one team to create a system that can fit into all these situations, but the tools to alter them with might and should be considered to work generally. The problems each implementation face is individual, if not there wouldn't be a need for a new implementation, but the unsteady situation and the need to adapt is constant in all situations the DHIS2 platform is implemented. In addition to the large time consumption for development of such tools, the notion that we need them suggest we should standardise so development processes are unaffected. By having unified platform tools that can work globally, developers would have an easier task of collaborating.

5.3.5 Partitioning evolution

Information systems are dependent on evolving with its environments to stay relevant to the end users. Evolution is however not an obvious trait, but something that need to be considered and facilitated to be effective. Tiwana et al. (2010) argue that "more decomposable complex systems evolve faster because they require less time to evolve by recombination and will undergo more diverse evolutionary experiments." He uses the argument to illustrate the advantages of what he calls "Modular Systems Theory", which in practice is how a platform is made up, or usually the preferred architecture. The modules that are created using the core of a platform to compose smaller subsystems that can individually evolve, where change is easier because the impact is smaller and therefore experiments are easier to conduct. If a change that fits into the whole system would work for one subsystems, the chances are that it would work for other subsystems as well.

5.3.6 HIS politics

Health systems often have a number of benefactors. The funding required for many of the projects and the impact they have on the environment is not possible without a lot of resources and knowledge about the surroundings. Therefore, often the same organisations fund different large projects. They have their interests in the projects and because they are funding it, they also have a part in the governance. For HISP, this is also true.

The core development team in Oslo, generic-level designers, make changes to the platform on request from stakeholders which is done via *Jira*, a project management tool. The core team have limited resources and therefore need to prioritise changes to the system. HISP is a non-profit, open-source organisation that is funded by many of the biggest health organisations around the globe. Each of these organisations are large stakeholders in the platform because of their funding and therefore have governance of the platform. Each organisation have their own focus and request that the core developers focus on their needs. Funding is crucial for HISP and therefore keeping the stakeholders pleased is paramount. These same organisations are often the ones that fund different projects utilising local developers, implementation-level. In the projects' development process the organisations choose whatever platform they have already funded, e.g DHIS2, OpenMRS, and influence the architecture by ensuring their interests are maintained. This mean that the organisations that are often based outside of Africa have the main say in the HIS. Implementation-level designers are also stakeholders and do request change through *Jira* but developers are often get ignored or their requests are not prioritised and after a while outdated. This impair governance quite badly for many of the local developers and is one of the pitfalls of making the platform to dependent on funding from these large organisations.

In Zomba specifically, the politics run by these large organisation limits the locals' influence over the system which leads to some issues. The level of governance goes through a path from top to bottom where each level have some governance which restricts the applications features and design choices and limits the next levels'. The developers at the last level are left with very little influence over the application design. It is the overall experience that the influence of each organisation has a positive correlation to the capital donated by that organisation.

5.4 DHIS2 improvements

5.4.1 Labs

Recent efforts have been made to create a lab at the University of Oslo. This labs work is mainly focused on the core of the system and organising research projects. These efforts are made at the top-level of the platform and with DHIS2 being as big as it is the distance information need to travel to get from top to bottom or from the bottom to the top is very far. DHIS2 has a global perspective and is use by 72 countries which account for 2.3 billion people, or 30% of the world population. The lab in Oslo is where all changes to the core is made, and the size of the team doing this job can in no way compare to the impact the system have today and can have in the future. Developers complain that they don't have any real impact on the platform and that they often need to do some tricks, or "hacks" as they called it, to make the system work. The system need to evolve faster to ever changing environments. Decentralising the governance would help these efforts, both by the "Modular System Theory" (Tiwana et al. (2010)) and by letting more specific decisions be made closer to the field. Having labs in major regions having more governance over the platform and sharing resources of use to developers would help in these efforts.

5.4.2 Form capabilities

How people prefer to communicate with different computer systems heavily depend on their experience with computers, the context and the system's interface itself. In our experience a skilled user usually don't mind typing and is happy to do so if the workload isn't to heavy and the work not to autonomous. However when the workload become either very autonomous or the workload increases clicking is simply quicker and less cumbersome. The speed of clicking through a set of choices can also be increased through clever design and premade common combinations of radio buttons or preset numbers. The clicking preference is also encouraged by the mobile friendly focus. The applications being made have often opted for familiarity when designing new solutions, which means that the system often look very like the forms they aim to replace. It is in our experience a large part of the functionality of the system, being able to quickly interact with large forms, this is however poorly supported in DHIS2. For instance the metadata defining won't let you define a *more then one answer* data entity. While multiple choice exist and can be implemented as intended in the Tracker system, the API query don't contain any information on how to interpret this functionality. Therefore making new functionality should focus on making it easy to create forms.

Chapter 6

Conclusion

Our focus in this thesis has been on designing effective evolutionary platforms. While platform ecosystems offer scale to systems so businesses can reach economies of scale, there is need for specialised solutions focusing on the user. This is inherently contradicting, and solved by making a divide between *implementation-level design* and *generic-level design* where ideally the *generic-level* exhibits low variety and high reusability while the *implementation-level* exhibit high variety and low reusability. In this way, developers solve specific use cases by developing modules for each need. Our experience with this implementation however, is that change and decision making often takes too long and inhibit evolution of the platform. Systems can often get to focused on the individual programs without any shared interfaces. The ecosystem then is experienced as very separate, and users can't find any coherence between modules that are part of the same ecosystem. We have called this *program centred design* because the focus of the developers are on the individual modules, rather then the collective ecosystem. What they should consider is making a generic program which is shared by further developments. In the HIS scenario from our fieldwork, the generic program would be a patient and we have therefore recommended a *patient centred design* approach. There is a need for a ecosystem which can share information across modules and make a collective user experience. A problem with this solution is that the evolutionary rate is not constant and differ greatly from location to location. The evolutionary dynamics therefore need to be localised and the subsystems must be able to evolve separately.

Platform governance can have an effect on local evolutionary dynamics. By partitioning the ecosystem into smaller subsystems, where decision rights is given to each part, they can evolve separately with the pace of their situation. We discussed the roll-out of national IDs in Malawi, and that more urban areas would roll it out faster than certain rural areas. Because of the predicted inconsistency in the roll-out, we argued for a more dynamic and agile approach to the implementation, where the local clinics could switch to the new ID system as they saw fit. We discussed how agile development and platform ecosystems don't work well together, because of the issue of scaling agile processes. We have explored the possibilities of partitioning a large development process into many

smaller processes, in our case representing the different local implementations. Normally, changes in systems must be done by altering the code base and this is often dependent on the developers that made the system. Throughout this thesis we have argued that adopting an Adaptive-Object Model (AOM) approach can shift this responsibility to the end users. By introducing a concept like AOM, where data interpreted at run-time decides what is rendered, all it takes to change local implementations are the end users manipulation of the metadata. It also aids the development processes to be able to be more agile, as it allows for more frequent changes locally.

Learning points

After writing this thesis we are left with a couple of learning points, which can be used by anyone who might write similar work in the future. If there is one thing we learned from the preliminary work and the field trip, it was the importance of being proactive and take initiative. We feel like it really payed of every time we were proactive, such as when we tried to determine the time and location for our field trip. In Malawi, things often took a long time. Looking back, we think we could've been more proactive in having meetings and activities every day. The trip to the clinics also took a long time to arrange, and we were virtually completely passive in the process. If we took the initiative here, we might be able to get more out of it, and go on more than one. We had our reasons of course, being that we wanted to stay true to the action research method. We wanted to capture exactly how things were handled, which meant not interfering with meeting scheduling, etc.

When it came to the actually data collection though, we had the opposite problem in the beginning. When we were new to this method, we tended to lean more towards participating than observing their methods. This lead to a lack of research on the processes taking place, before we realised our mistake.

When visiting a foreign country to do research, it is important to establish social connections. We experienced a huge difference in information flow and usefulness of meetings and conversations the more we got to know the team. Our trip to Malawi lasted for only one month, and most of our findings came in the last parts of the trip. We weren't able to iterate our findings with the team in Malawi in this short period of time. If we could've changed something it would definitely be to schedule a revisit for another month, after we had processed and reviewed the information we received.

Future work

One of the ideas we put the most emphasise on in this thesis, is the Adaptive-Object Model (AOM) and the approaches related to implementation of this model. We talk about the theories around these approaches and how they potentially can help large platform ecosystems becoming more agile. Further research on this topic would include to test the theories in the field over a period of time, and document the results. It could be interesting to see how a system like the coming Malawi health information system could benefit from utilising AOM with a more patient centred approach to their HIS.

Bibliography

Ajuwon, G. & Rhine, L. (2008), 'The level of internet access and ict training for health information professionals in sub-saharan africa'.

Berg, M. (2001), 'Implementing information systems in health care organisations: myths and challenges', *Int J Med Inform* pp. 143–156.

Bodker, K., Kensing, F. & Simonsen, J. (2004), 'Challenges of user participation in the design of a computer based system: the possibility of participatory customisation in low income countries', *IEEE Transactions on Professional Communication* .

Braa, J. & S., S. (2012), 'Integrated health information architecture - power to the users.'.

Bryman, A. & Bell, E. (2011), *Business Research Method*, Oxford University Press.

DHIS2 Fact sheet (n.d.).
URL: *https://s3-eu-west-1.amazonaws.com/content.dhis2.org/general/dhis-factsheet.pdf*

Garud, R., Jain, S. & Tuertscher, P. (2008), 'Incomplete by design and designing for incompleteness'.

GSMA (2014), 'Digital inclusion report 2014'.

GSMA (2019), 'The state of mobile internet connectivity report 2019'.

Hann, I., Roberts, J. & Slaughter, S. (2004), 'The governance of open source initiatives: What does it mean to be community managed?', *Association for Information Systems* .

Haux, R. (2006), 'Health information systems - past, present, future'.

Hilbert, M. (2014), 'Technological information inequality as an incessantly moving target: The redistribution of information and communication capacities between 1986 and 2010', *Journal of the American Society for Information Science and Technology* .

Kimaro, H. & Titlestad, O. (2008), 'Participatory it design: Designing for business and workplace realities'.

Kujala, S. (2003), *User involvement: A review of the benefits and challenges*, Behaviour and Information Technology.

Kushnriuk, A. & Nøhr, C. (2016), 'Participatory design, user involvement and health it evaluation', *Stud Health Technol Inform.* .

Lane, B., Car, N., Leonard, J., Lipkin, F. & Siggins, A. (2015), 'Mobile field data collection for post bushfire analysis and african farmers', *Springer, Cham* .

Li, M. & Nielsen, P. (2019), 'Making usable generic software. a matter of global or local design?'.

Li, Z., Wilson, C., Xu, T., Liu, Y., Lu, Z. & Wang, Y. (2015), 'Offline downloading in china: A comparative study', *ACM Conference* .

Lule, J. (2018), *Globalisation*, Rowman Littlefield.

Manya, A., Braa, J., Øverland, L., Titlestad, O., Mumo, J. & Nzioka, C. (2012), 'National roll out of district health information software (dhis 2) in kenya'.

Meyer, J. (2001), 'Qualitative research in health care: using qualitative methods in health related action research'.

O'Mahony, S. (2007), 'Why developers participate in open source software projects: An empirical investigation', *Journal of Management and Governance* .

opensource.com" (n.d.), 'What is open source?'.
URL: *https://opensource.com/resources/what-open-source*

Pollock, N., Williams, R. & D'Adderio, L. (2007), 'Generification work in the production of organizational software packages'.

Poppe, O. (2012), 'Health information systems in west africa : Implementing dhis2 in ghana', *University of Oslo* .

Poppe, O., Jolliffe, B., Adaletey, D., Braa, J. & Manya, A. (2013), 'Cloud computing for health information in africa? comparing the case of ghana to kenya'.

Reason, P. & Bradbury, H. (2008), 'The sage handbook of action research: Participative inquiry and practice (2nd edition)'.

Reynolds, C. & Wyatt, J. (2011), 'Open source, open standards, and health care information systems'.

Ries, E. (2011), *The lean startup: How today's entrepreneurs use continuous innovation to create radically successful businesses.*, Crown Publishing Group.

Rosling, H., Rosling, O. & Rõnnlund, A. (2018), *Factfulness*, Sceptre.

Simonsen, J. & Robertson, T. (2012), *Routledge International Handbook of Participatory Design.*, Routledge.

Tiwana, A., Kosynski, B. & Bush, A. (2010), 'Platform evolution: Coevolution of platform architecture, governance, and environmental dynamics', *informs* pp. 675–687.

Walls, E., Santer, M., Wills, G. & Vass, J. (2015), 'The dreams plan: A blupoint strategy for e-education provision in south africa', *Electronic Journal of Information Systems in Developing Countries* .

Weber, S. (2004), *The Success of Open Source*, Harvard University Press.

Webster, P. (2011), 'The rise of open-source electronic health records'.

Wever, R., Kuijk, J. & Boks, C. (2008), 'User-centred design for sustainable behaviour', *International journal of Sustainable Engineering* pp. 9–20.

Wyche, S., Smyth, T., Chetty, M., Aoki, P. & Grinter, R. (2010), 'Deliberate interactions: characterizing technology use in nairobi, kenya', *Association for Computing Machinery* .

Yoder, J., Balaguer, F. & Johnson, R. (2001), 'Architecture and design of adaptice object-models', *ACM SIGPLAN Notices* .

Yoder, J. & Johnson, R. (2002), 'The adaptive object-model architectural style'.

Appendix A

Forms

ART Patient Card **Yellow Card** ***Adult ARV Formulations*** *Version 4* Transfer-In Date ____ **ART Reg no** ____ **Year** ____

Patient / Guardian Details

Patient Name

Sex, Birth Date: **M** **F** DOB

Physical Address

Guardian Name

Phone: Patient | Guardian

Agrees to FUP: **N** **Y** Guardian Relation

Status at ART Initiation

WHO clinical conditions

Clin Stage: **1** **2** **3** **4** **P**SHD

CD4 / TLC: **%**

CD4/TLC Date

Height, Wt.: cm | kg

Age at Init.

TB Status at Initiation: **N**ever/ >2yrs | **L**ast 2yrs | **C**urr

KS: **N** **Y**

Pregnant / Lactating: **N** **P**reg **L**act

Ever taken ARVs: **N** **Y**

Last ARVs (drug, date)

Confirmatory HIV Test before ART Start

Site, HTC Serial No.

Test Date: **R**apid **PCR**

ART educat. done: **N** **Y** Date

TB treatm. Reg. No. | Start Date

ART Regimens Regimen | Start Date

Visit Date *day month year*	Hgt *cm*	Wt *kg*	Adverse Outcome	Outcome Date	ART Regimen *Adult Formulations*	Side Effects (Current) Specify 'Other' in Notes	TB Status (Curr.)* Suspected No Yes	Confirmed noRx Rx	Pill Count	Doses Missed	ARVs Given No. of tablets To	CPT No. of tablets	Family Plan. Depo given	No. of condom	Months on ART	Viral Load Sample taken	Result	Next Appointment
Jan			D Def Stop TO		**1**A **2**A **3**A **4**A **5**A **6**A **7**A **8**A Oth	No PN HP SK Lip Oth	N Y	C Rx			P G					Bled		
Feb			D Def Stop TO		**1**A **2**A **3**A **4**A **5**A **6**A **7**A **8**A Oth	No PN HP SK Lip Oth	N Y	C Rx			P G					Bled		
Mar			D Def Stop TO		**1**A **2**A **3**A **4**A **5**A **6**A **7**A **8**A Oth	No PN HP SK Lip Oth	N Y	C Rx			P G					Bled		
Apr			D Def Stop TO		**1**A **2**A **3**A **4**A **5**A **6**A **7**A **8**A Oth	No PN HP SK Lip Oth	N Y	C Rx			P G					Bled		
May			D Def Stop TO		**1**A **2**A **3**A **4**A **5**A **6**A **7**A **8**A Oth	No PN HP SK Lip Oth	N Y	C Rx			P G					Bled		
Jun			D Def Stop TO		**1**A **2**A **3**A **4**A **5**A **6**A **7**A **8**A Oth	No PN HP SK Lip Oth	N Y	C Rx			P G					Bled		
Jul			D Def Stop TO		**1**A **2**A **3**A **4**A **5**A **6**A **7**A **8**A Oth	No PN HP SK Lip Oth	N Y	C Rx			P G					Bled		
Aug			D Def Stop TO		**1**A **2**A **3**A **4**A **5**A **6**A **7**A **8**A Oth	No PN HP SK Lip Oth	N Y	C Rx			P G					Bled		
Sep			D Def Stop TO		**1**A **2**A **3**A **4**A **5**A **6**A **7**A **8**A Oth	No PN HP SK Lip Oth	N Y	C Rx			P G					Bled		
Oct			D Def Stop TO		**1**A **2**A **3**A **4**A **5**A **6**A **7**A **8**A Oth	No PN HP SK Lip Oth	N Y	C Rx			P G					Bled		
Nov			D Def Stop TO		**1**A **2**A **3**A **4**A **5**A **6**A **7**A **8**A Oth	No PN HP SK Lip Oth	N Y	C Rx			P G					Bled		
Dec			D Def Stop TO		**1**A **2**A **3**A **4**A **5**A **6**A **7**A **8**A Oth	No PN HP SK Lip Oth	N Y	C Rx			P G					Bled		

NTNU

Norwegian University of
Science and Technology