UNIVERSITY *of the*
WESTERN CAPE

An authoring tool for generalised scenario creation for SignSupport

By
Lindokuhle Sifiso Duma

A thesis submitted in fulfilment of the degree of
Master of Computer Science

January 2016

Declaration of Authorship

I, Lindokuhle S. Duma, declare that this thesis titled, Usability and content verification of a mobile tool to help a Deaf person with pharmaceutical instruction, and the work presented in it is my own. I confirm that:

- This work was done wholly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

_____

Date:

_____

Dedication

Dedicated to Sindisiwe Kubheka, my fiancee, for always being the pillar of my strength.

Abstract

This thesis describes the development cycles of an authoring tool that generalises scenario creation for SignSupport. SignSupport is a mobile communication tool for Deaf people that currently runs on an Android smartphone. The authoring tool is computer-based software that helps a domain expert, with little or no programming skills, design and populate a limited domain conversation scenario between a Deaf person and a hearing person, e.g., when a Deaf patient collects medication at a hospital pharmacy or when a Deaf learner is taking a computer literacy course. SignSupport provides instructions to the Deaf person in signed language videos on a mobile device. The authoring tool enables the creation and population of such scenarios on a computer for subsequent 'playback' on a mobile device. The output of this authoring tool is an XML script, alongside a repository of media files that can be used to render the SignSupport mobile app on any platform. Our concern was to iteratively develop the user interface for the authoring tool, focusing on the domain experts who create the overall flow and content for a given scenario. We had four development iterations, where the first three were evaluated for usability; for both pharmacy and ICDL course scenarios with purposive sampling. The fourth iteration focused on using the authoring tool to design an ICDL practise mobile app, recording the necessary SASL videos and using an XML parser to render the designs XML script into an Android app. The research conducted herein leveraged multiple approaches to content authoring and generalisation; and further that software generalisation can improve accessibility and affordability for the ultimate end users. The thesis concludes with a summary of recommendations and lessons learnt.

Acknowledgements

I would like to thank all my lab mates for their continued support and for keeping me on my toes on my research. I would like to thank Prang for always ensuring that I'm prepared when I'm collecting data or presenting my work; my Supervisor, Bill Tucker, always making tasks look easier and for always being there when I need him.

I would also thank the Deaf Community of Cape Town for their involvement, including Meryl Glaser. Thanks also to George Ngethe, Marshalan Reddy, and Edwin Blake at UCT for their collaboration on this project. We also thank Telkom, Cisco, and the THRIP (Technology and Human Resources for Industry Partnership) initiative of the South African Department of Trade and Industry for financial support via the Telkom Centre of Excellence (CoE) programme. THRIP funding (project TP13072623839) is managed by the National Research Foundation (NRF). Any opinion, findings and conclusions or recommendations expressed in this material are those of the authors and therefore the NRF/THRIP does not accept any liability in regard thereto.

Lastly, I would like to thank my parents and siblings for always being there for me.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Deaf with a capital D refers to a linguistic and cultural group of people with/without hearing loss who mainly use SASL as their mother tongue [8]. Deaf people often experience communication barriers while communicating with a hearing person who cannot sign [10, 12, 13, 14]. While most accessible technologies support voice and text communication, this presents usability difficulties for Deaf people with low functional text literacy. SASL interpreters are very expensive for Deaf people as they are very scarce, so Deaf people battle significantly to communicate with hearing people in the absence of an interpreter.

The current versions of SignSupport are two Android mobile applications; one for medical dispensing and another for computer literacy training scenario. The medical dispensing application helps a pharmacist to give comprehensible medical instructions to a Deaf patient during medicine dispensing in the form of pre-recorded South African Sign Language (SASL) videos [29]. These medical instructions are stored on a mobile phones memory card for the patient to view later on. Computer literacy training app helps a Deaf computer skills learner to work through the units of this course at their own pace. The unit instructions are also in SASL videos and are stored on a mobile phones memory card for the Deaf learner to view anytime. The limitations of these SignSupport apps are that they only have SASL instructions, run on an Android mobile platform and each app caters for only one scenario.

Therefore, this authoring tool is aimed to generalise SignSupport to accommodate multiple domain scenarios, for multiple mobile platforms and that can be populated by any language for low literacy end users. The authoring tool for SignSupport generalisation de-

scribed in the thesis is a computer-based application that helps a domain expert with basic or no programming skills to design a front-end interface for a communication flow between a Deaf and a hearing person; in any limited domain scenario, e.g., during medical dispensing from a pharmacist to a Deaf patient; or a self-learning process for computer literacy skills by a Deaf learner [5, 10]. The initial design of the authoring tool came in [5].

1.1 Aims and objectives

This section presents the aims and objectives of the research. SignSupport currently is an android app that only covers the pharmacy and Deaf patient scenario and in South African Sign Language only. This is limiting because it means that all Deaf people in South Africa should buy android phones to have access to this app and this also means that SignSupport only helps the Deaf people when they collect medicine at the pharmacy and nowhere else. Here are the three problems that the research aims to tackle.

1.1.1 Scenario independence

Deaf people experience communication challenges on a daily basis, in different scenarios when they have to communicate with a hearing person who cannot sign. Hence, there is a need for technologies that empowers Deaf people to communicate effectively in many scenarios, when interpreters are not available. Here, we aim to generalise SignSupport to enable Deaf people to communicate with hearing users in multiple limited domain scenarios.

1.1.2 Language independence

SASL has different dialects and signed languages can be significantly unique from country to country; just as for spoken languages. SignSupport needs to be generalised so that it can support different signed and spoken languages.

1.1.3 Mobile platform Independence

The are many different mobile platforms available today. It is not easy to predict which mobile platform is possessed by most Deaf people. Hence, SignSupport needs to accommodate

a wide variety of mobile platforms.

1.1.4 Anticipated authoring design

The authoring tool presents the domain expert with screen templates, icons and other features to create a scenario. These screen templates can be populated with text, icons and/or signed language videos (that would be provided by native signed language speakers, not the domain experts, unless they are Deaf, of course). The screen templates explained here are generic screen layouts that can be used as a starting point when creating a scenario [1]. These screens can also be linked to each other in a graph structure. The link between each screen indicates the user interaction on the real mobile app i.e. which screen will show next when the end user clicks a given button. The authoring tool can then produce an Extensible Markup Language (XML) script as an output after the flow is designed by the domain expert. This XML file is consumed by a XML parser to render a mobile application on any given mobile platform.

Figure 1.1 shows the flow of creating a scenario using the authoring tool and rendering a mobile app using the XML parser.

1.2 Background

This section describes the scenarios of SignSupport prior to the authoring tool. This section presents the design and implementation for each scenario.

1.2.1 Doctor and Deaf patient scenario

This is an Internet browser-based mock-up design [31, 27]. The main objective of this mock-up was to help the doctor understand the symptoms of a Deaf patient so that s/he could prescribe medication for the patient. The mock-up presents a set of questions in SASL which the Deaf patient answers before seeing the doctor. The doctor then views the summary of the patients answers in English text [10].

[1] http://techterms.com/definition/template

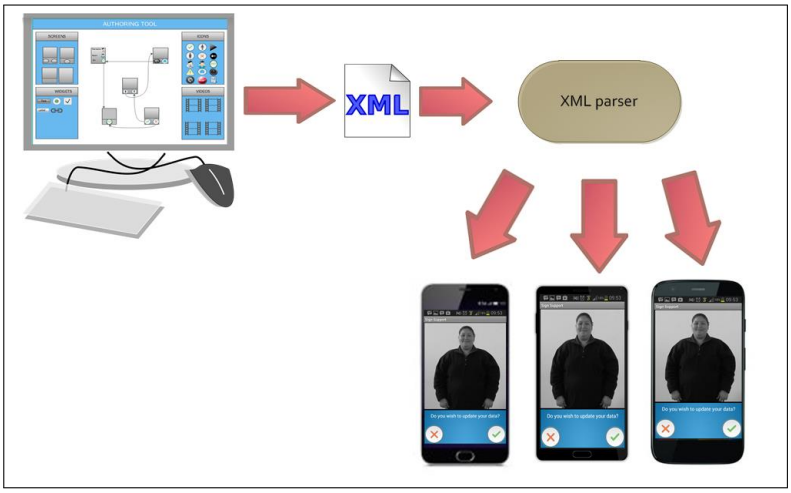

Figure 1.1: Scenario design process
Starting from an authoring tool and ending in a mobile app. The domain expert will design the scenario using the authoring tool, the authoring tool will produce an XML script as an output. The XML parser uses the XML script to render a mobile app for different mobile platforms

Version 2 was on the same scenario as version one and was developed as a Symbian mobile app with XHTML [31]. This mock-up required only a mobile phone with a data connection running a browser that supports Small web format. However, the doctor and Deaf patient scenario was found to be too wide and complicated to be constructed as an application [9].

1.2.2 Pharmacist and Deaf patient scenario

Version 3 then shifted into a more constrained communication domain between a pharmacist and a Deaf patient [9]. Its aim was to help the Deaf patient understand the medical instructions as prescribed by the pharmacists. This mock-up was designed to have 2 types of interfaces: one for the Deaf patient to input their background information prompted by SASL videos and icons and one for the pharmacist to view the Deaf patients background information and dispense medication [9, 29, 30].

Version 4 was a redesign by a multi-disciplinary and trans-university team and then implemented as an Android app by [29, 30]. The construction of this version is in the process of usability testing in a public hospital pharmacy [10].

1.2.3 Computer literacy training scenario

SignSupport mobile application for computer literacy training scenario, helps a Deaf computer skills learner to work through the units of this course at their own pace. The challenge within this course is that some learners understands the content of the course faster than others so the signing educator needs to repeat the lessons until all learners understand. This becomes a burden for the faster learners because they need to wait for the slower learners to understand before the teacher introduces new content. Another challenge is that the educator needs to travel around the world quite often. This puts the course on pause until she comes back. SignSupport for ICDL helps faster learners to move on to the following units of the course while the educator explains the current units to the learners who are struggling [5, 33]. This app also enables the course to go on while the educator is away. The learners just have to write down all the parts that they don't understand so that the educator can explain when she returns.

The authoring tool that designs the ICDL app only designs the front-end of the ICDL app. This authoring tool produces XML script as an output. This XML script represents the content of the app (in plain text) that was designed by the authoring tool. The difference between this authoring tool and the one discribed in this thesis is that this authoring tool can only be used to design ICDL course units whereas the authoring tool discribed in this thesis can be used to design ICDL course units and many other limited domain scenarios.

There is another researcher who is working on an XML parser which is a back-end of the authoring tool. The XML parser uses the XML script to render a platform dependent mobile. This XML parser currently renders Android apps[5].

1.3 Role of author

Our research team sees the potential of SignSupport for additional contexts and limited domain communication scenarios. This can be witnessed by the research listed in the background section above. Newer research streams, under SignSupport, includes health education for Deaf people, which focuses on diabetes and hypertension; testing the medical

dispensing app at a real pharmacy and an XML parser which uses XML to render a mobile app. The role of the author is to create a generalisation authoring tool that aims to generalise SignSupport for multiple platforms, signed languages and limited domain scenarios. This authoring tool would assist domain experts in designing communication flows that meet the specific needs of their Deaf clients. The focus of the research is mostly on multiple scenarios. This is because the current research topics and prototypes under SignSupport are mostly scenario dependent. The researcher works closely with three researchers: the industrial design engineer who designed the medical dispensing app; the computer scientist who created the authoring tool to designs a computer literacy app and then produces XML output; another computer scientist XML parser researcher who created an XML parser that uses the XML to render a mobile app. The industrial design engineer helps to evaluate the authoring tool interface in a pilot session. The two computer scientists helps to design the XML structure that the generalisation authoring tool should produce. This helps to ensure that the XML produces by the generalisation authoring tool meets the expectations of the XML parser. The source code of the generalisation authoring tool can be found in the this link: http://cs.uwc.ac.za/ lduma/masters/authoring_tool.rar. Please note that some of the content have been copied from the paper that the researcher wrote and published at South African Telecommunication and Network Applications Conference (SATNAC). The paper is cited in this thesis and is also attached in appendix F.

1.4 Thesis outline

This thesis is divided into 5 Chapters, chapter 1 introduces the research problem and the proposed solutions. Chapter 2 describes the related work that can be referred to when building up a solution for the research problem. Chapter 3 states the methods that are applied to carry out the research; building and testing the authoring tool. Chapter 4 presents and discusses the results obtained in three iterations of the authoring tool development; Chapter 5 concludes the research and presents a direction for future research. The computer science researcher

Chapter 2

Related work

This chapter presents the related work that is consulted to get ideas on how to create the authoring tool. Section 2.1 describes the authoring tools, the different types and commonly used authoring tools. Section 2.2 presents the cross-platform approaches in app development and programming languages. Section 2.3 presents universal design approaches. Approaches presented here include universal design in learning, housing and WYSIWYG editors.

2.1 Authoring tools

Authoring tools are software that allow authors (domain experts), with basic or no programming skills, to build and populate complex applications with data or hypermedia files for their domain scenarios e.g. E-learning web apps [15]. Authors can create these complex and attractive applications by merely clicking and defining relationships between objects, e.g. forms, text, pictures and videos. Authoring tools provide authors, with programming experience, direct access to common programming constructs such as functions, loops, conditions and variables that are useful to provide end-user interactivity in the application being created [38]. The code inserted by the author is automatically compiled by the authoring tool and reflects in the Graphical User Interface (GUI) pane. The author has an option to preview the scenario design to see how it will look when it is a complete app. S/he can then modify the design or its code until satisfied.

Authoring tools can be classified into three categories such as card-based, icon-based and time-based[1]. Card-based authoring tools allow the author to organise elements as if they were a stack

[1]http://www-it.fmi.uni-sofia.bg/courses/maten/module3/classification.html

of cards. These elements can be viewed individually like cards or pages in a book [40]. An example of such authoring tools are the web or presentation authoring tools, which allow you to view one web page or presentation slide at a time. Icon based authoring tools enable the authors to create a flow or manipulate objects directly to complete a task[2, 21]. Time-based authoring tools arrange an element in time frames e.g. a slide show that plays different music instruments in different time frames. These are commonly known as multimedia authoring tools [2].

Advantages of authoring tools are that they save much time because they have templates. Templates help authors to work faster as they are generic and can be adapted to any form within the intended use of that authoring tool. Therefore, the author can customise and populate with content in a timely manner [2]. Another advantage of authoring tools is that they are easy to use since authors do not have to possess programming skills in order to effectively use the authoring tool. The major drawback of authoring tools is that they have limited options and this might affect the author's creativity [3].

2.1.1 Types of authoring tools

2.1.1.1 Web authoring tools

Creating a website is traditionally done by typing HTML code on a notepad. The demand for more interactive and multi-purpose website drives the developers towards seeking for more automated website generating solutions. This led to the invention of web authoring tools. Web authoring tools generally present the developers with web page templates that the user can add and modify to suit his/her needs. These page templates carry the designs that are common in many websites. For example, a page template could have a banner and navigation buttons that a developer can modify and add links to other web pages or websites; and a footer at the bottom that has links that s/he wants to see on every page. A link manager also helps to verify that the links are done correctly. Other web authoring tools also have a feature to mark the progress of the page e.g. version 1, 2, 3 etc. The developer can also preview the website internally on the authoring tool or externally using a web browser.

[2]http://smallbusiness.chron.com/advantages-disadvantages-using-webauthoring-application-27288.html
[3]http://smallbusiness.chron.com/advantages-disadvantages-using-webauthoring-application-27288.html

2.1.1.2 Multimedia authoring tools

Media content includes text, images, videos and animations. Multimedia authoring tools allows the user to integrate multi media content into one application. These tools provide users, with programming skills, with common programming constructs (looping and conditional branching) that are useful in providing interactive multimedia applications to end-users. Multimedia authoring tools have built-in tools for the user to create/edit different media elements, however these built-in tools are not as comprehensive and flexible as the specialised software.

Multimedia projects are usually managed by a multi disciplinary team whose members include project manager, programmers, producers, graphic artists, video and audio experts. Some multimedia authoring tools enable collaboration such that team members can work on different tasks at the same time.

2.1.2 Examples of authoring tools

2.1.2.1 Prezi presentation tool

Prezi is an online tool that allows an author(s) to create and publish presentations. The author creates a presentation by adding elements (text, pictures, animations and videos) in a canvas [36]. The elements can also be grouped together in frames. Frames and individual elements are linked together to control the flow of the presentation. The elements can be arranged and edited using various tools within Prezi. Zoom in/out is used to add supplementary details and subtopics to the presentation

This tool offers a number of advantages as opposed to traditional presentation tools like PowerPoint [36]. It allows the author to work on any computer that has a high speed Internet connection. Authors in different locations can work collaboratively in one presentation [26]. The content of the presentation is not limited by the size of the slides since this tool uses a canvas, which has no limits.

Disadvantages of Prezi is that one needs to be connected to the Internet in order to work on this tool. This tool is not free and some of its features might not be affordable to many.

Both Prezi and PowerPoint support non-programmers by enabling their users to work only on the graphical user interface. Prezi presents the user with tools and elements that can be added on a canvas; and PowerPoint presents the user with menus and elements to add on the editable slide

templates. Users can also do a trial and error when manipulating objects on the canvas or slides, as these authoring tools enables users to easily recovers from errors. Both these authoring tools allow users to add their own media files to the presentation.

2.1.2.2 Dreamweaver

Dreamweaver is an offline based authoring tool that enables authors to create and manage web-sites. The four main features that make this tool powerful are point-and-click, drag-and-drop, HTML inspector and RoundTrip HTML. Point-and-click; and drag-and-drop enables authors to visually create websites without having knowledge about HTML or other scripting languages. This in turn makes the web authors work faster since they no longer have to type the HTML source code by hand. HTML inspector enables web authors to have access to and control the HTML source code generated by Dreamweaver. They can then edit this source code willingly. RoundTrip reads and validates the syntax of the HTML source code that is imported from other HTML editors.

Advantages of Dreamweaver are that the author can work faster by visually designing and developing web sites, and authors can organise their site content in directories from their desktop, in the same way that the web site will be stored in the server. This eliminates the need for a web server when the web site is still under construction. Dreamweaver can also push or pull web content from the server using FTP connections.

2.2 Cross-platform development

Cross platform means developing one application that can be installed on many platforms without having to modify its original source code [8]. Every platform is unique, e.g. android apps are created mainly with Java and iOS apps are built using objective-C. So App developers, who wish to reach out to many target app users, would have to build the same app separately for all mobile platforms. This is time consuming and might lead to app inefficiency. Cross-platform techniques aim to enable developers to build a single app that can be executed on different platforms without modifying its underlying source code. A number of techniques have been developed to achieve cross-platform development: web and hybrid apps, frameworks like PhoneGap and Titenium

2.2.1 Apps

Approaches to achieve cross-platform development include building web and hybrid apps. Both these apps are built using the latest technologies such as HTML5, CSS3 and JavaScript but they differ in the way that they are executed [19].

Web apps are not physically installed on the device but run on the device's browser via URLs. These apps store user data in the server or in browser-based cookies. The advantage of web apps is that they don't need physical installations, so any device with a web browser and Internet connection e.g. Facebook mobile. The drawbacks of web apps however, is that they are inaccessible to a device without an Internet connection. Some mobile web-browsers changes the look-n-feel of some web-apps or even deactivate other web-app's features, which can affect user experience. These apps also have limited access to the device's hardware features [23, 19].

Hybrid apps, on the other hand, try to combine the advantages of both web and native apps. These apps are installed on the device and use special APIs to gain access to the device's hardware features. Common API used in hybrid apps, to achieve the platform look-n-feel is called jQuery languages.

2.2.2 Programming languages

2.2.2.1 Java

Java is a programming language that supports multi-threading, automatic memory management and object orientated programming [11, 39]. Java source code is not directly compiled into a native language, but into bytecode. Bytecode is an architectural independent set of instructions and it enables ease of application migration as the computer systems evolves [11, 22]. Bytecode is created by the Java virtual machine (JVM). JVM can execute bytecode instructions one at a time, through just-in-time compilation, or bytecode instructions can be compiled into native instructions and then executed directly. Compiling one instruction at a time helps to verify untrustworthy source code. This is not desirable for frequently used instructions, as they will have to be compiled into native instructions every time they are needed. Compiling bytecode instructions into native instructions, before execution, is known as ahead-of-time compilation. This is also common in c/c++ programs [11]. Ahead-of-time compilation sacrifices the application portability and verification of source code.

Java performance lags behind compared with other programming languages like C, due to a number of issues [35]: Arrays in Java can only be one or two dimensional, there is not support for multidimensional arrays. Java also does not support complex arithmetic and operator overloading.

2.2.2.2 HTML5

HTML5 is a scripting programming language that has been adapted in the web apps due to its advanced features [23]. HTML5 app runs on a devices web browser. This means that there is no need for the target user to manually install and update such apps. HTML5 app can store data offline and it can fully function with no Internet connection. Some of the common features of HTML5 are hardware integration which gives the web access to the hardware features of the device; supports different multimedia formats and user interaction (touch, speech and vibration) [23, 19].

2.2.3 Markup languages

2.2.3.1 Extensible Markup Language

Extensible Markup Language (XMl) is a scripting language that is made up of tags that define data [44]. Unlike HTML, the tags of XML are created by the developer when creating XML documents. These tags are used to describe data within the XML document(s) for ease of storage and retrieval [6]. Each document has a root tag which all other tags belongs to. For example, an XML document that has information about books might have a root entity that gives information about where those books belong i.e. library. The following tags might be the faculties in which the books are made; then the other tags could be carrying the book name and its author number. This XML document can be used to search books for a certain faculty or a book written by a specific author. Figure 2.1 represents an example of such an XML document.

XML has a number of advantages, which are [6, 32]:

- XML makes is easy to send data between any incompatible applications or devices.
- XML entities can be used to optimise information retrieval in large databases.
- XML is stored as plain text so it is easy to upgrade systems without losing data.

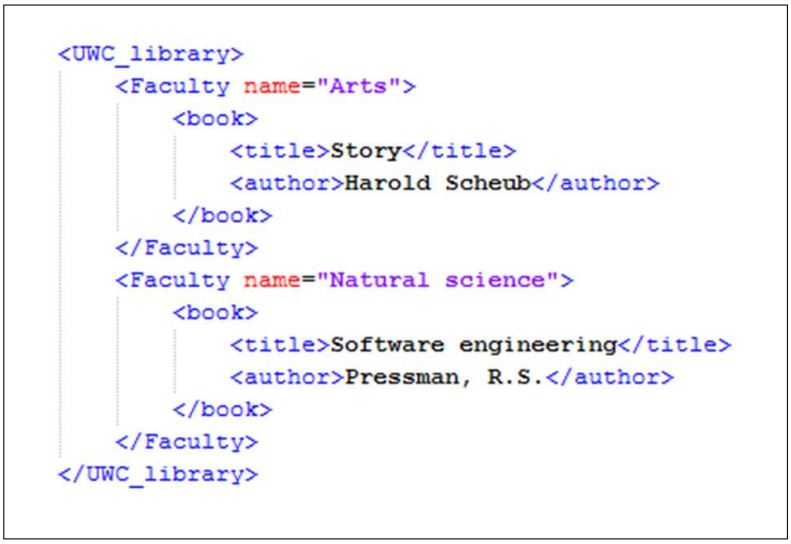

Figure 2.1: A screenshot of XML structure

that stores the information of the books in the library. This is an example of how XML can be used to represent scenario specific data.

XML is also accompanied with a number of technologies that assist in processing data in XML documents. These technologies include Extensible Stylesheet Language Transformation (XSLT), Xpath and Xlink [6, 32]. XSLT is used to transform XML documents to other browser-based languages like XHTML. XSLT can also edit XML documents like adding and removing elements and attributes to and from the XML file. Xpath is also used in XML transformation but its main goal is to navigate through the elements and attributes of an XML document. XLink is responsible for defining links to other resources [6]. It does this by allowing attributes to be inserted into XML documents to describe both internal and external linked resources .

2.2.3.2 JSON

JSON is a data exchange language that has been designed to be human readable and easy for computers to parse [34]. JSON is ideal for Javascript and it is well known for data exchange between JavaScript and web pages. JavaScript has an eval() function which directly parses JSON into an object [34]. This increases the performance of JSON in a way that it becomes faster to parse than XML, because XML required standalone libraries like DOM. JSON can represent both strings and arrays (non-ordered and ordered list) [34]. Here are some of the advantages of JSON [34]:

- It does not use the concept of tag.
- It can be parsed faster than XML.
- It needs less data size to describe the same data

Some of the drawbacks of JSON over XML are [34]:

- Lack of input validation
- Lack of support for data with mixed content
- Lack of namespace support

2.3 Universal design

Universal design entails creating software that can be used by all people effectively, without the need for adaptation or specialised resources [28]. Principles of this method include flexible to use, simple and intuitive to use, perceptible information, tolerance for error, size and space for approach and use [28, 7]. Some of the common approaches of universal design are described below.

2.3.1 Learning

Universal design in learning seeks to address the needs of all learners, even those with learning disabilities [24]. A learning process can be made flexible by having a virtual group where the learners can post questions or have discussions on a certain subject. The teacher can also be a member in that group engag with the learners or have individual sessions with the learners. This is very helpful especially for those learners that are shy to ask questions in class. The textbook used by learners can be assessed for the way their content is presented and organised so that their level of difficulty can be determined. Learning software should also give positive feedback to the learners when they make mistakes. This can happen by either the software indicating to the learner to try again or the software can provide hints on how to complete a certain task [24].

2.3.2 Housing

Universal design in housing has two concepts: visitability and accessibility [16]. Visitability refers to applying universal design in the house in order to accommodate an elderly or disabled person visiting that house. Accessibility here refers to applying universal design in the house to accommodate an elderly or disable person living in that house. Universal design applies to both exterior

and interior aspects of the house.

Exterior aspects includes accessible paths with no steps, that connects to other facilities like sidewalks, toilets etc. Automatic doors are also desirable for people who might have difficulty opening doors by themselves. Interior aspects includes having doors that open wide enough to accommodate a wheelchair, Bathrooms should have appropriate space and have grabs by the wall, near the toilet seats for physically challenged people for balance. Kitchens should accommodate multiple users. Its furniture can be movable and be in a reachable height to accommodate a person with a wheelchair. The floor could have a carpet to accommodate a person who uses crutches[16].

2.3.3 WYSIWYG editors

The acronym "WYSIWYG" means What You See Is What You Get, which refers to a system or an application that has a graphic interface and enables the user to view the end results of the product while s/he is still creating it [41] . An example of WYSIWYG editor is Microsoft Power Point application. Power point presentation r application allows you to see the overview of the slides presentation while you are still creating it. The principles of WYSIWYG editors state that [41]

- Physical representation of actions instead of complex language specific commands
- Reversible operations whose results can reflect on the object immediately.
- Layered approach to learning that enables novice users to learn useful commands until they become experts

Thus the WYSIWYG editors are visual and allow the user to directly apply changes to the product and the results are visible immediately on that product. The formatting commands of these editors are physical actions such as mouse clicks, button presses; as opposed to line command editors that require the user to recall and type all the formatting commands and their syntax; in the command line. The labeled menus these editors invoke commands and also inform the user about the available features within the editor [41]. This frees the user from having to memorise all the complex commands in order to use the editor effectively. Most line editors requires a display command in order for the user to confirm the modifications that s/he made on the product but with the WYSIWYG editor, changes made reflect immediately.

The benefits of WYSIWYG editors are that they could have an "undo" option, which invokes an inverses operation of the previous action(s). This would then reverse the changes that were made

previously. Error messages would rarely be needed in this case because users can immediately see the results of the action they just made; and if they dont like it then they press the undo button. Users would also experience less nervousness as all actions are easily reversible [41]. Experts can work faster and are able to complete many different tasks. Such editors also have built in tutorials or wizards that help novice users to learn about the editor in a step-by-step process. This also gives users confidence when using the editor because they can initiate actions and predict their outcome.

WYSIWYG editors also have a few issues. The graphical representation of their features might not be that obvious to the user and hence users might take more time to learn all the features of the editor [41]. Most editors that target international users might have a misleading graphical representation. This might lead to user frustration if they make wrong conclusions about the features. Some editors need bigger monitors to display the progress of their intended product and all its features. So users with smaller computer screens might have issues as the editor's interface might appear too small.

2.4 Summary

The structure and the features of the existing authoring tools are going to be used as a reference to inform the interface design of the SignSupport authoring tool. SignSupport generalisation authoring will be a computer-based front-end that allows the domain specialist to construct dialogues. The general goal of the research is to investigate the needs of users who are highly skilled but not computer specialists. Universal design emphasises the need to accommodate all possible users when building a product. The authoring tool will have tooltips; help options and a user guide for novice users. This enables any author to learn to use the authoring tool quickly and also to have assistance within the authoring tool in case s/he needs it. These helper options can also be adapted into signed language videos, as tutorials that demonstrate certain functions for Deaf authors. SignSupport generalisation authoring tools will produce an XML document that has all the details of the scenario that the domain expert has designed. XML is chosen over JSON because it has no predefined tags or restricted names. This means that any name can be used as a tag name or namespace. XML also has technologies like XPath, Xlink, DOM, SAX etc. which help to validate the XML structure and flow. So XML has more tools and support than JSON, its competitor. The authoring tool users will be working on the authoring tool's GUI. This enables them to manipulate objects directly and also to see the results as they progresses with their tasks. This is learned from WYSIWYG editors. The next chapter presents and discusses the methods that will be used to guide the research processes and authoring tool development.

Chapter 3

Methodology

This chapter discusses the methods that are used to design and implement the authoring tool. Section 3.1 introduces the research questions. Section 3.2 explains the methods that are used to answer our research questions. This includes the method that guides the research process and the implementation of the authoring tool. Section 3.3 describes how we are going to evaluate the authoring tool for usability and scenario creation. Section 3.4 explains the ethics that we need to follow as we are evaluating the authoring tool with human participants.

3.1 Research questions

The research question for the project is as follows: How can we generalise SignSupport for any given limited domain conversation scenario and also accommodate different signed languages, and multiple mobile platforms? This question is further broken into three sub-questions:

1. How can we make SignSupport platform independent?
2. How can we make SignSupport scenario independent?
3. How can we make SignSupport language independent?

The first sub-question investigates techniques that contribute to platform independence. In order to reach many Deaf people, SignSupport needs to be ported to many mobile platforms like Blackberry, iOS, Windows and more [19]. One cannot predict the kinds of mobile phones that Deaf people prefer. Although, research can be conducted to find out about the mobile phones that are widely used by Deaf people. This would narrow the research focus towards widely used mobile phones; and filter out the rest. The goal of this research is to accommodate all mobile phones

so that all Deaf people have access to the knowledge and information that they need, using their mobile phones. The only criteria is that the mobile phone should a high resolution screen to play high quality signed language videos.

The authoring tool needs to have features that enable the domain expert to design a scenario and then port it into a platform dependent mobile app. To achieve this, we aim to build an authoring tool that a domain expert can use to design a limited domain scenario and then produce an XML script, as an output, to be parsed into a mobile app of different platforms by an XML parser. XML script is a text file that contains information of the designed scenario flow and all its data assets (text, icons and videos) that go with each and every screen in the scenario. An XML parser consumes and parses the authoring tool's XML output in order to render platform-specific mobile apps e.g. Android, iOS or Windows Mobile.

The second sub-question investigates ways in which we can expand SignSupport to accommodate multiple limited domain scenarios. SignSupport currently covers the medical dispensing scenario between the Deaf patient and the pharmacist only [29]. Many other scenarios exist where Deaf people experience difficulties when they need to communicate with hearing people who cannot sign. SignSupport needs to accommodate a wide range of limited domain scenarios in order to effectively assist Deaf people in the absence of a SASL interpreter.

The authoring tool needs to provide the domain experts with different templates that they can customise and populate with scenario specific data (icons, text and videos) and then link these templates to define a conversation flow, in the form of a decision tree. Any domain expert will be able to use/re-use the provided templates or create new templates for a limited domain specific conversation flow.

The third sub-question investigates ways for SignSupport to accommodate multiple languages, both signed and spoken. SignSupport currently operates in SASL pre-recorded videos and English text. There could be a time where the same limited domain scenario needs to be created for other languages, e.g. American Sign Language and Spanish. This simply means that the flow of the scenario remains the same but the scenario data assets changes (signed language videos, icons and text).

The authoring tool needs to enable the author to create a limited domain scenario design, that can be changed and populated with different data assets and then update the XML script.

3.2 Methods

This section describes the methods that guides the research process and the steps of the prototype development and testing.

3.2.1 Action research

Action research (AR) is a methodology that is cyclic in nature and consists of spiral iteration steps that involves circles of planning, action and reflection [18]. This method encourages the researcher to conduct research with participants who are experiencing a problem under study, in their everyday lives; as opposed to reading articles about them and then creating case studies based on the literature. Researchers work collaboratively with the participants in a cycle of activities to try out a theory in real situations, gain feedback and reflect from it and try again in the next cycle [1]. The main focus is on the participants because the human aspect of each solution needs to be taken into consideration because people have different and conflicting views and attitudes; people also change over time. AR contributes to scientific knowledge and sustainable social change. AR can also be summarised into these points [17]:

- AR encourages collaborative action from both researchers and participants.
- The researcher needs to understand the values of the participants as they have an impact on the results at the end of the research.
- Long term collaborative work with the participants, other than surveys or single interviews, helps the researcher to better understand the participants and gain deep in-sight information that would have been impossible to obtain in a short space of time.
- The AR researcher needs to follow an ethical framework that ensures that the participants' rights are not compromised.

3.2.2 Iterative prototyping

Iterative prototyping is a software development method that is based on a cyclic process of prototyping, testing and obtaining feedback. The prototype is then refined based on feedback obtained; and then tested again in the next iteration [3]. This method is carried under the assumption that

the researcher doesn't know what the final product should look or function like; and whether the target users will find it useful to them. Unlike the waterfall method that delivers the software product to its intended users only when it is complete [4]; in iterative prototyping, the researcher builds the initial prototype that carries only the core features of the desirable software product [3] . The prototype is then taken to its target users for evaluation. The researcher obtains feedback and uses it to improve the prototype in the next iteration. The number of iterations is usually 2 or 4; where the current prototype has new features and other modifications that have been suggested from the previous iteration's feedback. Iterative prototyping helps to uncover software bugs and errors in the early stages of the software development life cycle. The advantage of finding software bugs early is that they are inexpensive to fix [3]. Other benefits of iterative design is that it exposes usability issues early and minimises the chances of user resistance.

3.3 Applying the methods

This section describes how we applied action research and iterative prototyping to achieve the goals of our research. We start by describing our participants, who are the domain experts; we then describe the scenarios that we investigated in the research; the motivation behind these scenarios and how the authoring tool evolved from one iteration to the next.

3.3.1 Domain experts

Our target participants were sophisticated Deaf people and hearing domain experts. Deaf participants are recruited from a non-government organisation (NGO) called Deaf Community of Cape Town (DCCT). The reason for choosing Deaf participants from DCCT is that DCCT has a long term relationship with the University of the Western Cape (UWC), whereby they have been working collaboratively on multiple projects that aimed at empowering Deaf people in their everyday lives. We also recruited hearing domain experts who specialise in these scenarios: medical dispensing, ICDL training and antenatal care. The criteria in choosing hearing domain expects is that they must have worked with Deaf people before or have significant knowledge about the communication challenges that Deaf people face when they have to communicate with hearing users who cannot sign.

3.3.2 Scenarios

These are the scenarios that we plan to create, using the authoring tool in the testing stage.

Medical dispensing: Traditionally, the doctor prescribes the medication and then gives the patient a medical prescription, to take to a pharmacy. The patient hands the prescription to the pharmacist, over the counter, at the pharmacy. The pharmacist can also verify the prescription and contact the doctor if she finds an error; otherwise s/he proceeds with medical dispensing. The pharmacist writes the medical consumption instruction on the label pasted around the medicine container. Lastly, she explains those instructions to the patient, to ensure that the patient understands how to consume his medicine. Communication between the patient and the pharmacist is important during medical dispensing because if the patient consumes his medication wrongly, that may lead to serious health hazards.

The current SignSupport mobile app is build specifically for this scenario. It has been tested for usability and proved to be very useful for Deaf people. We would like to use the authoring tool to produce the same app as SignSupport that can be ported to other mobile platforms.

ICDL course: the IDCL course is an internationally recognised course that teaches basic computer skills. The modules taught in this course include Microsoft Office units (Word Processing, Spreadsheets, Presentations and Databases); file management; online information and communication skills. This is currently offered at DCCT for Deaf adults and it creates an opportunity for Deaf people to have access to Information Technology and take part in research conducted by UWC and UCT Computer Science Department; for Deaf people.

Antenatal care Antenatal healthcare is important for both the pregnant woman and her child. A women needs to be constantly monitored for risks that might harm her or her child as the pregnancy progresses. A research project was initiated which aimed at promoting antenatal healthcare. The approach used by their project was to send SMS notifications to the participants (pregnant women) in the different stages of their pregnancy [25]. The SMS notifications included reminders of visits to the health-care facilities and useful health information that is relevant to their pregnancy stage.
This project can benefit from the authoring tool in a way that it can be used to design an antenatal care app which can be populated with pre-recorded signed language videos that can be played back, for Deaf pregnant women who are text illiterate. The domain expert in this scenario can choose between designing multiple apps for different stages of pregnancy or create one app that has information about all stages of pregnancy.

3.3.3 Usability testing with domain experts

Usability testing helps to measure the ease of use of the prototype, from a participant. Ease of use determines whether the target users will be satisfied with that system [20]. Usability sessions will be conducted with the real target users of the authoring tool, i.e. domain experts. This provides direct information about how the domain experts will use the authoring tool in reality and their problems with the current interface.

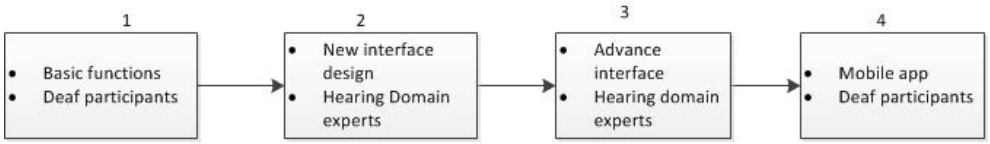

Figure 3.1: Four iterations of the authoring tool development
This figure indicates how each iteration is improved based on the feedback obtained in the previous one.

We plan to have four software development iterations, as indicated by Figure 3.1.

1. This prototype is built from the researcher's observation of authoring tools. The main purpose of this iteration is to lay a foundation of what an authoring tool could look like. This will also help to educate domain experts about authoring tools and their use. The prototype enables domain experts to judge and give feedback based on what they see, which is possibly more beneficial than asking them to design an authoring tool interface without an example. Deaf people are our participants when assessing this prototype for usability.

2. Feedback from the first iteration is used to improve the prototype. Here we use hearing professionals as our participants when evaluating the prototype for usability. The criteria in choosing the hearing participants is that they have worked with Deaf people before and/or they understand the communication needs of Deaf people.

3. This is the advanced prototype, which has evolved over the last two iterations. Here we measure if the participants are satisfied with the prototype and whether they see this prototype empowering their professional lives in the near future. All the recommendations obtained here will be treated as the future work of the research.

4. This iteration demonstrates all the steps that one needs to design and render a mobile app, using the authoring tool and the XML parser. The app that is created is an ICDL assessment mobile app. Firstly, SASL video instructions are recorded and prepared to be

used in the app. Secondly, the app is designed using the authoring tool to produce an XML file. Thirdly, the data assets (SASL videos and icons) of the app and the XML file are copied to the mobile phone. Lastly, the XML parser is launched to render the expected mobile app. The XML parser processes the XML file to find the location of the app's data assets and to get instructions of what the interface design of the app should look like.

3.4 Ethical considerations

This section discusses the ethical considerations that guide the process of dealing with real participants. The interest is on qualitative data because such data has the possibility to change people's lives [43]. We first present the ethical consideration points under consent form, confidentiality, the use of a SASL interpreter, the risk and benefits of the research. We then state the process of obtaining ethics clearance for this project.

3.4.1 Ethical standards

All participants were informed of their role within this project and what was required from them, and also that participation was voluntary. As such all participants provided consent that they have been informed and understood what was explained to them. In order to protect any and all information obtained during consultation sessions, all participants were required to sign the consent form, having been translated to them in SASL for the Deaf participants.

Confidentiality - Participants were trained on how to use the authoring tool and then asked to design a basic or a domain specific script dialogue using this tool. All interaction, e.g. interviews and focus groups, were recorded (paper, audio and/or video) and securely stored on a computer that was protected with a password to uphold privacy.

Use of an interpreter - Sophisticated Deaf participants working at DCCT that had previous experience using SignSupport were recruited to try out the authoring tool. An accredited interpreter was be used to facilitate the communication between the researcher and the Deaf when the interviews and focus groups were done.

Risk and benefit - This study aimed to improve SignSupport in a way that SignSupport will accommodate different limited domain communication scenarios, multiple signed languages and mobile platforms. Another benefit from this study was that it empowered Deaf people and domain specialists to develop SignSupport scenarios together. The risk was that the script dialogue created from the tool might not cover all parts of the conversation within the limited domain scenario; and this might lead to a communication breakdown between a Deaf and a hearing person. This script dialogue that has been produced by the authoring tool was mainly be used for the purpose of this study and not in the real world.

3.4.2 Ethical clearance process

The researcher needs to fill in an ethics clearance application form. Within that application form, the researcher needs to state the description of the project and it's process.

The application needs to be approved by the UWC Faculty Board Research and Ethics Committees and then by the UWC Senate Research Committee [SR]. SR may also consult outsiders on ethics questions, or consult the UWC ethics subcommittees, before registration of the project and clearance of the ethics. The general rule is that no project can proceed before project ethical clearance has been granted.

SignSupport is our umbrella project that already has ethics clearance. The authoring tool falls under SignSupport; and so the researcher applied for the authoring tool's ethics clearance under SignSupport.

3.5 Summary

The main aim of this research project is to improve SignSupport and increase its impact on Deaf people's lives based on active engagement with the Deaf community and domain expects. We believe that an authoring tool can make SignSupport both scenario and platform (and even language) independent. The primary interest is in evolving the authoring tool's palette and canvas interface in order to better support the design and content population processes with both Deaf and domain expert users. The above-mentioned methods, together with the ethical considerations, will help to carry out the research and authoring tool development. Creating an initial prototype and conduct usability testing sessions to get feedback from our participants will follow an iterative prototyping method. The results obtained in each iteration will be used to inform the next iteration;

and participants will be encouraged to assess the interface design and to suggest more features that they would like to add on to the authoring tool. The next chapter presents and discusses the results obtained from this iterative development.

Chapter 4

Results

This chapter presents and discusses results from iteratively developing the authoring tool and testing it for usability. Iterative development helps to get participants involved in the development of the authoring tool and to accommodate their different needs. Usability testing helps to improve the authoring tool functionality and UI. We had 3 iterations, which ended with a usability evaluation. In each evaluation, participants created and populated different scenarios using the authoring tool. The purpose of creating a scenario was emphasised as the purpose of the authoring tool to the participants. Section 4.1 presents the first iteration, section 4.2 represents the second iteration and section 4.3 represents the last iteration. Section 4.4 demonstrates the process of designing the scenario using the authoring tool, recording signed language videos and finally generating a mobile application using an XML parser.

4.1 First iteration

The information obtained from the literature review and the research questions led us to conclude that we need to create an authoring tool. The authoring tool user requirements were that it should allow the domain expert to independently create and populate a domain specific scenario. The scenario design should indicate the flow of screens and the relationship between them. The authoring tool also needs to produce an XML script to be consumed by an XML parser. The XML parser then renders the XML script into a real app on any platform.

4.1.1 Interface design

The prototype of this iteration has multiple screen templates that are added to the canvas area and then populated with icons and text. Each button is linked to a screen by initially clicking on the button which confirms that you want to create a link. You then click the screen that you would like to link the button with. After creating the link, a blue line is drawn from the button to another screen. The line serves as visible proof that the link has been created. Linking the icon with another screen also indicates user interaction in the scenario when it has been converted into a real app. Here is the screenshot of the interface design in Figure 4.1.

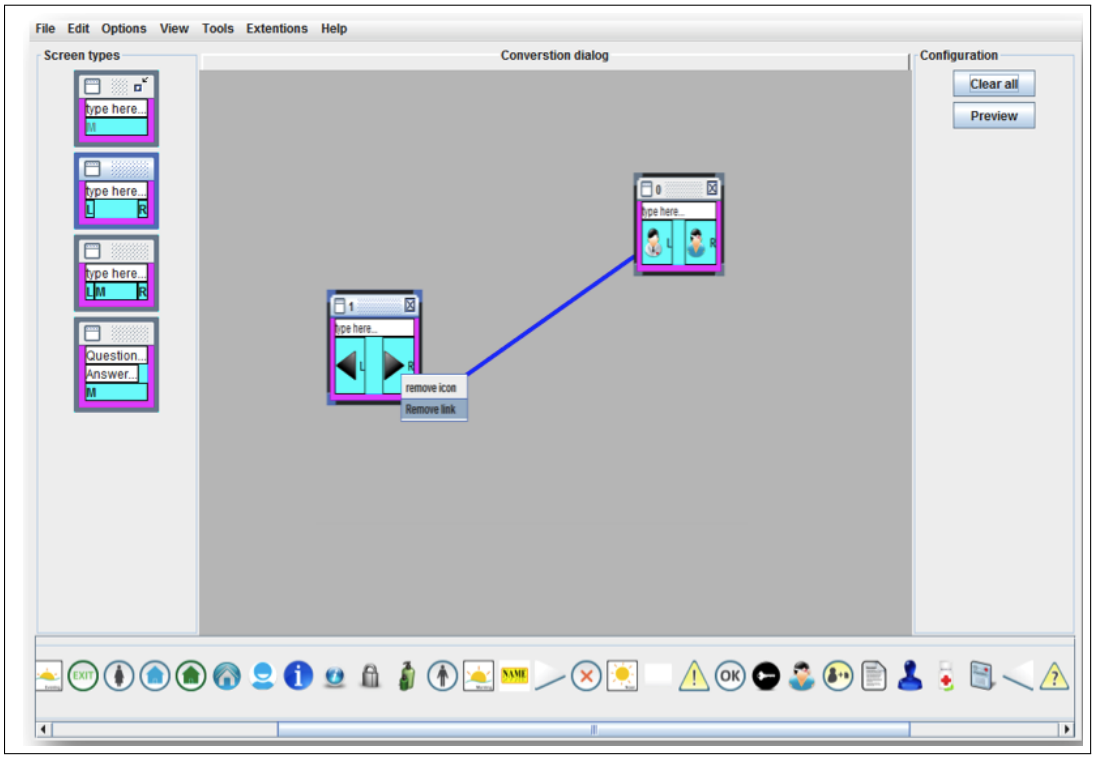

Figure 4.1: A screen shot of the first authoring tool prototype while a user (domain expert participant) is testing it..

The purpose of this iteration was to invite domain experts to take part in designing the authoring tool. By first trying out the current authoring tool then give feedback on its usability. The expected feedback from the participants is about how can we improve the authoring tool interface in a way that all its functions are visible and that it's easy to use. After the first prototype was implemented, we then evaluated it's interface design for usability with Deaf domain experts.

4.1.2 Experiment design

4.1.2.1 Sample

This experiment has 5 Deaf people as participants. All of them work at DCCT, and their job description includes empowering other Deaf clients. The researcher believes that these are the best participants for this session because they experience communication barriers in their daily lives and hence they know the communication needs of Deaf people. We also had an SASL interpreter, who assisted in translating information between the Deaf participants and the researcher.

4.1.2.2 Procedure

The purpose of this session is to test the current features of the authoring tool for usability. Features include dragging and dropping icons to screens, adding and removing screen data, links and also different mouse click features. Testing these features will give an idea of what needs to be improved. The procedure for this session was initially divided into three stages, namely training, testing, and feedback. However, it ended up having training and feedback sessions only. This is because only 1/5 of the participants managed to finish building the training scenario. Hence it was concluded that it would be less useful to progress to the testing stage as the training results indicated that the prototype is difficult to use. Here are the details of the training and the feedback stage below:

4.1.2.2.1 Training The training stage was divided into 4 parts, as shown in Figure 4.2.

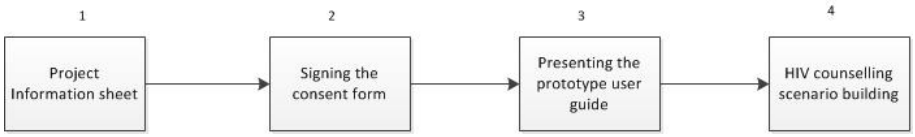

Figure 4.2: Steps of the training stage.

Information sheet and Consent form: An interpreter presents the project information sheet to the Deaf participants in SASL. This information sheet informs the participants about the aims of the authoring tool, how it is built and how the participants will contribute to the project. The information sheet also informs the participants that their contribution is voluntary and that all information collected from them will be treated with high confidentiality. After the information sheet has been signed by the participants, they then sign a consent form to witness that they agree to participate in the session.

Prototype user guide: The prototype user guide is presented to the participants in a PowerPoint presentation. The user guide shows screenshots of the authoring tool features as they are being explained. The purpose of the screenshots is to ensure understanding and also to convey visual information. After presenting the user guide, the HIV counseling conversation scenario is then given to the participants to create using the authoring tool.

HIV counseling scenario: The HIV counseling scenario is presented in a binary tree flow diagram. It consisted of a series of questions that the HIV counselor would ask a client in order to determine whether s/he needs to get tested for HIV. The client is expected to respond by a yes or a no; so that the HIV counselor can decide which question to ask next.

4.1.3 Results

Only one participant managed to finish building the scenario using the authoring tool. This participant was also able to view the conversation scenario design in a PDF file and verified the relationship between the icons and the screens that they linked to. Other participants were confused by the scenario flow diagram, e.g. has difficulty in differentiating between the client and the HIV counselor; and the instructions that were given to them on how they should build this conversation scenario. One participant could not link the icons with the screens.

4.1.4 Analysis

The results indicate that the authoring tool prototype was not user-friendly. The number of participants that managed to complete the task given in the training scenario indicates this. The confusion from the diagram shows that the instructions given to the participants were not clear enough. More work needed to be done on the authoring tool interface in order to make it more user friendly and and less frustrating. We also needed to improve the way we explain the features of the authoring tool and how we present task instructions to the participants.

4.1.4.1 Plan for the second iteration

We will explain the features of the authoring tool by demonstrating them on a projector. Participants will also be given a chance to try each feature on their computer, as it was demonstrated on the projector. This will help the participants to have a better understanding on how to use the features of the authoring tool. The participants will be reminded about the current SignSupport app. All participants are familiar with this app, so it will be a good example for them. We can

then use the authoring tool to create, populate and link a few screens from this app.

The next section presents the second iteration of the authoring tool prototype. The feedback obtained from the first iteration was used to improve the authoring tool interface and the way that instructions are presented to the participants.

4.2 Second iteration

The participants are changed in this iteration, from Deaf DCCT staff to hearing domain experts. The main reason for this change is that we wanted to address the needs of professionals who are likely to have Deaf people as their clients. The idea here is that the domain experts design the domain specific scenarios and then the Deaf people, and the interpreters, help with SASL video recording. Domain experts knows the set of questions that they ask their clients in order to find ways to assist them and the challenge that arises when both the domain expert and his/her client do not have a language that they both understand. For example, a doctor has basic questions that they ask patients when they're trying to find the disease that they might have. The challenge that the doctor might have is when they have to ask these questions to a Deaf person who communicates in a different language and cannot read or write in a language that the doctor understands. This is the reason why we focused on hearing domain experts for this iteration [10]. See appendix F for more details.

4.2.1 Interface design

This section describes the features available in the authoring tool. The authoring tool helps a domain expert to design an interface for a limited domain conversation flow between a Deaf and a hearing person. The software presents the domain expert with screen templates. These screen templates can be populated with text, icons and/or signed language videos (that would be provided by native signed language speakers, not the domain experts, unless they are Deaf). The screens can also be linked to each other in a graph structure. The link between each screen indicates the user interaction on the real mobile app i.e. which screen will show next when the end user clicks a given button. Figure 4.2 presents a screenshot of the authoring tool interface while a domain expert is using it. The interface of the authoring tool is divided into four areas [10]:

1. The button area contains four buttons, namely Add template, Clear all, Add video and Export to XML. When the domain expert clicks on the Add template button, a window

pops up and presents all the default screen templates of the authoring tool. The user then chooses the template that s/he would like to add to the canvas by clicking the checkbox on the left hand side of the preferred template and then clicks the OK button at the bottom of the window. The selected screen template then appears on the canvas as active. The Clear all button clears all the screen templates from the canvas. The Add video button helps the domain expert add videos from the computer as an asset within the authoring tool. The videos line up vertically as they are being added from the computer to the video area. The Export to XML button invokes a method that reads all the screen templates, their assets and connections to each other; and then produces an XML file that includes all of this information. Since the XML file contains only text, it only stores an asset's information such as file name and relative path, in text. An XML parser on a target device consumes the XML file in order to render a platform-specific mobile app.

2. The video area contains a series of signed language videos that are added by the domain expert from the computer. The videos can be recorded before or after the scenario based conversation flow is designed. Then each video is given an English tag/file name as a way to help the scenario domain expert include the correct video in the right place [29]. Videos in this area are played (and paused) by clicking on them. They can also be dragged and dropped to other screens.

3. The icons tab area contains icons in various categories. The domain expert can also add new tabs of icons. The icons can be dragged and dropped to the screens, and linked to other screens and/or actions.

4. The Canvas area is where the selected screen templates appear when added. The domain expert populates a screen by typing in the intended message and dragging and dropping icons or a video to the indicated areas in the screen. If the domain expert drops an icon into the wrong area, s/he can right click on that area and select the remove icon option and the icon will be removed. To link the screens to one another, the domain expert clicks on the icon that represents a button from one screen, confirms that s/he wants to connect that icon to another screen at a prompt dialogue pop-up, then selects the screen to link with. A blue link line will appear in the Canvas, between these two screens. This link gets updated and redrawn as the screens are moved from one position to another.

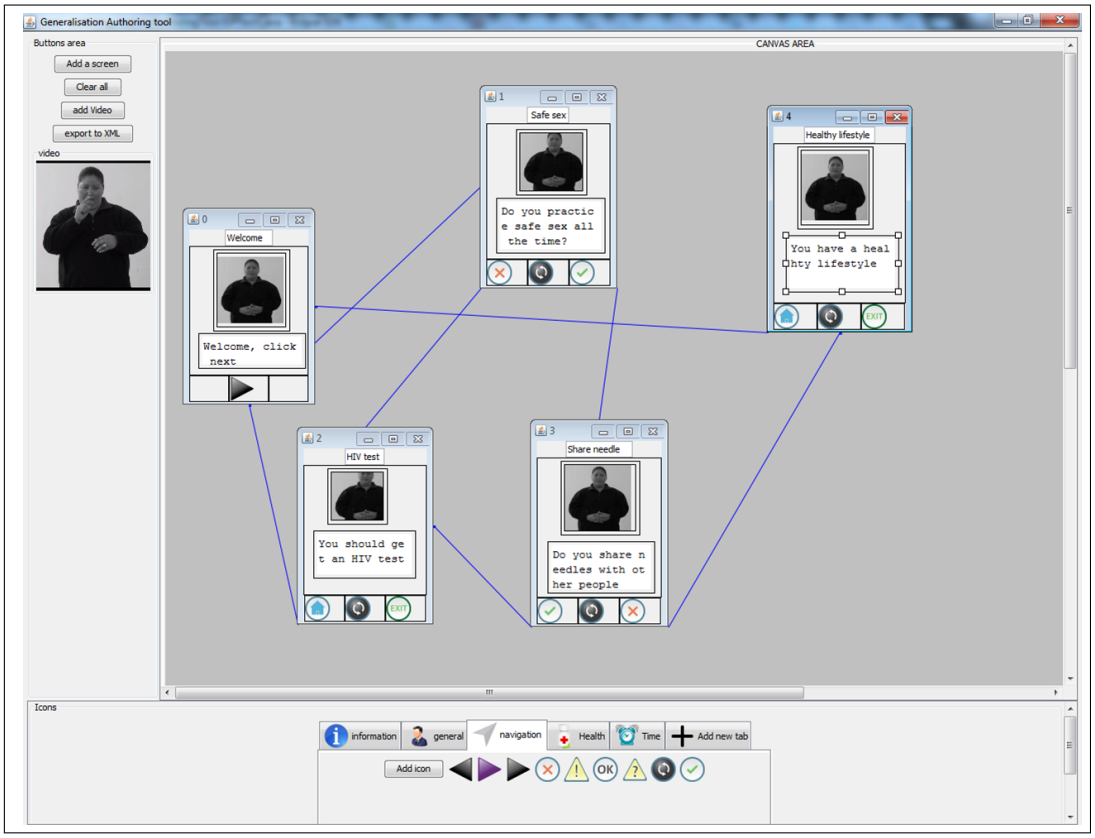

Figure 4.3: A screenshot of the second Authoring tool interface while a user (domain expert participant) is testing it.

The authoring tool is built using the Java programming language, and the output of the authoring tool is an XML file. The XML file serves as a database that stores the design information in text. XML is supported by a variety of technologies in various platforms. This in turn helps us to achieve platform independence for scenarios designed from the authoring tool.

4.2.2 Experiment design

User-based testing was employed for the usability experiment [14]. The goal of the experiment was to evaluate how effective the authoring tool can be in enabling a domain expert with little or no programming skills, to create a limited domain conversation scenario. The main focus is how the domain expert searches for icons from the icon tabs, adds and removes icons and videos from the screen templates, recovers from errors; reuses screen templates in the same scenario and reuses the same icons in different screen templates are also observed. The domain expert was also encouraged

to give suggestions on how the prototype can be improved to better suit their needs [10].

4.2.2.1 Sampling

Participants were selected through a purposive sampling method. Purposive sampling is purposefully selecting participants in terms of the qualities/skills they have [42]. The communication flow design scenarios for the experiment had already been defined, as the medication dispensing process and the (ICDL) course. The pilot session of the authoring tool was evaluated with the aid of an industrial design engineer. Participants recruited were a pharmacist for the pharmacy setting scenario, a educator of Deaf adults and a computer science researcher for the ICDL course scenario for the actual usability testing sessions. All participants have unique work experience with Deaf people. The industrial design engineer designed the SignSupport mobile app and the pharmacist is involved in the co-design and co-testing of the previous and current versions of the SignSupport mobile app with Deaf users. The educator of adults gives ICDL lessons in SASL every Thursday at a local Deaf community; and the Computer science researcher works on another authoring tool for creating ICDL lessons only [10]. All our participants are the potential users of this authoring tool and hence ideal candidates for the experiment.

4.2.2.2 Testing procedure

The testing was divided into three stages, namely training, testing and gaining user feedback. One participant attended the usability test at a time. During the training stage, the participant was introduced to the authoring tool prototype and all the features were demonstrated. The researcher then encouraged the participant to try out the features on their own e.g. add screens to the canvas area, drag and drop icons and videos to the indicated area, and link icon labels with other screens. Afterwards a training exercise was given to the participant to design a HIV counseling scenario. This applied to all participants. This scenario was given to the participants as a decision flow graph diagram, as shown in Figure 4.2 [10].

During the testing stage, participants were first asked to design and populate a limited scenario that was specific to their domain. The participants from the pharmacy scenario were asked to design a few screens from the familiar SignSupport mobile app for medicine dispensing. The participants from the ICDL course scenario were asked to design a specific lesson from the ICDL course book, which can be populated with SASL video instructions. The participants were assigned the tasks specific to their domain to give them a feeling of how they would use the authoring tool to best suit their needs. This session was video recorded. The video recorder was directed to the computer

screen to record how the participant interacted with the authoring tool. Participants were asked to use the think aloud strategy to voice out what they were doing and all the thoughts that came to mind as a result of using the tool [10]. After the testing stage, the researcher conducted a semi-structured interview with each participant. The main questions asked in the interview were:

- What did you like/dislike about the authoring tool?
- What did you find easy to use when building the scenarios?
- What did you find confusing or difficult?
- Any features you would like to be changed or added to the authoring tool?

Data collected from the trial ave summarised and organised according to the four areas of the authoring tool interface, which is stated and discussed in the results section below.

4.2.3 Results

All participants finished the training scenario successfully and with no difficulties. They all repeatedly used one screen template with the same icons for the training scenario design. All of them also managed to create and populate the testing scenario design. The participants feedback from both the pilot and the actual usability test session is described in accordance with the aforementioned four areas of the authoring tool interface features [10].

Buttons area Here is the description of feedback from all the participants regarding the available buttons in the button area. All participants understood the use of each button on the interface.

1. Add screen: All participants praised the interface and notification designed for the add screen and its process. The selecting process was easy to understand, the background colour of the selected screen template changed as it was clicked, and this assured the participants that the template was indeed selected.

2. The canvas was cleared immediately after clicking the Clear all button; and the XML output was created and saved to a file after clicking the Export to XML button. Two participants were also interested in seeing what the XML output looked like.

3. Suggestion on the additional button: Two participants said that there should be an Add picture button that they can use to add pictures to the screens, which works similarly to the Add video for adding videos to the screen.

Video Area

1. Adding video to the video area:
   All participants found the video addition process to the video area easy to do.

2. Suggestion of a scroll-bar:
   One participant suggested that there should be a scroll-bar in the video area, so that multiple videos which will be added to the video area can be seen.

3. Suggestion of video controls:
   One participant suggested that there should be video control buttons that show on each video in the video area, when the mouse is hovered over them, so that he could click on these video controls to play each video.

Icon tabs area All participants could navigate through the icon tabs easily. Names of all the icons appeared when the participants moved the mouse over them. Two participants commented that the tool-tip assured them that they were choosing the icon with the correct semantic meaning to drag to the screens. Two participants struggled to find the home and the exit icons as they expected to find these icons under the navigation icon tab instead.

4.2.3.1 Canvas Area

1. Adjustment of the screen and its component sizes: All participants disliked that the components of the screen remained in the same size and position when the screen was resized. They then had to resize and position each of them one by one. So, two participants suggested that the whole screen and its components should be proportionally resized all at once.

2. The design of the default screen templates: One participant found the default screen templates limited in terms of design. Most of the component areas were fixed in their position, which did not allow the user to modify the default design. One participant mentioned placing the next and back buttons on the sides of the screen, whereby the default screen template did not match this need. Therefore, she suggested having a blank screen template available for users to customise to their own screen design(s).

3. Screen links: All participants found the screen linking option easy to understand and use; however, they preferred that the linking process be done with fewer mouse clicks. The linking line should go around the screens instead of going underneath them. With the suggested linking line, it would be easier for the user to trace back and recheck each link. In addition, one participant suggested that the line indicating the link should have a message that indicates its origin and the destination of linking.

4. Drag and drop option: All participants said that they liked the drag and drop options for adding icons and videos to the screen. It was observed that all participants missed the icon drop area several times when they were populating the screens with icons. However, they tried again even though the authoring tool did not give them any warning/notification

Table 4.1 lists features that the participants suggested the authoring tool should have. Participants made these suggestions during the feedback session, when they were asked about what they would add to the authoring tool. These features will be added to the authoring tool and will also be evaluated for usability in the next iteration.

Table 4.1: Recommended features and their motivation

Feature	**Motivation**
Cut and paste	To duplicate screens that are already populated; and modify them for a similar purpose so that there wont be a need to start over all the time.
User guide and help function	To enable the author to learn to use the authoring tool independently and to get assistance in the absence of the researcher.
Tooltip text on Buttons and mouse-hovers	Tooltip text on a button or mouse-hover option reminds the author what each button or option is used for.
Scenario design preview	This gives authors an idea of how the scenario would look as a real app and also to confirm that the links have been defined correctly.
Undo and redo	This option helps to recover from errors or to repeat recent events.

4.2.4 Results analysis

The results indicate that there are several additional features required from the authoring tool to improve its usability. The robustness of the authoring tool needs to be thoroughly evaluated before taking it to its potential users for usability testing of the next iteration. The creation of the three limited domain scenarios (HIV pre-counseling, ICDL course lessons and rebuilding a thread of the SignSupport pharmacy app) demonstrates that the authoring tool can generalise SignSupport scenario creation for limited domain communication between Deaf and hearing people. The reuse of screen templates and data assets (icons and videos), combined with the production of an XML file representing a given scenario, appear to indicate that the authoring tool can generalise the SignSupport mobile app for multiple signed languages and mobile platforms [10].

The next iteration is for improving the authoring tool by fixing its errors and adding features recommended by the participants. Another usability testing exercise will then be conducted with more participants and one more limited domain scenario. Thereafter, one complete limited domain scenario will be built using the authoring tool, with recorded SASL videos with the help of a Deaf person and a SASL interpreter. That requires a fully functioning XML parser. The XML parser will enable the authoring tool to produce a mobile app that can be run on a mobile phone. Sign language videos will be recorded separately, and then integrated into the scenario.

4.3 Third iteration

Here we aim to confirm usability and that the participant see the authoring tool as useful to them. The last iteration focused on how the participants navigate through the authoring tool interface when building.

4.3.1 Interface design

We addressed the interface design issues that were raised in the previous iteration and we also added a few more features that optimise scenario creation and hints the author as they use the authoring tool. The buttons now have tooltip text, which appears on mouse hover. Tooltip text informs the author about the purpose of the button. The links between the screens are changed from lines to arrows. This enables the author to follow the flow of the scenario by following the arrows from one screen to another. The screens can also be cloned in order to avoid having to recreate similar screens. Other parts of the interface design remained the same, as the participants,

in the last iteration, were all satisfied with it. Here is the interface design of the authoring tool in Figure 4.4.

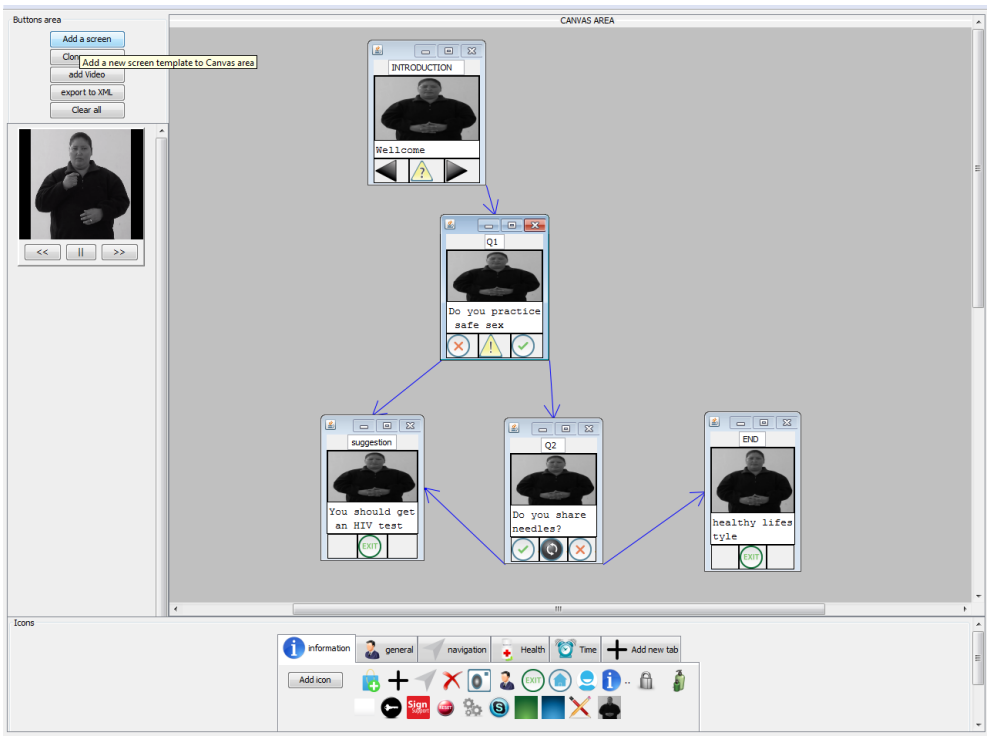

Figure 4.4: A screenshot of the new authoring tool interface with text tooltip showing on mouse hover and a clone button that is used to create a copy of the screen. The participant (domain expert) is testing it for usability.

4.3.2 Experiment design

We evaluate the authoring tool for usability using the user-based testing method. The aims of our experiment are:

- Confirm the authoring tool ease of use.
- Understand how the participants view the authoring tool.
- Give participants an opportunity to rate usability of the authoring tool
- To get recommendations on how we can improve the authoring tool in the future.

The experiment consists of three parts, which are the sampling, testing procedure and data collection.

4.3.2.1 Sampling

Purposive sampling is used again in this iteration. The participants are the computer science researcher, educator and the pharmacist; who participated in the authoring tool evaluation of the previous iteration. We also have two SASL interpreters, who have many years of experience interpreting for Deaf people and also working in various projects that empower Deaf people. The experiment consists of three scenarios: HIV pre-counseling, medical dispensing, ICDL course and antenatal care. The HIV pre-counseling scenario is used to demonstrate the authoring tool to the participants, the other three are used for the actual evaluation. For the convenience of this experiment, the pharmacist, educator, computer science researcher, the first and the second interpreter will be referred to as participant A, B, C, D and E respectively.

4.3.2.2 Procedure

This procedure has training and testing stages; and this session was conducted with each participant independently. In the training stage, we initially remind the participants what the project is about and what value s/he be adding to the project. The participant signs a consent form as an indication that s/he agrees to participate. Thereafter the researcher explains all the features that the authoring tool has, by adding screens, populating them with videos, text and icons; and linking them while the participant is watching. The participant is then given a chance to tryout each feature that the researcher just demonstrated.

The participant is then given an HIV pre-counseling scenario in the form of a printed diagram on paper. The researcher explains the flow of the scenario diagram to the participant before s/he can get to the authoring tool. This helps to ensure that the participant understands the scenario design and its conditional branches. The participant is allowed to ask for help and to ask questions at this stage since the aim here is to teach him/her how the authoring tool works.

The testing involves building a different scenario independent of the researcher's help. Here the participant is asked to use the think aloud protocol as s/he is building the scenario. Participant A builds the medical dispensing scenario; B and C build the ICDL course scenario as in the previous session. Participant D and E build the antenatal care scenario.

4.3.2.3 Data collection

This experiment is recorded with a video camera, which is pointed to the computer screen as the participant is interacting with the authoring tool. The video camera records the participants

voice as s/he uses the think aloud protocol. A semi-structured interview is conducted with each participant when they finish building the testing scenario. Here are some of the questions asked in this interview:

- What did you like about the authoring tool?
- What did you not like?
- What would you like to change?
- Can you explain the authoring tool in your own words?
- Please rate the authoring tool's usability from 0 - 10
- If the authoring tool was a car, which car would it be, and why?
- How does this authoring tool compare with other similar tools that you know?
- Would you recommend this authoring tool to other professionals who have contact with Deaf clients?

All these questions are asked in order to ensure that the participant understands the purpose of the authoring tool and to get feedback on whether they think the authoring tool is useful. The results are stated and discussed next.

4.3.3 Results

The results are divided into two parts, which are the interface design and participants' response to the interview questions. The results of the interface design are presented according to the sections of the user interface. The responses of the participants are presented according to the questions.

4.3.3.1 Interface design

- **Buttons area:** All the participants added the screens successfully by clicking the "add screen" button. The researcher observed that 2 participants clicked on the screen multiple times instead of clicking the checkbox area. Only 3 participants used the "Clone screen" button to replicate the screens. One of these participants couldn't see cloned as she expected it to appear next to the original one. This made her pause for some time and search for the cloned screen around the canvas. Adding a video from the computer to canvas, using the "Add video" button was simple. The only issue was that the videos took long to be loaded to the video area such that the whole authoring tool becomes non-responsive. One participants suggested that multi-threading should be used so that the user can do other tasks

in the authoring tool while the video is loading. The "clear all" and the "Export to XML" button worked well when they were used. One participant suggested that there should be a progress bar that appears as the XML file is being created; and lists all the steps taken to create an XML file. This keeps the user informed about the progress of creating an XML file.

- **Video area:** Videos had playback controls (rewind, play/pause and forward) which showed at the bottom centre of each video. One participant suggested that the names of the videos should appear on mouse hovers so that they can remember what the video is about, especially if it's a signed language video.
- **Icon tabs:** All participants could navigate through the icon tabs and could find their desired icons for the screens that they had. 3 participants praised the drag-n-drop feature for the icons.
- **Canvas:** All participants could easily build and finish building scenarios on the canvas. This includes adding screens and populating them with data and linking them. It was observed that participants could easily move, resize and delete screens that are in the canvas area. Deleting icons, links and videos from the screens was also easy to do, as observed by the researcher. One participant mentioned that she finds it difficult to remember when to left or right click when adding a link or deleting an icon. Removing the link was confusing at times, as the 3 participants try to click on the link arrow in order to delete it. However, 2 participants, saying that they make the scenario design easy to follow, praised the arrows that represent the links. The colour of the canvas area was too similar to the added screens, as said by one of the participants. She then asked that the screens and the canvas area should have distinct colours. One last suggestion for the canvas area was that there should be a space where the authoring tool user can write the title of the scenario.

4.3.3.2 Participants' view of the authoring tool

The participants were asked to define the authoring tool in their own words. The aim of this question was to reveal their understanding about the authoring tool. Their definitions are represented in Table 4.2.

The definitions given by the participants indicate that they understand that the authoring tool aims to empower Deaf people in multiple communication scenarios between themselves and hearing people. Two participants also mentioned, in their definition, that the scenarios created on the authoring tool will be installed on the mobile phone for Deaf people to access. These definitions can be summarised into one definition: The authoring tool helps a domain expert

Table 4.2: Participants' definitions of the authoring tool

Participant	Definitions
A	Allow you to independently create a conversation scenario for a Deaf and hearing person. This is achieved by populating screens with information and then link them according to the screen options.
B	Build customized lessons for computer skills training, so like you can build sign language lessons for Deaf people so that they are appropriate and user-friendly them.
C	Structure content for Deaf people to consume on a mobile phone. You can also build different communication scenarios for Deaf people.
D	Used to create multiple scenarios to accommodate Deaf people, e.g. Deaf education or scenarios that raise awareness for Deaf people in any sign language. This tool also benefits hearing people who have Deaf clients in their professional lives.
E	Create scenarios for the Deaf and hearing people to communicate when the interpreter is not present. The scenarios that the tool creates are stored on a mobile phone and acts as an interpreter on a Deaf persons mobile phone.

to create multiple communication scenarios, in multiple sign languages, for Deaf people to access on their mobile phones. This is similar to the research question of this project, stated in Chapter 3.

Participants gave the authoring tool a score (0 - 10) for usability, as indicted in Figure 4.5.

The average of these scores is 7.1. This clearly means that the participants are satisfied with the user interface design of the authoring tool. The general comment was it is simple and very easy to use; and all the features are visible.

Below is Table 4.3 that represents the participants' responses when they were asked to familiarise the authoring tool with a car or a form of transportation that they know. This question was asked to understand the participants' perception of the authoring tool.

Table 4.3: Authoring tool related to cars.

Participant	Transport	Motivation
A	VW Golf	The tool is reliable and will gain popularity as time goes by.
B	Qhubeka Bicycles	Very basic and works really well.
C	Citi golf	Basic tool with normal functions to take you from point A to B. Not fancy just the main purpose.
D	New avanza	All the features are visible and it aims to accommodate multiple signed languages and mobile phones.
E	Old 5-gear car	Easy to use and it does everything as expected.

Participant A believes that the authoring tool will be widely known for its reliability and usefulness in the lives of Deaf people. She also mentioned that the authoring tool needs to be advertised to the officials who specialises with content delivery to Deaf people (DeafSA and other Deaf NGOs), she feels that those are the people who will find the authoring tool useful. Domain experts like pharmacists and police might be resistant towards using the authoring tool as they might feel that it is not their "job" to ensure that Deaf people understand them. Participant B finds the authoring tool very basic but working very well. She says that all the features of the authoring tool makes it so easy to create a scenario in a way that one can create the scenario on the authoring tool, without having to draw it out on paper first. Participant C and E feel that they authoring tool fulfills its purpose but it can be improved by adding text or multimedia formatting features. Participant D likes the idea of accommodating multiple signed languages and mobile phones although she was disappointed that the authoring tool does not produce a real mobile app.

4.3.4 Analysis

The participants' feedback seems to be more and more positive as the authoring tool is being tested and improved; from one iteration to another. The interface design of the authoring tool seems to have standardised as well as the participants didn't mention any part that needs to be changed. The next step is to add more features whiles trying to keep the interface design the same.

The next section demonstrates the steps needed to create an app by using an authoring tool and an XML parser.

4.4 Fourth iteration

This iteration uses an authoring tool to design an ICDL exercise app and then using the XML parser to render its XML output into a mobile app.

4.4.1 Process

Video recording and editing The authoring tool requires that the domain expert should have the SASL videos already recorded. So the first step was to write down the exercise questions and then record SASL videos for each of them. The videos were recorded with the help of one of the Deaf learners and an educator. The educator read the questions in English text and then signs the question to the Deaf learner, and then the Deaf learner would sign the question to the video camera. The reason we asked an educator instead of a SASL interpreter is that sign language has different signs for one item and so a SASL interpreter might use a sign that the target Deaf learners might not be familiar with. This might lead to confusion and hence affect the learners' grades when they do the exercise.

After recording the videos, we needed to edit them and prepare them to be used in the mobile app. Sound was removed completely from the videos as it was not needed. We then changed the videos to black and white so that they take less memory on the mobile device. Lastly, we cropped the videos to a portrait orientation so that they are not squashed but appear clearly on a mobile phone's screen. Each video was given a unique name, which was the keyword of the question e.g. Bold, Underline or Save.

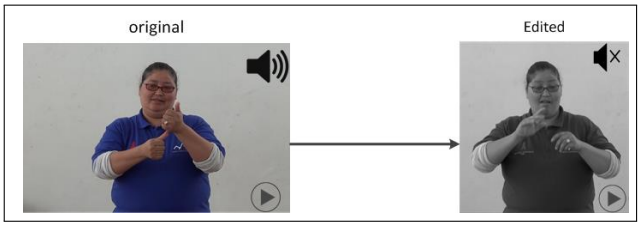

Figure 4.6: Signed language video editing
The original video in the left was edited to black and white to reduce its size; cropped around to best fit a mobile phone's screen and also had its sound muted.

Scenario creation We imported all the ICDL videos to the authoring tool. We then added screens to the canvas area and populated them with icons and videos. Each screen was given a

unique name, which was the keyword of the question e.g. Bold, Underline or Save; and had two navigation icons ("back" and "next"). So the screen that represents current question links its "back" icon with the screen that represents the previous question; and then links its "next" icon with the screen that represents question that follows. Each screen also had a text field that states the question that is being signed in the SASL video; in English text. The English text would be helpful to verify that the correct video is used for all the questions.

The "Export to XML" button in the authoring tool to get the XML file that represents the ICDL exercise design, from the authoring tool. We then copied the XML file from our computer onto a mobile phone to be consumed by the XML parser (see appendix D for algorithm). We also copied the necessary SASL videos and the icons to the mobile phone so that the XML parser can find them when generating an app.

4.4.2 Results

The XML parser uses fixed file path to get the file and its assets. Every time the researcher needs to launch the generated app, he had to launch it via launching the XML parser. This is because the XML parser rebuilds the app every time one clicks on the XML file. The XML parser does not produce the source code for the rendered application so you cannot edit its interface design. The only thing that can be changed is the XML file. The XML parser also assumes that the scenario screens follow each other sequentially. This might not always be the case because in some scenarios, the author might want to link screens that are far apart but the XML parser will not cater for that.

4.4.3 Analysis

XML parser seems to have its own predefined interface for rendering apps and it then uses the XML file as an address to get the necessary assets to be included in the rendered app. It does not give an option for the user to modify the app's interface design or source code. This has potential problems because the rendered app can neither be edited nor installed to other mobile phones because it depends on the XML parser since it is rebuilt every time the XML parser is launched and that the XML parser does not generate source code for the app.

4.5 Discussion

The interface of the authoring tool improves from one iteration to another. This can be witnessed by the screenshots shown in the interface design of each iteration. The failure of the first iteration was a big learning curve for the researcher and it helped him to prepare better for the other iterations. The second and the third iteration stimulated creativity within the participants, as they were able to think of other features that can be added to improve the authoring tool interface. Using the SignSupport mobile app helped the participants understand better the purpose of the authoring tool.

Some of the interesting observations were that in the usability session of the first iteration, we had five participants in one room but only three of them gave feedback; and the other two remained silent up to the end of the session. The procedure was changed in the second and the third iteration in a way that each participant evaluated the authoring tool prototype at their own time. All participants gave feedback freely and even gave recommendations on how the authoring tool can be improved. This can be said to be a proof that conducting usability sessions with one participant at a time is more beneficial than having more then one participant in the same room. Applying the Think aloud protocol would have been impossible as well, if we had more than one user at the same time. Participants were not happy when the authoring tool didn't show them a preview of their scenarios after creating them; or generated an app that they can install on their mobile phones.

The three main research questions seemed to have been answered in the results. The multiple scenario creation can be witnessed by the success of participants' scenario creation task. The scenarios created had unique content, in videos and text. This can be said to address language independence because all spoken languages can be represented in text and the content of the specific signed language can be recorded into videos. Cross platform independence relies mostly on the XML parser as the authoring tool only produces an XML script.

4.6 Summary

Results were not positive in the first iteration but they became better in the second and third iteration. The authoring tool user interface in the first iteration could only accommodate icons and text as the scenario content. The user interface in the second iteration was more visual and

could also accommodate video content within scenario creation. The user interface in the third iteration had tooltip text and the links between the screens were arrows instead of lines. The arrows made it easier for the domain experts to follow the scenario flow. The domain experts were mainly hearing people who work or do research for Deaf people. The input from the participants exposes what needs to be improved in the authoring tool interface. The next chapter concludes the research and provides direction for future research.

Chapter 5

Conclusion

The research conducted herein leveraged multiple approaches to content authoring and generalisation; and further that software generalisation can improve accessibility and affordability for the ultimate end users. The research aimed at generalising the SignSupport for multiple scenarios, languages and mobile platforms. Four iterative steps were taken (building a prototype, testing it for usability and using feedback, from the participants, to improve the prototype for the next step), guided by Iterative software development and Action research method, in order to achieve this. Generalisation techniques were investigated in related work and some were adopted in carrying out this research and building the authoring tool. Data collection focused on usability and scenario creation; and was done in three iterative steps. Data collected in each step was used to improve the authoring tool for the next iteration. The last step of the research was to design and render an ICDL scenario into a real mobile app. This was done in order to create a step-by-step process on how to design and render a scenario into a real app; using the authoring tool and an XML parser.

The results from the first iterative step were that the authoring tool interface was not user friendly for the participants. The authoring tool was then redesigned into a more visual interface, which mainly used point-and-click and drag-and-drop functions. The second iterative step was conducted with hearing domain experts who work with Deaf people. The scenarios created were HIV pre-counseling, medical dispensing and ICDL course. Results showed that the authoring tool can generalise SignSupport for multiple scenarios as all these scenarios were created successfully with the authoring tool. The feedback from the participants was mainly about the interface design that needed to be improved. Some of them also gave recommendations on features that can be added to the authoring tool. The third iterative step was more about acceptance testing and usability. The questions asked to the participants were aimed at ensuring that the participants

understand the purpose of the authoring tool and the challenges that it tries to tackle. For example, participants were asked to explain the authoring tool in their own words and to compare it with other similar software that they know. The results from this iteration proved that the participants approve the authoring tool as useful, and sees it as a powerful tool in the future. The fourth iteration demonstrated the steps on building a limited domain scenario app, using the authoring tool and an XML parser.

5.1 Answering the research questions

The main research question is: How can we generalise SignSupport for multiple limited domain scenarios, signed and spoken languages; and mobile platforms? This question was then broken into three parts, as presented below.

5.1.1 Limited domain scenarios

The current SignSupport mobile apps only accommodated one scenario (medical dispensing and computer literacy training). The approach used in the authoring tool in order to enable multiple scenario creation was to have customisable templates that can be populated with scenario specific content such as text, pictures, icons and videos. These templates can also be linked together in order to define a flow of the scenario. The content differentiates the scenarios from each other but the templates remained the same.

5.1.2 Signed and spoken languages

Text, pictures and icons represent the spoken languages in the scenario created; and text is mainly for hearing experts who cannot understand signed language. A scenario designer can represent any spoken language with text. Any signed language can also be recorded into videos for Deaf people to place back later. Representing signed and spoken languages with videos and text; in turn, makes the SignSupport language independent.

5.1.3 Mobile platforms

The scenario designed in the authoring tool is converted into XML. XML contains the information about the templates used in the scenario and the data which they were populated with. XML is a cross-platform language that is commonly used for data transportation and database representation. An XML parser, used in the fourth iteration of the previous chapter, uses the XML produced

by the authoring tool to render a platform dependent app. Hence, the XML parser is outsourced in order to accommodate multiple mobile platforms.

5.2 System specifications and limitations

This section discusses the specifications of the authoring tool, from the final iteration, and its limitations.

5.2.1 System specifications

The authoring tool has default screen templates that the author can add and populate with text, icons or videos; in the canvas area. These templates can also be linked together to define a flow of the scenario. One icon on a screen is linked to another screen to indicate which screen will show next if that icon is clicked, when the scenario has been rendered into a real mobile app. It was observed from SignSupport that the scenario is not always linear and that a users chosen option, in the current screen, determines which screen will show next. One example could be if the user chooses the female icon in the screen that asks for gender, the screen that will show next will ask if the user is pregnant, otherwise a different screen will show. Here is a summary of other system specifications:

5.2.1.1 Button area

The button area has five buttons, namely: Add screen, Clone screen, Add video, Export to XML and Clear all. Add screen button is used to add screen templates to the canvas area. When the author clicks on this button, a dialog box appears that shows five different screens, the first four screens were most repeatedly used in the SignSupport mobile app. The fifth screen template is an empty screen that allows the author to design a new screen from scratch, by choosing elements (labels, buttons, radio buttons, text-box, check-boxes etc.) to add onto that screen. Clone screen allows the screens to be duplicated by first selecting the screen, on the canvas, to clone and then clicking the "Clone screen" button. Add video allows the author to import a video from the computer to the authoring tool. Export to XML button gathers all the content of the scenario in the canvas area and creates an XML file that contains the same content and structure as that scenario. This XML file is then used by the XML parser to render the scenario into a real app, as explained in the fourth iteration of the previous chapter. The Clear all button clears the canvas area by removing all content that the canvas area contains.

5.2.1.2 Video area

The video area shows a vertical list of all the videos that have been imported from the computer to the authoring tool. Here, each video has its own controls, so the author can play/stop, fast forward or rewind a video. This is useful in ensuring that the correct video was imported.

5.2.1.3 Icons area

These are the default icons that the authoring tool comes with. They were taken from SignSupport. The icons are divided into different categories: information, general, navigation, Health and Time. All icons in these tabs are added to the screens through drag-and-drop. The author can also add a new tab or icons if s/he would like them to be in the scenario. All the newly added icons can also be added to the screens in the same way as the default ones.

5.2.1.4 Canvas area

This is the area where the scenario creation happens. Screen templates are added and populated in this area. Screens can be removed individually by clicking the button on the top-right-hand corner of the screen. Icons and links added to the screens can be removed by right clicking on the icon and selecting an option to remove an icon or link. This area has a vertical and a horizontal scrollbar, which can be used to show extra space on the canvas area if the visible area is not enough for the scenario being created. In order to remove every element on this area, you use the Clear all button, as explained in the button area.

5.2.2 System limits

The authoring tool has a number of limitations, although it has proven to be very effective in enabling the author to create a limited domain scenario. Here are some of the limitations of the authoring tool:

5.2.2.1 OS dependent

The authoring tool was developed and tested in a windows computer. All the user trails were also conducted in a windows computer. The authoring tool showed negative results when it was installed in another operating systems: the interface of the authoring tool changed and was less user friendly; and the drag-and-drop process was not smooth. It can be concluded from this that the authoring tool works more effectively when running on windows. However, this can be changed in the future by making the authoring tool platform independent.

5.2.2.2 Java dependent

The authoring tool is mainly built using Java programming language, videos are played using the library called VLCj and the output of the authoring tool is an XML file. VLCj enables the authoring tool to play modern multiple video formats because the generic Java media libraries have limitations. The XML file serves as a database that stores the design information in text. This file contains all the data from screen templates like text instructions, icons and video file names. The author needs to install Java in his/her computer before s/he can run the authoring tool. VLCj library is made part of the authoring tool but it also depends on Java.

5.3 Lessons learned

5.3.1 Research with human participants

There a number of formal steps that need to be followed when conducting research with human participants. Firstly, the research project needs to get ethics clearance. The ethics clearance is a guidance that ensures that the research results in benefits and minimises harm to the participants. This includes ensuring that the participants are well informed about their role in the research and how their information is going to be used. This helps the participants to make an informed decision on whether they should participate in the research. Ethics clearance also protects the researcher should the participants make a claim against the research.

Here are some of the accepted ethical standards:

Consent form: informs the participants about the research and whether the research is voluntarily or they will be paid for participating. They are also informed about the risks and benefits of this research to them. The consent form is necessary because the participants have the right to know who could have access to their information and what is their role in the research.

Confidentiality: Everyone have a right to privacy, according to the Constitution of South Africa. Participants have the right to know how their information is going to be stored, used and presented in the final report.

5.3.2 Generalisation activities

Generally, software is built to run on a specific platform and hardware. If people see that software as useful, then they see the need to buy that specific device in order to have access to the software. The challenge comes when one cannot afford to buy that device. This clearly means that person would have no access to that software. So software generalisation helps to increase its accessibility.

Software generalisation can be done to achieve a number of goals. Some of the common goals for generalisation include hardware, Operating system and human language independence. Hardware independence is about making software to run on machines with different processing capabilities or created by different manufactures. Operating system and human language independence includes creating software can be installed in different operating systems and usable by end-users who speaks different languages respectively. An example of language independent software is the ATM software. This software enables its target users to choose the language that they best understand in order to do banking.

There are number of approaches to achieve generalisation. The same software can have different flavours for different operating systems. This means that a computer with any operating system can install that software and the end-user will have access to all its features. This activity takes time because for the same software needs to be created from scratch for each operating system. Another approach is by creating a web app, which can be accessed by any computer with a web browser. The advantage of web apps is that they are device independent. Their disadvantage however, is that they have limited access to the device's hardware features. The generalisation approach used in this research is creating an authoring tool that a domain expert can use to design a front-end interface for a communication flow between a Deaf and a hearing person; in any limited domain scenario. The authoring tool then produces an XML script that can be rendered by an XML parser into a platform dependent app. This approach has two major advantages: it achieves technology accessibility for Deaf people because they just need any mobile device with a bigger screen and high resolution, so that the signed language videos are clear to see. Another advantage of this approach is that it empowers the domain experts and Deaf people. Domain experts benefit in this by communicating better with their Deaf clients, and Deaf people also play a role in the signed language video recording and verification.

5.4 Recommendations and future work

This section discusses the features of the authoring tool that can be implemented in the future and approaches to conduct research under assistive technologies for Deaf people.

5.4.1 Authoring tool features

A scenario preview could be made the same as the PowerPoint slideshow. The author can show which screen should the slideshow begin at. The icon(s) clicked in the currently displayed screen will determine the flow of the scenario preview. This will help the participants confirm their scenario design and the links between the screens.

Screen templates that the authoring tool provides can be customised in different ways to suit the needs of the author. Having to re-customise the screen template every time you create a scenario might have a negative impact on the author. A screen palette can be developed in the future where customised screens can be pulled to, and saved for future use. This will optimise the time an author takes to finish creating the scenario. This also gives a sense of ownership to the author because s/he can either modify or create new screen templates that suit personal needs.

The current authoring tool cannot save a scenario for later retrieval. This makes it difficult for an author who would like to close and continue with the scenario later. A Save and retrieve feature could be implemented to allow the author to save the current state of the scenario and retrieve it later. An XML file produced by the authoring tool could also be used in order to trace back the state and the content of the scenario when retrieved later.

The authoring tool connects to a webcam. The webcam could be used to record signed language videos while a scenario is being designed. A Deaf person or any person that can sign fluently could do this. The recorded signed language videos could also be edited within the authoring tool and then added to the scenario being created. Once the scenario is complete, it can be installed to a mobile phone, together with the recorded signed language videos, and be made available for use immediately.

5.4.2 Conducting research for Deaf people

The researcher helped with ICDL classes every Wednesdays or Thursdays at DCCT. This helped him to engage with Deaf people and understand the culture and appropriate approach when communicating with Deaf adults. This is important for a researcher to have this background knowledge about Deaf people as they interact differently than hearing people. One example is that looking at a Deaf person in the face means that they have all your attention.

Conducting a research session with a Deaf people requires that you hire a SASL interpreter. The interpreter should be qualified to interpret for Deaf people. It is best to ask the Deaf community to recommend interpreters for you so that you are sure that the interpreter understands all the dialects that your Deaf participants use. Institutions need details of the interpreter before they can pay them and some institutions only pay on a certain date of the month. It is important to get all the necessary details from the interpreter before the day of the research; and also inform the interpreter if your institution pays salaries only on specific dates of the month.

The prototype that you create needs to be tested by the actual target users so that you can measure its usability. If your target users are Deaf people then you need to ensure that your prototype interface is visual and has minimal text. It is also important to include the stakeholders in all the steps of your research so that they can give you feedback early. They can help to verify your research plan and help you rehearse your research sessions. This will help you to adjust your plan early and avoid mistakes. Presenting your research plan to a person who has experience working with Deaf people can also give you valuable feedback.

Bibliography

[1] Avison, D. E., Lau, F., Myers, M. D.,& Nielsen, P. A. (1999). Action research. *Communications of the ACM*, 42(1), 94-97.

[2] Barrett, H. C. (2000). Create your own electronic portfolio. Learning and leading with technology, 27(7), 14-21.

[3] Basil, V. R., & Turner, A. J. (1975). Iterative enhancement: A practical technique for software development. *IEEE Transactions on Software engineering*, (4), 390-396.

[4] Bassil, Y. (2012). A simulation model for the waterfall software development life cycle. arXiv preprint arXiv:1205.6904.

[5] Blake, E., Tucker, W., & Glaser, M. (2014). Towards communication and information access for Deaf people. SACJ, (54), 10–19

[6] Bourret, R. (1999). XML and Databases. http://www.rpbourret.com/xml/XMLDatabasesProds.htm

[7] Burgstahler, S. (2009). Universal design of instruction: Definition,principles, guidelines and examples. Retrieved from www.washington/doit

[8] Cavender, A., Trewin, S., & Hanson, V. (2014). Accessible Writing Guide. Special Interest Group on Accessibility in Computing. Retrieved from http://www.sigaccess.org/welcome-to-sigaccess/resources/accessible-writing-guide/

[9] Chininthorn, P., Glaser, M., Freudenthal, A., & Tucker, W. D. (2012). Mobile Communication Tools for a South African Deaf Patient in a Pharmacy Context. In P. Cunningham & M. Cunningham (Eds.), *Proc. IST-Africa*. Dar es Salaam, Tanzania; Dublin: IIMC International Information Management Corporation.

[10] Duma, L., Chininthorn, P., Glaser, M., & Tucker, W. D. (2015). Usability of an authoring tool for generalised scenario creation for signsupport. In Proceedings of the *Southern Africa Telecom-*

munications Networks and Applications Conference (SATNAC), Hermanus, South Africa (pp. 225-260).

[11] Farkas, K. I., Flinn, J., Back, G., Grunwald, D., & Anderson, J. M. (2000). Quantifying the energy consumption of a pocket computer and a Java virtual machine. *ACM SIGMETRICS* Performance Evaluation Review, 28(1), 252–263.

[12] Glaser, M. (2000). A field trial and evaluation of Telkoms Teldem terminal in a Deaf community in the Western Cape. In *2nd South African Telecommunication Networks and Applications Conference*. Stellenbosch, South Africa: Pretoria: Telkom.

[13] Glaser, M., & Tucker, W. D. (2004). Telecommunications bridging between Deaf and hearing users in South Africa. In *Conference and Workshop on Assistive Technologies for Vision and Hearing Impairment (CVHI)*. Granada, Spain.

[14] Glaser, M., & Lorenzo, T. (2006). Developing literacy with Deaf adults. In B. Watermeyer, L. Swartz, T. Lorenzo, M. Schneider, & M. Priestley (Eds.), *Disability and social change: A South African agenda* (pp. 192–205). Cape Town, South Africa: HSRC Press.

[15] Gorman, W., Oster, S., & Kniffin, B. (2001). U.S. Patent Application No. 09/917,435.

[16] Hansler, S., & Glas, B. (2011). Universal Design in Housing. Age in Action, 26(2), 1.

[17] Hayes, G. R. (2011). The relationship of action research to human-computer interaction. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 18(3), 1–5.

[18] Hayes, G. R. (2012). Taking action in your research. Interactions, 19(4), 50–53.

[19] Heitktter, H., Hanschke, S., & Majchrzak, T. A (2013). Comparing Cross-Platform Development Approaches for Mobile Applications. *Web Information Systems and Technologies*. Springer Berlin Heidelberg, 120–138.

[20] Holzinger, A. (2005). Usability engineering methods for software developers. *Communications of the ACM*, 48(1), 71–74.

[21] Hutchins, E. L., Hollan, J. D., & Norman, D. A. (1985). Direct manipulation interfaces. *Human Computer Interaction*, 1(4), 311–338.

[22] Ishizaki, K., Takeuchi, M., Kawachiya, K., Suganuma, T., Gohda, O., Inagaki, T., ... & Ogasawara, T. (2003, October). Effectiveness of cross-platform optimizations for a Java just-in-time compiler. In *ACM SIGPLAN* Notices, 38, 11, (pp 187–204). ACM.

[23] Juntunen, A., Jalonen, E., & Luukkainen, S. (2013, January). HTML 5 in Mobile Devices–Drivers and Restraints. , *46th Hawaii International Conference on In System Sciences (HICSS)*, (pp. 1053–1062). IEEE.

[24] King-Sears, M. (2009). Universal design for learning: Technology and pedagogy. *Learning Disability Quarterly*, 199–201.

[25] Lau, Y. K., Cassidy, T., Hacking, D., Brittain, K., Haricharan, H. J., & Heap, M. (2014). Antenatal health promotion via short message service at a Midwife Obstetrics Unit in South Africa: a mixed methods study. *BMC pregnancy and childbirth*, 14(1), 284.

[26] Laufer, L., Halacsy, P., & Somlai-Fischer, A. (2011, May). Prezi meeting: collaboration in a zoomable canvas based environment. In *CHI'11 Extended Abstracts on Human Factors in Computing Systems* (pp. 749–752). ACM.

[27] Looijesteijn, K. (2009). The design of a Deaf-to-hearing communication aid for South Africans. MSc thesis, Delft University of Technology, The Netherlands

[28] Mace, R. (1997). What is universal design. The Center for Universal Design at North Carolina State University. Retrieved November, 19, 2004.

[29] Motlhabi, M. B., Tucker, W. D., Parker, M., & Glaser, M. (2013a). Improving Usability and Correctness of a Mobile Tool to help a Deaf person with Pharmaceutical Instruction. *In Proc. DEV-4* (Article 13). Cape Town; New York: ACM Press.

[30] Motlhabi, M. B., Glaser, M., Parker, M., & Tucker, W. D. (2013b). SignSupport: A Limited Communication Domain Mobile Aid for a Deaf patient at the Pharmacy. In R. Volkwyn (Ed.), *Southern African Telecommunication Networks & Applications Conference* (pp. 173–178). Stellenbosch, South Africa: Pretoria: Telkom.

[31] Mutemwa, M., & Tucker, W. D. (2010). A mobile Deaf-to-hearing communication aid for medical diagnosis. In D. Browne (Ed.), Proc. *Southern African Telecommunication Networks & Applications Conference* (pp. 379–384). Stellenbosch, South Africa; Pretoria: Telkom.

[32] Myllymaki, J. (2002). Effective web data extraction with standard XML technologies. *Computer Networks, 39*(5), 635–644.

[33] Ngethe, G. G., Blake, E. H., & Glaser, M. (2015). SignSupport: A Mobile Aid for Deaf People Learning Computer Literacy Skills.

[34] Nurseitov, N., Paulson, M., Reynolds, R., & Izurieta, C. (2009). Comparison of JSON and XML Data Interchange Formats: A Case Study. Caine, 9, 157–162.

[35] Oancea, B., Rosca, I. G., Andrei, T., & Iacob, A. I. (2011). Evaluating Java performance for linear algebra numerical computations. Procedia Computer Science, 3, 474–478.

[36] Perron, B., & Stearns, A. (2010). A review of a presentation technology: Prezi.

[37] Pressman, R. S. (2005). *Software engineering: a practitioners approach*. McGraw Hill.

[38] Rabin, M. D., & Burns, M. J. (1996, April). Multimedia authoring tools. In *Conference Companion on Human Factors in Computing Systems* (pp. 380–381). ACM.

[39] Sangappa, S., Palaniappan, K., & Tollerton, R. (2002, November). Benchmarking Java against C/C++ for interactive scientific visualization. In *Proceedings of the 2002 joint ACM-ISCOPE conference on Java Grande* (pp. 236–236). ACM.

[40] Signer, B., & Norrie, M. C. (2007, February). PaperPoint: a paper-based presentation and interactive paper prototyping tool. In *Proceedings of the 1st international conference on Tangible and embedded interaction* (pp. 57–64). ACM.

[41] Shneiderman, B. (1981, May). Direct manipulation: A step beyond programming languages. In ACM SIGSOC Bulletin (Vol. 13, No. 2-3, p. 143). ACM.

[42] Tongco, M. D. C. (2007). Purposive sampling as a tool for informant selection.

[43] Tucker, W. D. (2015). Beyond traditional ethics when developing assistive technology for and with Deaf people in developing regions. In M. Hersh (Ed.), Ethical Engineering for International Development and Environmental Sustainability (pp. 293–324). Springer: London. doi:10.1007/978-1-4471-6618-4

[44] World Wide Web Consortium. (2006). Extensible markup language (XML) 1.1

Appendix A

Information sheet

1. What is this research project about?
    - I am going to tell you about a computer-based project, called Authoring tool
    - This project aims to improve communication between a Deaf and hearing person in any limited domain conversation scenario.
    - You might be familiar with the mobile application called SignSupport,
    - The mobile application that helps a Deaf person to communicate with a pharmacist during medical dispensing.
    - SignSupport only support android mobile platform, South African Sign language and a pharmacy scenario only.
    - The Authoring tool project aims to expand SignSupport in a way that it can accommodate multiple limited domain conversation scenarios, mobile platforms and Sign languages.
2. Who is running the project?
    - We are Computer Scientists from the University of the Western Cape.
    - You might know Bill Tucker. Hes the project leader.
    - The student responsible for this particular project is Lindokuhle Duma.
3. What do we want to achieve?
    - We want to improve SignSupport in a way that it:

- Accommodate multiple limited domain communication scenarios.
- Support multiple signed languages for illiterate Deaf users in developing regions
- Become platform independent to work on a variety of mobile devices.

4. What will we do?

   - We will design and build an application that runs on the computer.
   - The application will allow you to design mobile application for handling a limited domain scenario between a Deaf and a hearing person, e.g. when a Deaf person is visiting a pharmacist or the department of Home Affairs to get an ID book.
   - You will have to initially prepare a conversation script dialog aside then use the authoring tool to design the application.
   - The authoring tool will present you with screen templates and screen components like image buttons, text boxes, video frames etc.
   - The user will then be able to build the application interface using drag-n-drop features.
   - The video frames, in the application interface, will be used for displaying Sign language videos for conveying information to the Deaf person; and the text will be for conveying information to the hearing person.
   - You can also connect the screens by clicking on the button and then type the screen number to connect to, in the popup window.
   - The connection represents the conversation dialog and how you or other end users will interact with the application.
   - After you have completed designing the application interface, You will then the view the application design in a PDf document.

5. What do we expect of you?

   - We want Deaf people to help design and evaluate this system.
   - You will be asked to create simple conversation dialogs for different scenarios, using the system; and then view them in a PDF file.
   - Participation is of your own choice.
   - If you agree to participate, we will ask you to sign a consent form.

6. Benefits

- This study will improve SignSupport in a way that SignSupport will accommodate different limited domain communication scenarios, multiple signed languages and mobile platforms.
- Another benefit from this study empowers Deaf people and domain specialist to develop SignSupport scenarios together.

7. Risks AND difficulties

   - The risk is that the script dialog created from the tool might not cover all parts of the conversation within the limited domain scenario; and this might lead to a communication breakdown between a Deaf and a hearing person.
   - These script dialogs that will be produced by the tool will mainly be used for the purpose of this study and will not be used in the real world

8. Withdrawal and confidentiality

   - All videos recorded during the research session will be kept confidential and will be stored on a computer with password, which is known only by the responsible researcher.
   - Your identity will not be disclosed in public unless we receive permission from you.
   - Please be informed that you have the right to withdraw from any research session by informing the session leader.
   - As soon as you withdraw from the session, all material about your information will be destroyed.

9. Dissemination of the study results

   - All information will be disseminated when the study is completed in the form of several papers at various conferences.
   - Ultimately the rest of the results will be published in a form of a Masters Thesis.
   - Deaf participants will be kept informed via several presentations at DCCT at strategic times of the project life.

....

Signature Date

.....

Name and surname

Appendix B

Deaf participant consent form

I ..... fully understand the Authoring tool project and agree to participate. I understand that I can withdraw from the study at any time, and any information collected pertaining my contribution will be destroyed at once. I also understand that all information that I provide will be kept confidential, and that my identity will not be revealed in any publication resulting from the research unless I choose to give permission. I acknowledge that all in-formation attained in this study or test will be stored on a computer that has a password that is only known by the researcher. Furthermore, all recorded interview media and transcripts will be destroyed after the project is completed. I am also free to withdraw from the project at any time. I understand that an interpreter will be used for this trial and the information he/she translates will be kept confidential and not repeated.

For further information, please do not hesitate to contact: Lindokuhle Duma & Bill Tucker

Dept. of Computer Science
University of the Western Cape
Private Bag X17
Bellville 7535
Email: 3063344@myuwc.ac.za / wdtucker@gmail.com

.....

Signature (Participant)

Appendix C

Interpreter consent form

I ..... fully understand the Authoring tool project and agree to interpret. I understand South African Sign Language and will provide sign language translation. I am bound by Deaf South Africa's (DEAFSA) code of ethics for SASL interpreter to adhere all aspects of the Code of Ethics at all times during and after assignments; keep all assignment-related information strictly confidential and adhere to professional standards of confidentiality; and render the message faithfully, always conveying the content, intent and spirit of the speaker using the language most readily understood by the person(s)whom they serve.

I also pledge that I have explained all the aspects of the research to the participants.

For further information, please do not hesitate to contact: Lindokuhle Duma & Bill Tucker

Dept. of Computer Science
University of the Western Cape
Private Bag X17
Bellville 7535

Email: 3063344@myuwc.ac.za / wdtucker@gmail.com

Appendix D

Scenario graph to XML algorithm

The screens are made of InternalJFrames and the Canvas area is a JDesktopPane. They can be moved around, by default, by a mouse inside the canvas. The components of the screens are the components of the swing library (JLabel, JTextArea, JPanel) and are used to hold the content of the screens (text, icons and videos). The links between the screens are drawings from the AWT library. The method call repaint is executed every time a screen changes position, in order to redraw the link lines, to point to the new position of that screen. The screens are stored in an array as they are added into the canvas. This makes it easy to access them, as we use the array index to refer to them.

The video area is a uses JLabels to store videos when they are imported from the computer. VLCj is a library that is used to play the videos. This library is chosen because it is capable of playing many videos formats. This library also requires that the VLC is installed on the computer which the authoring tool is installed at. The videos are dragged and dropped to the screens as images. Xuggler library capture a video frame at a specific time point and stores it as an image file, with the same name as that video. That is the image file that gets dragged and dropped on behalf of the video.

After the scenario has been created, the XML file is then generated. The screens are accessed using an array index (screen [0] screen[n]), through a forloop. Each of these five screens has their own method, in a class file, which is meant to extract the screens content, as text, and copy it to the XML file, as shown in the figure above. Each screen has an integer variable called screentype, which has a value (1 - 5); and informs the method that should be called in order to extract the content of that screen. The text content like the title and the video captions are extracted by the

getText method. The videos and icons are represented by their file names in the XML file, so their file names get copied to a variable like videoName and iconName, during the drag-n-drop process. The possible changes of Screen type 1 4, are predictable so the content extraction methods of these screens are efficient to get all the content of these screens, regardless of the modifications made to them. Screen type 5 is unpredictable as it is unknown which swing component the author will use and what data type will be stored in those components. The algorithm used to extract content in screen type 5 is that all its components are stored in an array of type Component. This is the parent type of the Swing components so it allows the array to store different Swing components. Initially, the method needs to find the actual instance of the component in order to be informed on how to how to extract the content stored in that component. The content is then extracted from each component using the appropriate methods.

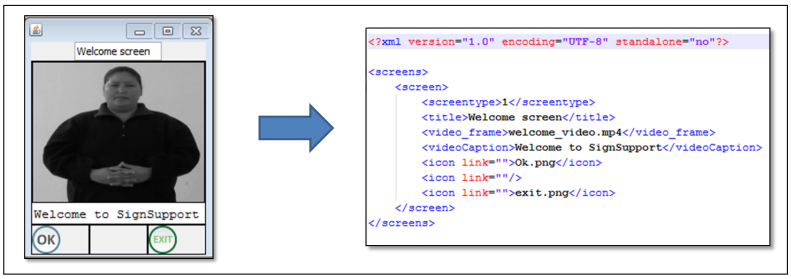

Figure D.1: Screen's output in the XML file

Appendix E

Turnitin results

Please note that the plagiarism percentage is 27% because some of the content in the thesis have been copied from a paper that the researcher wrote and published at South African Telecommunication and Network Applications Conference (SATNAC). The paper is cited in this thesis and is also attached in appendix F.

Turnitin Originality Report

Final Thesis V6 by Lindokhuhle Sifiso Duma

From Thesis (Msc 2015)

- Processed on 11-Jan-2016 07:24 GMT
- ID: 619640182
- Word Count: 22602

Similarity Index
27%
Similarity by Source

Internet Sources:
4%
Publications:
2%
Student Papers:
26%

sources:

1 13% match (student papers from 15-Jun-2015)
Class: SATNAC 2015
Assignment:
Paper ID: 550632177

2 10% match (student papers from 10-Aug-2015)
Class: MSc2014
Assignment:
Paper ID: 560522969

3 1% match (student papers from 13-Jul-2015)
Submitted to Kwame Nkrumah University of Science and Technology on 2015-07-13

4 < 1% match (Internet from 17-May-2014)
http://www.deafsa.co.za/code-of-ethnics/

5 < 1% match (Internet from 18-May-2009)
http://www.cs.ubc.ca/grads/resources/thesis/May04/Martin_Robillard.pdf

6 < 1% match (Internet from 25-Jun-2010)
http://www.tibb.co.za/Rep%20cupping.pdf

7 < 1% match (student papers from 17-May-2015)
Submitted to University of Mauritius on 2015-05-17

8 < 1% match (Internet from 13-Sep-2015)
http://cdn.entelectonline.co.za/wm-418498-cmsimages/ARJv102.v2.pdf

9 < 1% match (publications)
Xanthopoulos, Spyros, and Stelios Xinogalos. "A comparative analysis of cross-platform development approaches for mobile applications", Proceedings of the 6th Balkan Conference in Informatics on - BCI 13, 2013.

10 < 1% match (student papers from 28-Aug-2013)
Submitted to University of Warwick on 2013-08-28

11 < 1% match (Internet from 19-Aug-2008)
http://cehd.umn.edu/NCEO/OnlinePubs/Synthesis44.html

12 < 1% match ()
http://lrd.yahooapis.com/**http%3A//www.hta.ac.uk/fullmono/mon808.pdf

13 < 1% match (publications)
Ethical Engineering for International Development and Environmental Sustainability, 2015.

14 < 1% match (publications)
Aluko, Babatunde, Dafni Smonou, Michael Kampouridis, and Edward Tsang. "Combining different meta-heuristics to improve the predictability of a Financial Forecasting algorithm", 2014 IEEE Conference on Computational Intelligence for Financial Engineering & Economics (CIFEr), 2014.

15 < 1% match (student papers from 10-Feb-2015)
Submitted to University of Hertfordshire on 2015-02-10

16 < 1% match (Internet from 20-Aug-2010)
http://www.actionresearch.net/living/marianlothianphdopt.pdf

17 < 1% match (Internet from 14-May-2015)
http://eprints.soton.ac.uk/66322/1.hasCoversheetVersion/Sitthisak_O._Thesis_(2009).pdf

18 < 1% match (Internet from 25-Jan-2010)
http://cse.yeditepe.edu.tr/tnl/cbasaran.pdf

19 < 1% match (Internet from 06-Nov-2009)
http://www.cs.ubc.ca/grads/resources/thesis/Nov08/George_Sharath.pdf

20 < 1% match (student papers from 15-Mar-2013)
Submitted to University of Hertfordshire on 2013-03-15

21 < 1% match (Internet from 21-Sep-2013)
http://bpm.q-e.at/wp-content/uploads/2013/04/masterthesis_peter_zangerl.pdf

22 < 1% match (Internet from 13-Apr-2009)
http://ils.unc.edu/MSpapers/2866.pdf

23 < 1% match (Internet from 28-Dec-2014)
http://eprints.chi.ac.uk/806/1/571606.pdf

24 < 1% match (Internet from 29-Sep-2015)
http://eprints.soton.ac.uk/359840/1.hasCoversheetVersion/Laura%20Harris%20-

25 < 1% match (Internet from 25-Oct-2010)
http://www.cet.uct.ac.za/files/KnowledgeBase/VirtualMobiusCompleteAppendices.pdf

26 < 1% match (Internet from 03-Apr-2015)
http://hd.media.mit.edu/tech-reports/TR-559.pdf

27 < 1% match (Internet from 22-Aug-2015)
http://scholar.sun.ac.za/bitstream/handle/10019.1/79800/wiggetbarnard_disability_2013.pdf.txt?sequence=6

28 < 1% match (Internet from 09-Apr-2012)
http://www.doria.fi/bitstream/handle/10024/69761/isbn%209789522650986.pdf?sequence=1

29 < 1% match (Internet from 19-Jun-2014)
http://www.futuresteps.co.nz/PhD_University_Leadership_for_Sustainability.pdf

30 < 1% match (Internet from 05-Feb-2015)
http://soar.wichita.edu/bitstream/handle/10057/3734/t10105_Marrapu.pdf?sequence=1

31 < 1% match (publications)

Michael J. Burns. "Multimedia authoring tools", Conference companion on Human factors in computing systems common ground - CHI 96 CHI 96, 1996

32 < 1% match (Internet from 29-Nov-2015)
http://www.orc.soton.ac.uk/publications/theses/6460_gyc/6460gyc.pdf

33 < 1% match (Internet from 27-Jan-2015)
http://www.cimne.com/cm-master/doc/thesis/MasterThesisDibakarDatta.pdf

34 < 1% match (Internet from 13-Nov-2015)
http://www.ijhssnet.com/journals/Vol_5_No_4_April_2015/5.pdf

35 < 1% match (Internet from 18-Jun-2015)
http://www.orc.org.uk/publications/theses/3204_cch/3204_cch.pdf

36 < 1% match (Internet from 06-Mar-2014)
http://www.cs.uwc.ac.za/~mmotlhabi/Masters/SATNAC%202013%20Michael%20Motlhabi.pdf

37 < 1% match (Internet from 13-Dec-2015)
http://clok.uclan.ac.uk/8562/1/Cowan%20Hillary%20Final%20e-Thesis%20%28Master%20Copy%29.pdf

38 < 1% match (Internet from 12-Oct-2011)
http://matrix.memf.uwindsor.ca/publications/theses/DAlimonte-MScThesis-100324.pdf

39 < 1% match (Internet from 03-Jun-2010)
http://www.feweb.vu.nl/gis/research/LUMOS/publications/docs/Thesis_Rohan_final.pdf

40 < 1% match (Internet from 13-Aug-2014)
http://theses.gla.ac.uk/590/1/2009Galvezcruzphd.pdf

41 < 1% match (Internet from 09-Oct-2009)
http://warren.info/dr/?q=node/13

42 < 1% match (Internet from 01-Oct-2015)
http://www.dr.library.brocku.ca/bitstream/handle/10464/5616/Brock_Gleeson_Laura_2014.pdf?sequence=1&isAllowed=y

43 < 1% match (Internet from 11-Oct-2015)
http://www.thesis.bilkent.edu.tr/0006352.pdf

44 < 1% match (Internet from 24-Oct-2015)
http://etd.uovs.ac.za/ETD-db/theses/available/etd-11112011-093413/unrestricted/KhomokhoanaPJ.pdf

45 < 1% match (publications)
The Teaching and Learning of Statistics, 2016.

46 < 1% match (publications)
"Migration of Web Applications with Seamless Execution", Proceedings of the 11th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments - VEE 15, 2015.

47 < 1% match (publications)
Lecture Notes in Computer Science, 2014.

48 < 1% match (publications)
Yu, Chia-Jen, and Jian Kang. "Acoustic Sustainability in Urban Residential Areas", Procedia Environmental Sciences, 2011.

49 < 1% match (publications)
Vankan, Wilhelmus, and Martin Laban. "A SPINEware Based Computational Design Engine for Integrated Multi-Disciplinary Aircraft Design", 9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, 2002.

50 < 1% match (publications)
Lecture Notes in Business Information Processing, 2015.

51 < 1% match (publications)
Lecture Notes in Computer Science, 2015.

52 < 1% match (publications)
Heimerl, Kurtis, Janani Vasudev, Kelly G. Buchanan, Tapan Parikh, and Eric Brewer. "Metamouse : improving multi-user sharing of existing educational applications", Proceedings of the 4th ACM/IEEE International Conference on Information and Communication Technologies and Development - ICTD 10 ICTD 10, 2010.

paper text:

An authoring tool for generalised scenario creation for SignSupport By Lindokuhle Sifiso Duma

19**A thesis submitted in fulfilment of the degree of Master of** Computer **Science**

January 2016

33**Declaration of Authorship I,** Lindokuhle S. Duma, **declare that this thesis titled,**

13**Usability and content verification of a mobile tool to help a Deaf person with pharmaceutical instruction,**

3**and the work presented in it** is **my own. I confirm that: • This work was done wholly while in candidature for a research degree at this University. • Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated. • Where I have consulted the published work of others, this is always clearly** attributed. • **Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work. • I have acknowledged all main sources of help. • Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself. Signed: ———————————— Date:———————————** iii

Dedication Dedicated **to** Sindisiwe Kubheka, **my**

fiancee, for always being the pillar of my strength. v Abstract This thesis describes the development cycles

1**of an authoring tool** that generalises **scenario creation for SignSupport.**

1**SignSupport is a mobile communication tool for Deaf people that currently runs on** an **Android** smartphone. **The authoring tool is computer-based software that helps a domain expert, with little or no programming skills, design and populate a limited domain conversation scenario between a Deaf person and a hearing person, e.g., when a Deaf patient collects medication at**

a hospital pharmacy or when a Deaf learner is taking a computer literacy course. SignSupport provides instructions to the Deaf person in signed language videos on a mobile device. The authoring tool enables the creation and population of such scenarios on a computer for subsequent 'playback' on a mobile device. The output of this authoring tool is an XML script, alongside a repository of media files that can be used to render the SignSupport mobile app on any platform. Our concern was to iteratively develop the user interface for the authoring tool, focusing on the domain experts who create the overall flow and content for a given scenario. We

had four development iterations, where the first three were evaluated for usability;

1for both pharmacy and ICDL course scenarios with purposive sampling. The

fourth iteration focused on using the authoring tool to design an ICDL practise mobile app, recording the necessary SASL videos and using an XML parser to render the designs XML script into an Android app. The research conducted herein leveraged multiple approaches to content authoring and generalisation; and further that software generalisation can improve accessibility and affordability for the ultimate end users. The thesis concludes with a summary of recommendations and lessons learnt. Acknowledgements

21I would like to thank all my lab mates for their continued support and

for keeping me on my toes on

35my research. I would like to thank Prang for

always ensuring that I'm prepared when I'm collecting data or presenting my work; my Supervisor, Bill Tucker, always making tasks look easier and for always being there when I need him. I would also

1thank the Deaf Community of Cape Town for their involvement,

includ- ing Meryl Glaser.

1Thanks also to George Ngethe, Marshalan Reddy, and Edwin Blake at UCT for their collaboration on this project. We also thank Telkom, Cisco, and the THRIP (Technology and Human Resources for Industry Partnership) initiative of the South African Department of Trade and Industry for financial support via the Telkom Centre of Excellence (CoE) programme. THRIP funding (project TP13072623839) is managed by the National Research Foundation (NRF). Any opinion, findings and conclusions or recommendations expressed in this material are those of the authors and therefore the NRF/THRIP does not accept any liability in regard thereto.

Lastly,

17I would like to thank my parents and siblings for always being there for me.

human participants . . . . . 5.3.2 Generalisation activities . . . . . 5.4 Recommendations and future work . . . . . 5.4.1 Authoring tool features . . . . . 5.4.2 Conducting research for Deaf people . . . . . Bibliography

20**Appendix A Information sheet Appendix B** Deaf participant **consent form Appendix C** Interpreter consent **form Appendix**

D Scenario graph to XML file 53 53 54 55 55 56 xiii List of Figures 1.1 Scenario design process . . . . . 2.1 A screenshot of XML structure . . . . . 3.1 Four iterations of the authoring tool development . . . . . 4.1 A screen shot of the first authoring tool prototype . . . . . 4.2 Steps of the training stage. . . . . 4.3 A screenshot of the second Authoring tool interface . . . . . 4.4 A screenshot of the new authoring tool interface . . . . . 4.6 Signed language video editing . . . . . D.1 Screen's output in the XML file . . . . . 28 29 33 39 45 70 xv List of Tables 4.1 Recommended features and their motivation . . . . . 4.2 Participants' definitions of the authoring tool . . . . . 4.3 Authoring tool related to cars. . . . . xvii Chapter 1 Intro duction

1**Deaf with a capital D refers to a linguistic and cultural group of people with/** without **hearing loss who mainly use SASL as their mother tongue** [8]. **Deaf people often experience** commu- nication **barriers while communicating with a hearing person who cannot sign** [11, 12, 13]. **While most accessible technologies support voice and text communication, this presents** us- ability **difficulties for Deaf people with low functional text literacy. SASL interpreters are very expensive for Deaf people as they are very scarce, so Deaf people battle significantly to communicate with hearing people in the absence of an interpreter.**

The current versions of SignSupport are two Android mobile applications; one for medi- cal dispensing and another for computer literacy training scenario. The medical dispensing application

1**helps a pharmacist to give comprehensible medical instructions to a Deaf** pa- tient **during medicine dispensing in the form of pre-recorded South African Sign Language (SASL) videos** [28]. **These medical instructions are stored on a mobile** phones **memory card for the patient to view later on.**

Computer literacy training app helps a Deaf computer skills learner to work through the units of this course at their own pace. The unit instructions are also in SASL videos and

1**are stored on a mobile** phones **memory card for the** Deaf learner **to view** anytime. **The limitations of** these **SignSupport** apps **are that** they **only**

have SASL instructions, run on an Android mobile platform and each app caters for only one scenario.

1**Therefore, this authoring tool is aimed to generalise SignSupport to accommodate** mul- tiple **domain scenarios, for multiple mobile platforms and that can be populated by any language for low literacy end users.**

The

1**authoring tool for SignSupport generalisation** de- 1 scribed **in the** thesis **is a computer-based application that helps a domain expert with** basic **or no programming skills to design a front-end interface for a communication flow between a Deaf and a hearing person; in any limited domain scenario, e.g.,**

during medical dispensing from a pharmacist to a Deaf patient; or a self-learning process for computer literacy skills by a Deaf learner [5]. The

initial design of the authoring tool came in [5]. 1.1 Aims and ob jectives This section presents the aims and objectives of the research. SignSupport currently is an android app that only covers the pharmacy and Deaf patient scenario and in South African Sign Language only. This is limiting because it means that all Deaf people in South Africa should buy android phones to have access to this app and this also means that SignSupport only helps the Deaf people when they collect medicine at the pharmacy and nowhere else. Here are the three problems that the research aims to tackle. 1.1.1 Scenario independence Deaf people experience communication challenges on a daily basis, in different scenarios

2when they have to communicate with a hearing person who cannot sign. Hence, there is a

need for technologies that empowers Deaf people to communicate effectively in many scenarios, when interpreters are not available. Here, we aim to generalise SignSupport to enable Deaf people to communicate with hearing users in multiple limited domain scenarios. 1.1.2 Language independence SASL has different dialects and signed languages can be significantly unique from country to country; just as for spoken languages. SignSupport needs to be generalised so that it can support different signed and spoken languages. 1.1.3 Mobile platform Independence The are many different mobile platforms available today. It is not easy to predict which mo- bile platform is possessed by most Deaf people. Hence, SignSupport needs to accommodate 2 a wide variety of mobile platforms. 1.1.4 Anticipated authoring design The authoring tool presents the domain expert with screen templates, icons and other fea- tures to create a scenario.

1These screen templates can be populated with text, icons and/or signed language videos (that would be provided by native signed language speakers, not the domain experts,

unless they are Deaf, of course). The screen templates explained here are generic screen layouts that can be used as a starting point when creating a scenario 1. These

1screens can also be linked to each other in a graph structure. The link between each screen indicates the user interaction on the real mobile app i.e. which screen will show next when the end user clicks a given button.

The

1authoring tool can then produce an Extensible Markup Language (XML) script as an output after the flow is designed by the domain expert. This XML file is consumed by a XML parser to render a mobile application on any given mobile platform.

Figure 1.1 shows the flow of creating a scenario using the authoring tool and rendering a mobile app using the XML parser. 1

38.2 Background This section describes the scenarios of SignSupport prior to the

authoring tool. This section presents the design and implementation for each scenario. 1.2.1 Doctor and Deaf patient scenario This is

1an Internet browser-based mock-up design [30, 26]. The main objective of this mock- up was to help the doctor understand the symptoms of a Deaf

patient so that s/he could prescribe medication for the patient. The mock-up presents a set of questions in SASL which the Deaf patient answers before seeing the doctor. The doctor then views the summary of the patients **answers in English text.**

1 http://techterms.com/definition/template 3 Figure 1.1: Scenario design process Starting from an authoring tool and ending in a mobile app. The domain expert will design the scenario using the authoring tool, the authoring tool will produce an XML script as an output. The XML parser uses the XML script to render a mobile app for different mobile platforms

1**Version 2 was on the same scenario as version one and was developed as a Symbian mobile app with XHTML** [30]. **This mock-up required only a mobile phone with a data connection running a browser that supports Small web format. However, the doctor and Deaf patient scenario was found to be too wide and complicated to be constructed as an application**

[9]. 1.2.2 Pharmacist and Deaf patient scenario

1**Version 3 then shifted into a more constrained communication domain between a** pharma- cist **and a Deaf patient** [9]. **Its aim was to help the Deaf patient understand the medical instructions as prescribed by the pharmacists. This mock-up was designed to have 2 types of interfaces: one for the Deaf patient to input their background information prompted by SASL videos and icons and one for the pharmacist to view the Deaf** patients **background information and dispense medication** [9, 28, 29]. **Version 4 was a redesign by a multi-disciplinary and trans-university team and then implemented as an Android app by** [28, 29]. **The construction of this version is in the process of usability testing in a public hospital pharmacy.**

4 1.2.3 Computer literacy training scenario SignSupport mobile application for computer literacy training scenario, helps a Deaf com- puter skills learner to work through the units of this course at their own pace. The challenge within this course is that some learners understands the content of the course faster than others so the signing educator needs to repeat the lessons until all learners understand. This becomes a burden for the faster learners because they need to wait for the slower learners to understand before the teacher introduces new content. Another challenge is that the educator needs to travel around the world quite often. This puts the course on pause until she comes back. SignSupport for ICDL helps faster learners to move on to the following units of the course while the educator explains the current units to the learners who are struggling [5, 32]. This app also enables the course to go on while the educator is away. The learners just have to write down all the parts that they don't understand so that the educator can explain when she returns. The authoring tool that designs the ICDL app only designs the front-end of the ICDL app. This authoring tool produces XML script as an output. This XML script represents the content of the app (in plain text) that was designed by the authoring tool. The differ- ence between this authoring tool and the one discribed in this thesis is that this authoring tool can only be used to design ICDL course units whereas the authoring tool discribed in this thesis can be used to design ICDL course units and many other limited domain scenarios. There is another researcher who is working on an XML parser which is a back-end of the authoring tool. The XML parser uses the XML script to render a platform dependent mobile. This XML parser currently renders Android apps[5]. 1.3 Role of author

1**Our research team sees the potential of SignSupport for additional contexts and limited domain communication scenarios.**

This can be witnessed by the research listed in the back- ground section above. Newer research streams, under SignSupport, includes health edu- cation for Deaf people, which focuses on diabetes and hypertension; testing the medical 5 dispensing app at a real pharmacy and an XML parser which uses XML to render a mo- bile app. The role of the author is to create a generalisation authoring tool that aims to generalise SignSupport for multiple platforms, signed languages and limited domain scenar- ios. This

authoring tool would assist domain experts in designing communication flows that meet the specific needs of their Deaf clients. The focus of the research is mostly on multiple scenarios. This is because the current research topics and prototypes under SignSupport are mostly scenario dependent. The researcher works closely with three researchers: the industrial design engineer who designed the medical dispensing app; the computer scientist who created the authoring tool to designs a computer literacy app and then produces XML output; another computer scientist XML parser researcher who created an XML parser that uses the XML to render a mobile app. The industrial design engineer helps to evaluate the authoring tool interface in a pilot session. The two computer scientists helps to design the XML structure that the generalisation authoring tool should produce. This helps to ensure that the XML produces by the generalisation authoring tool meets the expectations of the XML parser. The source code of the generalisation authoring tool can be found in the this link: http://cs.uwc.ac.za/ lduma/masters/authoring tool.rar 1.4 Thesis outline

> 32**This thesis is divided into** 5 **Chapters, chapter** 1 **introduces the**

research problem and the proposed solutions. Chapter 2 describes the related work that can be referred to when building up a solution for the research problem. Chapter 3 states the methods that are applied to carry out the research; building and testing the authoring tool. Chapter 4

> 45**presents and discusses the results** obtained in **three** iterations **of the**

authoring tool development; Chapter 5 concludes the research and presents a direction for future research. The computer science researcher 6

> 30**Chapter 2 Related work This chapter presents the related work**

that is consulted to get ideas on how to create the authoring tool. Section 2.1 describes the authoring tools, the different types and commonly used authoring tools. Section 2.2 presents the cross-platform approaches in app development and programming languages. Section 2.3 presents universal design approaches. Approaches presented here include universal design in learning, housing and WYSIWYG editors. 2

> 2**.1 Authoring tools Authoring tools are** software **that allow authors (domain experts), with** basic **or no programming skills, to build and populate complex applications with data or hypermedia files for their domain scenarios e.g. E-learning web apps** [14]. **Authors can create these complex and attractive** applica- tions **by merely clicking and defining relationships between objects, e.g. forms, text, pictures and videos. Authoring tools** provide **authors, with programming experience,** direct access **to**

> 31**common programming constructs such as functions,** loops, conditions **and** variables **that are** useful **to**

pro- vide end-user interactivity in the application being created [37]. The

> 2**code inserted by the author is automatically compiled by the authoring tool and reflects in the Graphical User Interface (GUI) pane. The author has an option to preview the scenario design to see how it will look when it is a complete app. S/he can then modify the design or its code until satisfied.**

Authoring tools can be classified into three categories such as card-based, icon-based and time- based1. Card-based authoring tools allow the author to organise elements as if they were a stack 1http://www-it.fmi.uni-sofia.bg/courses/maten/module3/classification.html 7 of cards. These elements can be viewed individually like cards or pages in a book [39]. An example of such authoring tools are the web or

presentation authoring tools, which allow you to view one web page or presentation slide at a time. Icon based authoring tools enable the authors to create a flow or manipulate objects directly to complete a task[2, 20]. Time-based authoring tools arrange an element in time frames e.g. a slide show that plays different music instruments in different time frames. These are commonly known as multimedia authoring tools [2].

> 2**Advantages of authoring tools are that they save** much **time because they have templates. Templates help authors to work faster as they are generic and can be adapted to any form within the intended use of that authoring tool. Therefore, the author can** customise **and populate with content in a timely** manner 2. **Another advantage of authoring tools is that they are easy to use since authors do not have to possess programming skills in order to effectively use the authoring tool. The major drawback of authoring tools is that they have limited options and this might affect the author's creativity**

3. 2.1.1 Types of authoring tools 2.1.1.1 Web authoring tools Creating a website is traditionally done by typing HTML code on a notepad. The demand for more interactive and multi-purpose website drives the developers towards seeking for more automated website generating solutions. This led to the invention of web authoring tools. Web authoring tools generally present the developers with web page templates that the user can add and modify to suit his/her needs. These page templates carry the designs that are common in many websites. For example, a page template could have a banner and navigation buttons that a developer can modify and add links to other web pages or websites; and a footer at the bottom that has links that s/he wants to see on every page. A link manager also helps to verify that the links are done correctly. Other web authoring tools also have a feature to mark the progress of the page e.g. version 1, 2, 3 etc. The developer can also preview the website internally on the authoring tool or externally using a web browser. 2http

> 10**://smallbusiness.chron.com/advantages-disadvantages-using-webauthoring-application-27288.html**

3http

> 10**://smallbusiness.chron.com/advantages-disadvantages-using-webauthoring-application-27288.html**

8 2.1.1.2 Multimedia authoring tools Media content includes text, images, videos and animations. Multimedia authoring tools allows the user to integrate multi media content into one application. These tools provide users, with programming skills, with common programming constructs (looping and conditional branching) that are useful in providing interactive multimedia applications to end-users. Multimedia author- ing tools have built-in tools for the user to create/edit different media elements, however these built-in tools are not as comprehensive and flexible as the specialised software. Multimedia projects are usually managed by a multi disciplinary team whose members include project manager, programmers, producers, graphic artists, video and audio experts. Some multi- media authoring tools enable collaboration such that team members can work on different tasks at the same time. 2.1.2 Examples of authoring tools 2.1.2.1 Prezi presentation tool Prezi is an online tool that allows an author(s) to create and publish presentations. The author creates a presentation by adding elements (text, pictures, animations and videos) in a canvas [35]. The elements can also be grouped together in frames. Frames and individual elements are linked together to control the flow of the presentation. The elements can be arranged and edited using various tools within Prezi. Zoom in/out is used to add supplementary details and subtopics to the presentation This tool offers a number of advantages as opposed to traditional presentation tools like Power- Point [35]. It allows the author to work on any computer that has a high speed Internet connection. Authors in different locations can work collaboratively in one presentation [25]. The content of the presentation is not limited by the size of the slides since this tool uses a canvas, which has no limits. Disadvantages of Prezi is that one needs to be connected to the Internet in order to work on this tool. This tool is not free and some of its features might not be affordable to many. Both Prezi and PowerPoint support non-programmers by enabling their users to work only on the graphical user interface. Prezi presents the user with tools and elements that can be added on a canvas; and PowerPoint presents the user with menus and elements to add on the editable slide 9 templates. Users can also do a trial and error when manipulating objects on the canvas or slides, as these authoring tools enables users to easily

recovers from errors. Both these authoring tools allow users to add their own media files to the presentation. 2.1.2.2 Dreamweaver Dreamweaver is an offline based authoring tool that enables authors to create and manage web- sites. The four main features that make this tool powerful are

> 52**point-and-click, drag-and-drop,** HTML inspector **and**

RoundTrip HTML.

> 49**Point-and-click; and drag-and-drop** enables authors **to**

visually create websites without having knowledge about HTML or other scripting languages. This in turn makes the web authors work faster since they no longer have to type the HTML source code by hand. HTML inspector enables web authors to have access to and control the HTML source code generated by Dreamweaver. They can then edit this source code willingly. RoundTrip reads and validates the syntax of the HTML source code that is imported from other HTML editors. Advantages of Dreamweaver are that the author can work faster by visually designing and developing web sites, and authors can organise their site content in directories from their desktop, in the same way that the web site will be stored in the server. This eliminates the need for a web server when the web site is still under construction. Dreamweaver can also push or pull web content from the server using FTP connections. 2.2

> 1**Cross-platform development Cross platform means developing one application that can be installed on many platforms without having to modify its original source code [8].** Every **platform**

is unique, e.g. android apps are created mainly with Java and iOS apps are built using objective-C. So App developers, who wish to reach out to many target app users, would have to build the same app separately for all mobile platforms. This is time consuming and might lead to app inefficiency. Cross-platform techniques aim to enable developers to build a single app that can be executed on different platforms without modifying its underlying source code.

> 48**A number of techniques have been developed to**

achieve cross-platform development: web and hybrid apps, frameworks like PhoneGap and Titenium 10 2.2.1 Apps Approaches to achieve cross-platform development include building web and hybrid apps. Both these

> 1**apps are built using the latest technologies such as HTML5, CSS3 and JavaScript**

but they differ in the way that they are executed [18]. Web apps are not physically installed on the device but run on the device's browser via URLs. These apps store user data in the server or in browser-based cookies. The advantage of web apps is that they don't need physical installations, so any device with a web browser and Internet con- nection e.g. Facebook mobile. The drawbacks of web apps however, is that they are inaccessible to a device without an Internet connection. Some mobile web-browsers changes the look-n-feel of some web-apps or even deactivate other web-app's features, which can affect user experience. These apps also have

> 9**limited access to the device's hardware**

features [22, 18]. Hybrid apps, on the other hand,

> 9**try to combine the advantages of** both **web and native apps.** These **apps are**

> 9**installed on the device and** use special APIs **to** gain access to **the device'** s

hardware

features. Common API used in hybrid apps, to achieve the platform look-n-feel is called jQuery languages.

7**2.2.2 Programming languages 2.2.2.1 Java Java is a programming language**

that supports multi-threading, automatic memory management and object orientated programming [10, 38]. Java source code is not directly compiled into a native language, but into bytecode. Bytecode is an architectural independent set of instructions and it enables ease of application migration as the computer systems evolves [10, 21]. Bytecode is created by the Java virtual machine (JVM). JVM can execute bytecode instructions one at a time, through just-in-time compilation, or bytecode instructions can be compiled into native instructions and then executed directly. Compiling one instruction at a time helps to verify untrustworthy source code. This is not desirable for frequently used instructions, as they will have to be compiled into native instructions every time they are needed. Compiling bytecode instructions into native instructions, before execution, is known as ahead-of-time compilation. This is also common in c/c++ programs [10]. Ahead-of-time compilation sacrifices the application portability and verification of source code. 11 Java performance lags behind compared with other programming languages like C, due to a number of issues [34]: Arrays in Java can only be one or two dimensional, there is not support for multidimensional arrays. Java also does not support complex arithmetic and operator overloading. 2.2.2.2 HTML5 HTML5 is a scripting programming language that has been adapted in the web apps due to its advanced features [22]. HTML5 app runs on a devices web browser. This means that there is no need for the target user to manually install and update such apps. HTML5 app can store data offline and it can fully function with no Internet connection. Some of the common features of HTML5 are hardware integration which gives the web access to the hardware features of the device; supports different multimedia formats and user interaction (touch, speech and vibration) [22, 18]. 2.2.3 Markup languages 2.2.3

2**.1 Extensible Markup Language Extensible Markup Language (XMl) is a scripting language that is made up of tags that define data** [43]. **Unlike HTML, the tags of XML are created by the developer when creating XML documents. These tags are used to describe data within the XML document(s) for ease of storage and retrieval** [6]. **Each document has a root tag which all other tags belongs to. For example, an XML document that has information about books might have a root entity that gives** information **about where those books belong i.e. library. The following tags might be the faculties in which the books are made; then the other tags could be carrying the book name and its** author **number. This XML document can be used to search books for a certain faculty or a book** written by **a specific**

author. Figure 2.1 represents an example of such an XML document. XML has a number of advantages, which are [6, 31]: •

2**XML makes is easy to send data between any incompatible applications or devices. • XML entities can be used to** optimise **information retrieval in large databases. • XML is stored as plain text so it is easy to upgrade systems without losing data.**

12 Figure 2.1: A screenshot of XML structure that stores the information of the books in the library. This is an example of how XML can be used to represent scenario specific

2**data. XML is also accompanied with a number of technologies that assist in processing data in XML documents. These technologies include Extensible Stylesheet Language Transformation (XSLT), Xpath and Xlink** [6, 31]. **XSLT is used to transform XML documents to other browser-based** lan- guages **like XHTML. XSLT can also edit XML documents like adding and removing** elements **and attributes to and from the XML file. Xpath is also used in XML**

transformation but its main goal is to navigate through the elements and attributes **of an XML document. XLink is responsible for defining links to other resources** [6]. **It does this by allowing attributes to** be **inserted** into **XML documents to describe both internal and external linked resources**

˙ 2.2.3.2 JSON JSON is a data exchange language that has been designed to be

15**human readable and easy for computers to parse** [33]. **JSON is** ideal for **Javascript and** it **is** well known **for** data exchange between **JavaScript**

and web pages. JavaScript has an eval() function which directly parses JSON into an object [33]. This increases the performance of JSON in a way that it becomes faster to parse than XML, because XML required standalone libraries like DOM. JSON can represent both strings and arrays (non-ordered and ordered list) [33]. Here are some of the advantages of JSON [33]: 13 • It does not use the concept of tag. • It can be parsed faster than XML. • It needs less data size to describe the same data Some of the drawbacks of JSON over XML are [33]: • Lack of input validation • Lack of support for data with mixed content • Lack of namespace support 2

2**.3 Universal design Universal design entails creating software that can be used by all people** effectively, **without the need for adaptation or** specialised **resources** [27]. **Principles of this method include**

flexible to

11**use, simple and intuitive** to **use, perceptible information, tolerance for error, size and space for approach and use**

[27, 7]. Some of the common approaches of universal design are described below. 2.3.1 Learning Universal design in learning seeks to address the needs of all learners, even those with learning disabilities [23]. A learning process can be made flexible by having a virtual group where the learners can post questions or have discussions on a certain subject. The teacher can also be a member in that group engag with the learners or have individual sessions with the learners. This is very helpful especially for those learners that

44**are shy to ask questions in class. The**

textbook used by learners can be assessed for the way their content is presented and organised so that their level of difficulty can be determined. Learning software should also give positive feedback to the learners when they make mistakes. This can happen by either the software indicating to the learner to try again or the software can provide hints on how to complete a certain task [23]. 2.3.2 Housing Universal design in housing has two concepts: visitability and accessibility [15]. Visitability refers to applying universal design in the house in order to accommodate an elderly or disabled person visiting that house. Accessibility here refers to applying universal design in the house to accom- modate an elderly or disable person living in that house. Universal design applies to both exterior 14 and interior aspects of the house. Exterior aspects includes accessible paths with no steps, that connects to other facilities like sidewalks, toilets etc. Automatic doors are also desirable for people who might have difficulty opening doors by themselves. Interior aspects includes having doors that open wide enough to accommodate a wheelchair, Bathrooms should have appropriate space and have grabs by the wall, near the toilet seats for physically challenged people for balance. Kitchens should accommodate multiple users. Its furniture can be movable and be in a reachable height to accommodate a person with a wheelchair. The floor could have a carpet to accommodate a person who uses crutches[15]. 2.3.3 WYSIWYG editors The

2**acronym "WYSIWYG" means What You See Is What You Get, which refers to a system or an application that has a graphic interface and enables the user to view the end results of the product while s/he is still creating it** [40] . **An example of WYSIWYG editor is Microsoft Power Point application. Power point**

presentation r application allows you to see the overview of the slides presentation while you are still creating it. The principles of WYSIWYG editors state that [40] • Physical representation of actions instead of complex language specific commands • Reversible operations whose results can reflect on the object immediately. • Layered approach to learning that enables novice users to learn useful commands until they become experts Thus the WYSIWYG editors are visual and allow the user to directly apply changes to the product and the results are visible immediately on that product. The formatting commands of these editors are physical actions such as mouse clicks, button presses; as opposed to line command editors that require the user to recall and type all the formatting commands and their syntax; in the command line. The labeled menus these editors invoke commands and also inform the user about the available features within the editor [40]. This frees the user from having to memorise all the complex commands in order to use the editor effectively. Most line editors requires a display command in order for the user to confirm the modifications that s/he made on the product but with the WYSIWYG editor, changes made reflect immediately. The benefits of WYSIWYG editors are that they could have an "undo" option, which invokes an inverses operation of the previous action(s). This would then reverse the changes that were made 15 previously. Error messages would rarely be needed in this case because users can immediately see the results of the action they just made; and if they dont like it then they press the undo button. Users would also experience less nervousness as all actions are easily reversible [40]. Experts can work faster and are able to complete many different tasks. Such editors also have built in tutorials or wizards that help novice users to learn about the editor in a step-by-step process. This also gives users confidence when using the editor because they can initiate actions and predict their outcome. WYSIWYG editors also have a few issues. The graphical representation of their features might not be that obvious to the user and hence users might take more time to learn all the features of the editor [40]. Most editors that target international users might have a misleading graphical representation. This might lead to user frustration if they make wrong conclusions about the features. Some editors need bigger monitors to display the progress of their intended product and all its features. So users with smaller computer screens might have issues as the editor's interface might appear too small. 2.4 Summary The

2structure and the features of the existing authoring tools

are going to be

2used as a reference to inform the interface design of the SignSupport authoring tool. SignSupport generalisation author- ing will be a computer-based front-end that allows the domain specialist to construct dialogues. The general goal of the research is to investigate the needs of users who are highly skilled but not computer specialists.

Universal design emphasises the need to accommodate all possible users when building a product. The

2authoring tool will have tooltips; help options and a user guide for novice

users. This enables any author to learn to use the authoring tool quickly and also to have assistance within the authoring tool in case s/he needs it. These helper options can also be adapted into signed language videos, as tutorials that demonstrate certain functions for Deaf authors.

2SignSupport generalisation authoring tools will produce an XML document that has all the details of the scenario that the domain expert has designed. XML

is chosen over JSON because it has no predefined tags or restricted names. This means that any name can be used as a tag name or namespace. XML also has technologies like XPath, Xlink, DOM, SAX etc. which help to validate the XML structure and flow. So XML has more tools and support than JSON, its competitor. The authoring tool users will be working on the authoring tool's GUI. This enables them to manipulate objects directly and also to see the results as they progresses with their tasks. This is learned from WYSIWYG editors. The next chapter presents and discusses the methods that will be used to guide the research processes and authoring tool development. 16 Chapter 3 Metho dology This chapter discusses the methods that are used to design and implement the authoring tool. Section 3.1 introduces the research questions. Section 3.2 explains the methods that are used to answer our research questions. This includes the method that guides the research process and the implementation of the authoring tool. Section 3.3 describes how we are going to evaluate the authoring tool for usability and scenario creation. Section 3.4 explains the ethics that we need to follow as we are evaluating the authoring tool with human participants.

503.1 Research questions The research question for the

project is as follows: How can we generalise SignSupport for any given limited domain conversation scenario and also accommodate different signed languages, and multiple mobile platforms? This question is further broken into three sub-questions: 1. How can we make SignSupport platform independent? 2. How can we

2make SignSupport scenario independent? 3. How can we

make SignSupport language independent? The first sub-question investigates techniques that contribute to platform

2independence. In order to reach many Deaf people, SignSupport needs to be ported to many mobile platforms like Blackberry, iOS, Windows and more

[18]. One cannot predict the kinds of mobile phones that Deaf people prefer. Although, research can be conducted to find out about the mobile phones that are widely used by Deaf people. This would narrow the research focus towards widely used mobile phones; and filter out the rest. The goal of this research is to accommodate all mobile phones 17 so that all Deaf people have access to the knowledge and information that they need, using their mobile phones. The only criteria is that the mobile phone should a high resolution screen to play high quality signed language videos. The authoring tool needs to have features that enable the domain expert to design a scenario and then port it into a platform dependent mobile app. To

2achieve this, we aim to build an au- thoring tool that a domain expert can use to design

a limited domain scenario

2and then produce an XML script, as an output, to be parsed into

a mobile app of

2**different platforms by an XML parser. XML script** is **a text file that contains information**

of the designed scenario flow

2**and all its data assets (text, icons and videos) that** go **with each and every screen in the**

scenario. An XML parser consumes and parses the

2**authoring tool's XML output in order to render platform-specific mobile apps e.g. Android, iOS or Windows Mobile.**

The second sub-question investigates

2**ways in which we can expand SignSupport to** accom- modate **multiple limited domain**

scenarios.

2**SignSupport currently covers the medical dispensing scenario between the Deaf patient and the pharmacist only**

[28]. Many

2**other scenarios** exist **where Deaf people experience difficulties when they need to communicate with hearing people who** can- not **sign.**

SignSupport needs to accommodate a

2**wide range of limited domain scenarios in order to effectively assist Deaf people in the absence of** a **SASL interpreter. The authoring tool** needs to **provide the domain experts with different templates that they can** customise **and populate with scenario specific data** (icons, text **and** videos) and **then link these templates to define a conversation flow, in** the **form of a decision tree. Any domain expert will be able to use/re-use the provided templates or create new templates for** a **limited domain specific conversation flow. The**

third sub-question investigates ways for SignSupport to accommodate multiple languages, both signed and spoken. SignSupport currently operates in SASL pre-recorded videos and English text. There could be a time where the same limited domain scenario needs to be created for other languages, e.g. American Sign Language and Spanish. This simply means that the flow of the scenario remains the same but the scenario data assets changes (signed language videos, icons and text). The authoring tool needs to enable the author to create a limited domain scenario design, that can be changed and populated with different data assets and then update the XML script. 18 3.2 Methods

37**This section describes the methods** that guides **the** research process **and the**

steps of the prototype development and testing. 3

2.2.1 Action research Action research (AR) is a methodology that is cyclic in nature and consists of spiral iteration steps that involves circles of planning, action and reflection

[17]. This method encourages the researcher to conduct

2research with participants who are experiencing a problem under study, in their everyday lives; as opposed to reading articles about them and then creating case studies based on the literature.

2Researchers work collaboratively with the participants in a cycle of activities to try out a theory in real situations, gain feedback and reflect from it and try again in the next cycle

[1]. The

2main focus is on the participants because the human aspect of each solution needs to be taken into consideration because people have different and conflicting views and attitudes; people also change over time.

AR contributes to scientific knowledge and sustainable social change. AR can also be summarised into these points [16]: • AR encourages

2collaborative action from both researchers and participants. • The researcher needs to understand the values of the participants as they

have an impact on the results

2at the end of the research. • Long term collaborative work with the participants,

other than surveys or single interviews,

2helps the researcher to better understand the participants and gain deep in-sight information that would have been impossible to obtain

in a short space of time. • The

2AR researcher needs to follow an ethical framework that ensures that the participants'

rights are not compromised. 3

2.2.2 Iterative prototyping Iterative prototyping is a software development method that is based on a cyclic process of proto- typing, testing and obtaining feedback. The

prototype is then refined based on feedback obtained; and then tested again in the

2next iteration [3]. This method is carried under the assumption that 19 the

researcher doesn't know what the final product should look or function like; and whether the target users will find it useful to them. Unlike the waterfall method that delivers the software product to its intended users only when it is complete [4]; in iterative prototyping, the researcher builds the initial prototype that carries only the core features of the desirable software product [3] . The prototype is then taken to its target users for evaluation. The researcher obtains feedback

and uses it to improve the prototype in the next iteration. The number of iterations is usually 2 or 4; where the current prototype has new features and other modifications that have been suggested from the

2previous iteration's feedback. Iterative prototyping helps to uncover software bugs and errors in the early stages of the software development life cycle. The advantage of finding software bugs early is that they are inexpensive to fix [3]. Other benefits of iterative design is that it exposes usability issues early and minimises the chances of user resistance.

3.3 Applying the methods This section describes how we applied action research and iterative prototyping to achieve the goals of our research. We start by describing our participants, who are the domain experts; we then describe the scenarios that we investigated in the research; the motivation behind these scenarios and how the authoring tool evolved from one iteration to the next. 3.3.1 Domain experts Our target participants

2were sophisticated Deaf people and hearing domain experts. Deaf participants are recruited from a non-government organisation (NGO) called Deaf Community of Cape Town (DCCT). The reason for choosing Deaf participants from DCCT is that DCCT has a long term relationship with the University of the Western Cape (UWC), whereby they have been work- ing collaboratively on multiple projects that aimed at empowering Deaf people in their everyday lives.

We also recruited hearing domain experts who specialise in these scenarios: medical dispens- ing, ICDL training and antenatal care. The

2criteria in choosing hearing domain expects is that they must have worked with Deaf people before or have significant knowledge about the communication challenges that Deaf people face when they have to communicate with hearing users who cannot sign.

3.3.2 Scenarios These are the scenarios that we plan to create, using the authoring tool in the testing stage. 20 Medical dispensing: Traditionally, the doctor prescribes the medication and then gives the patient a medical prescription, to take to a pharmacy. The

36patient hands the prescription to the pharmacist, over the counter, at the

pharmacy. The pharmacist can also verify the prescription and contact the doctor if she finds an error; otherwise s/he proceeds with medical dispensing. The pharmacist writes the medical consumption instruction on the label pasted around the medicine container. Lastly, she explains those instructions to the patient, to ensure that the patient under- stands how to consume his medicine. Communication between the patient and the pharmacist is important during medical dispensing because if the patient consumes his medication wrongly, that may lead to serious health hazards. The current SignSupport mobile app is build specifically for this scenario. It has been tested for usability and proved to be very useful for Deaf people. We would like to use the authoring tool to produce the same app as SignSupport that can be ported to other mobile platforms. ICDL course: the IDCL course is an internationally recognised course that teaches

basic computer skills. The modules taught in this course include Microsoft Office units (Word Processing, Spreadsheets, Presentations and Databases); file management; online information and communi- cation skills. This is currently offered at DCCT for Deaf adults and it creates an opportunity for Deaf people to have access to Information Technology and take part in research conducted by UWC and UCT Computer Science Department; for Deaf people. Antenatal care Antenatal healthcare is important for both the pregnant woman and her child. A women needs to be constantly monitored for risks that might harm her or her child as the pregnancy progresses. A research project was initiated which aimed at promoting antenatal healthcare. The approach used by their project was to send SMS notifications to the participants (pregnant women) in the different stages of their pregnancy [24]. The SMS notifications included reminders of visits to the health-care facilities and useful health information that is relevant to their pregnancy stage. This project can benefit from the authoring tool in a way that it can be used to design an antenatal care app which can be populated with pre-recorded signed language videos that can be played back, for Deaf pregnant women who are text illiterate. The domain expert in this scenario can choose between designing multiple apps for different stages of pregnancy or create one app that has information about all stages of pregnancy. 21 3.3.3 Usability testing with domain experts Usability testing helps to measure the ease of use of the prototype, from a participant. Ease of use determines whether the target users will be satisfied with that system [19]. Usability sessions will be conducted with the real target users of the authoring tool, i.e. domain experts. This provides direct information about how the domain experts will use the authoring tool in reality and their problems with the current interface. Figure 3.1: Four iterations of the authoring tool development This figure indicates how each iteration is improved based on the feedback obtained in the previous one. We plan to have four software development iterations, as indicated by Figure 3.1. 1. This prototype is built from the researcher's observation of authoring tools. The main purpose of this iteration is to lay a foundation of what an authoring tool could look like. This will also help to educate domain experts about authoring tools and their use. The prototype enables domain experts to judge and give feedback based on what they see, which is possibly more beneficial than asking them to design an authoring tool interface without an example. Deaf people are our participants when assessing this prototype for usability. 2. Feedback from the first iteration is used to improve the prototype. Here we use hearing professionals as our participants when evaluating the prototype for usability. The criteria in choosing the hearing participants is that they have worked with Deaf people before and/or they understand the communication needs of Deaf people. 3. This is the advanced prototype, which has evolved over the last two iterations. Here we mea- sure if the participants are satisfied with the prototype and whether they see this prototype empowering their professional lives in the near future. All the recommendations obtained here will be treated as the future work of the research. 4. This iteration demonstrates all the steps that one needs to design and render a mobile app, using the authoring tool and the XML parser. The app that is created is an ICDL assessment mobile app. Firstly, SASL video instructions are recorded and prepared to be 22 used in the app. Secondly, the app is designed using the authoring tool to produce an XML file. Thirdly, the data assets (SASL videos and icons) of the app and the XML file are copied to the mobile phone. Lastly, the XML parser is launched to render the expected mobile app. The XML parser processes the XML file to find the location of the app's data assets and to get instructions of what the interface design of the app should look like. 3.4 Ethical considerations This section discusses the ethical considerations that guide the process of dealing with real partic- ipants. The interest is on qualitative data because such data has the possibility to change people's lives [42]. We first present the ethical consideration points under consent form, confidentiality, the use of a SASL interpreter, the risk and benefits of the research. We then state the process of obtaining ethics clearance for this project. 3.4.1 Ethical standards

2**All participants were informed of their role within this project and what was required from them, and also that participation was voluntary. As such all** participants **provided consent that they have been informed and understood what was explained to them. In order to protect any and all information obtained during consultation sessions, all participants were required to sign the consent form, having been translated to them in SASL for the Deaf participants. Confidentiality - Participants were trained on how to use the authoring tool and then asked to design a basic or a domain specific script** dialogue **using this tool. All interaction, e.g. interviews and focus groups, were recorded (paper, audio and/or video) and securely stored on a computer that was protected with a password to uphold privacy. Use of an interpreter - Sophisticated Deaf participants working at DCCT that had** pre- vious **experience using SignSupport were recruited to try out the** authoring **tool. An accredited interpreter was be used to facilitate the communication between the researcher and the Deaf when the interviews and focus groups were**

done. 23 Risk and benefit - This study aimed to improve SignSupport in a way that SignSupport will accommodate different limited domain communication scenarios, multiple signed languages and mobile platforms. Another benefit from this study was that it empowered Deaf people and domain specialists to develop SignSupport scenarios together. The risk was that the script dialogue created from the tool might not cover all parts of the conversation within the limited domain scenario; and this might lead to a communication breakdown between a Deaf and a hearing person. This script dialogue that has been produced by the authoring tool was mainly be used for the purpose of this study and not in the real world.

3.4.2 Ethical clearance process The researcher needs to fill in an ethics clearance application form. Within that application form, the researcher needs to state the description of the project and it's process. The application needs to be approved

6by the UWC Faculty Board Research and Ethics Com- mittees and then by the UWC Senate Research Committee [SR]. SR may also consult outsiders on ethics questions, or consult the UWC ethics subcommittees, before registration of the project and clearance of the ethics.

The general rule is that

25no project can proceed before project ethical clearance has been granted.

SignSupport is our umbrella project that already has ethics clearance. The authoring tool falls under SignSupport; and so the researcher applied for the authoring tool's ethics clearance under SignSupport. 3.5 Summary The main aim of this research project is to

2improve SignSupport and increase its impact on Deaf people's lives based on active engagement with the Deaf community and domain expects. We believe that an authoring tool can make SignSupport both scenario and platform (and even language) independent. The primary interest is

2in evolving the authoring tool's palette and canvas interface in order to better support the design and content population processes with both Deaf and domain expert users.

The above-mentioned methods, together with the ethical considerations, will help to carry out the research and authoring tool development.

2Creating an initial prototype and conduct usability testing sessions to get feedback from our participants

will follow an iterative prototyping method. The

2results obtained in each iteration will be used to inform the next iteration;

24 and participants will be encouraged to

2assess the interface design and to suggest more features that they would like to add on to the authoring tool.

The next

14**chapter presents and discusses the results** obtained **from**

this iterative development. 25 Chapter 4

14**Results This chapter presents and discusses results from**

iteratively developing the authoring tool and test- ing it for usability. Iterative development helps to get participants involved in the development of the authoring tool and to accommodate their different needs. Usability testing helps to improve the authoring tool functionality and UI. We had 3 iterations, which ended with a usability evaluation. In each evaluation, participants created and populated different scenarios using the authoring tool. The purpose of creating a scenario was emphasised as the purpose of the authoring tool to the participants. Section 4.1 presents the first iteration, section 4.2 represents the second iteration and section 4.3 represents the last iteration. Section 4.4 demonstrates the process of designing the scenario using the authoring tool, recording signed language videos and finally generating a mobile application using an XML parser. 4.1 First iteration The information obtained from the literature review and the research questions led us to conclude that we need to create an authoring tool. The authoring tool user requirements were that it should allow the domain expert to independently create and populate a domain specific scenario. The scenario design should indicate the flow of screens and the relationship between them. The au- thoring tool also needs to produce an XML script to be consumed by an XML parser. The XML parser then renders the XML script into a real app on any platform. 27 4.1.1 Interface design The prototype of this iteration has multiple screen templates that are added to the canvas area and then populated with icons and text. Each button is linked to a screen by initially clicking on the button which confirms that you want to create a link. You then click the screen that you would like to link the button with. After creating the link, a blue line is drawn from the button to another screen. The line serves as visible proof that the link has been created. Linking the icon with another screen also indicates user interaction in the scenario when it has been converted into a real app. Here is the screenshot of the interface design in Figure 4.1. Figure 4

1**.1: A screen** shot **of the** first **authoring tool** prototype **while a user (domain expert participant) is testing it..**

The purpose of this iteration was to invite domain experts to take part in designing the au- thoring tool. By first trying out the current authoring tool then give feedback on its usability. The expected feedback from the participants is about how can we improve the authoring tool interface in a way that all its functions are visible and that it's easy to use. After the first prototype was implemented, we then evaluated it's interface design for usability with Deaf domain experts. 28 4.1.2 Experiment design 4.1.2.1 Sample This experiment has 5 Deaf people as participants. All of them work at DCCT, and their job description includes empowering other Deaf clients. The researcher believes that these are the best participants for this session because they experience communication barriers in their daily lives and hence they know the communication needs of Deaf people. We also had an SASL interpreter, who assisted in translating information between the Deaf participants and the researcher. 4.1.2.2 Procedure The purpose of this session is to test the current features of the authoring tool for usability. Features include dragging and dropping icons to screens, adding and removing screen data, links and also different mouse click features. Testing these features will give an idea of what needs to be improved. The procedure for this session was initially divided into three stages, namely training, testing, and feedback. However, it ended up having training and feedback sessions only. This is because only 1/5 of the participants managed to finish building the training scenario. Hence it was concluded that it would be less useful to progress to the testing stage as the training results indicated that the prototype is difficult to use. Here are the details of the training and the feedback stage below: 4.1.2.2.1 Training The training stage was divided into 4 parts, as shown in Figure 4.2. Figure 4.2: Steps of the training stage. Information sheet and Consent form: An interpreter presents the project information sheet to the Deaf participants in SASL. This information sheet informs the participants about the aims of the authoring tool, how it is built and how the participants will contribute to the project. The information sheet also informs the participants that their contribution is voluntary and that all information collected from them will be treated with high confidentiality. After the information sheet has been signed by the participants, they then sign a consent form to witness that they agree to participate in the session. 29 Prototype user guide: The prototype user guide is presented to the participants in a PowerPoint presentation. The user guide shows screenshots of the authoring tool

features as they are being explained. The purpose of the screenshots is to ensure understanding and also to convey visual information. After presenting the user guide, the HIV counseling conversation scenario is then given to the participants to create using the authoring tool. HIV counseling scenario: The HIV counseling scenario is presented in a binary tree flow diagram. It consisted of a series of questions that the HIV counselor would ask a client in order to determine whether s/he needs to get tested for HIV. The client is expected to respond by a yes or a no; so that the HIV counselor can decide which question to ask next. 4.1.3 Results Only one participant managed to finish building the scenario using the authoring tool. This participant was also able to view the conversation scenario design in a PDF file and verified the relationship between the icons and the screens that they linked to. Other participants were confused by the scenario flow diagram, e.g. has difficulty in differentiating between the client and the HIV counselor; and the instructions that were given to them on how they should build this conversation scenario. One participant could not link the icons with the screens. 4.1.4 Analysis The results indicate that the authoring tool prototype was not user-friendly. The number of participants that managed to complete the task given in the training scenario indicates this. The confusion from the diagram shows that the instructions given to the participants were not clear enough. More work needed to be done on the authoring tool interface in order to make it more user friendly and and less frustrating. We also needed to improve the way we explain the features of the authoring tool and how we present task instructions to the participants. 4.1.4.1 Plan for the second iteration We will explain the features of the authoring tool by demonstrating them on a projector. Partic- ipants will also be given a chance to try each feature on their computer, as it was demonstrated on the projector. This will help the participants to have a

> 51**better understanding** on **how to use the** features **of**

the authoring tool. The participants will be reminded about the current SignSupport app. All participants are familiar with this app, so it will be a good example for them. We can 30 then use the authoring tool to create, populate and link a few screens from this app. The next section presents the second iteration of the authoring tool prototype. The feedback obtained from the first iteration was used to improve the authoring tool interface and the way that instructions are presented to the participants. 4.2 Second iteration The participants are changed in this iteration, from Deaf DCCT staff to hearing domain experts. The main reason for this change is that we wanted to address the needs of professionals who are likely to have Deaf people as their clients. The idea here is that the domain experts design the domain specific scenarios and then the Deaf people, and the interpreters, help with SASL video recording. Domain experts knows the set of questions that they ask their clients in order to find ways to assist them and the challenge that arises when both the domain expert and his/her client do not have a language that they both understand. For example, a doctor has basic questions that they ask patients when they're trying to find the disease that they might have. The challenge that the doctor might have is when they have to ask these questions to a Deaf person who communicates in a different language and cannot read or write in a language that the doctor understands. This is the reason why we focused on hearing domain experts for this iteration. 4.2.1 Interface design

> 1**This section describes the features available in the authoring tool. The authoring tool helps a domain expert to design an interface for a limited domain conversation flow between a Deaf and a hearing person. The software presents the domain expert with screen templates. These screen templates can be populated with text, icons and/or signed language videos (that would be provided by native signed language speakers, not the domain experts,**

unless they are Deaf). The

> 1**screens can also be linked to each other in a** graph **structure. The link between each screen indicates the user interaction on the real mobile app i.e. which screen will show next when the end user clicks a** given **button. Figure** 4.2 **presents a screenshot of the authoring tool interface while a domain expert is using it. The interface of the authoring tool is divided into four areas: 1. The button area contains four buttons, namely Add template, Clear all, Add video and Export to XML. When the domain expert clicks on the Add template button, a window pops up and presents all the default screen templates of the authoring tool. The user then** 31 **chooses the template that**

s/he would like to add to the canvas by clicking the checkbox on the left hand side of the preferred template and then clicks the OK button at the bottom of the window. The selected screen template then appears on the canvas as active. The Clear all button clears all the screen templates from the canvas. The Add video button helps the domain expert add videos from the computer

as an asset within the

1authoring tool. The videos line up vertically as they are being added from the computer to the video area. The Export to XML button invokes a method that reads all the screen templates, their assets and connections to each other; and then produces an XML file that includes all of this infor- mation. Since the XML file contains only text, it only stores an asset's information such as file name and relative path, in text. An XML

parser on a target device consumes the XML file in order

1to render a platform-specific mobile app. 2. The video area contains a series of signed language videos that are added by the domain expert from the computer. The videos can be recorded before or after the scenario based conversation flow is designed. Then each video is given an English tag/ file name as a way to help the scenario domain expert include the correct video in the right place [28]. Videos in this area are played (and paused) by clicking on them. They can also be dragged and dropped to other screens. 3. The icons tab area contains icons in various categories. The domain expert can also add new tabs of icons. The icons can be dragged and dropped to the screens,

and linked to other screens and/or actions.

14. The Canvas area is where the selected screen templates appear when added. The domain expert populates a screen by typing in the intended message and dragging and dropping icons or a video to the indicated areas in the screen. If the domain expert drops an icon into the wrong area, s/he can right click on that area and select the remove icon option and the icon will be removed. To link the screens to one another, the domain expert clicks on the icon that represents a button from one screen, confirms that s/he wants to connect that icon to another screen at a prompt dialogue pop-up, then selects the screen to link with. A blue link line will appear in the Canvas, between these two screens. This link gets updated and redrawn as the screens are moved from one position to another.

32 Figure 4.3:

1A screenshot of the second Authoring tool interface while a user (domain expert participant) is testing it. The

1authoring tool is built using the Java programming language, and the output of the authoring tool is an XML file. The XML file serves as a database that stores the design information in text. XML is supported by a variety of technologies in various platforms. This in turn helps us to achieve platform

independence for scenarios designed from the authoring tool.

4.2.2 Experiment design

1**User-based testing** was employed **for** the **usability experiment** [14]. **The goal of the experiment was to evaluate how effective the authoring tool can be in enabling** a **domain expert with little or no programming skills, to create** a **limited domain conversation**

scenario. The main focus is

1**how the domain expert searches for icons from the icon tabs, adds and removes icons and videos from the screen templates, recovers from errors; reuses screen templates in the same scenario and reuses the same icons in different screen templates are also observed.**

The domain expert was also encouraged 33 to give suggestions on how the prototype can be improved to better suit their needs. 4.2.2.1 Sampling Participants were selected through a purposive sampling method.

1**Purposive sampling is** purpose- fully **selecting participants in terms of the qualities/skills they have** [41]. **The communication flow design scenarios for the experiment had already been defined, as the medication dispensing process and the (ICDL) course. The pilot session of the authoring tool was evaluated with the aid of an industrial design engineer.** Participants recruited were **a pharmacist for the pharmacy setting scenario, a educator** of Deaf adults **and a computer science researcher for the ICDL course scenario for the actual usability testing sessions. All participants have unique work experience with Deaf people. The industrial design engineer designed the SignSupport mobile app and the pharmacist is involved in the co-design and co-** testing **of the previous and current** versions **of the SignSupport mobile app with Deaf users. The educator** of adults **gives ICDL lessons in SASL every** Thursday **at**

a local Deaf community; and the

1**Computer science researcher works on** another **authoring tool for creating ICDL lessons** only. **All our participants are the potential users of this authoring tool and hence ideal candidates for the experiment.**

4.2.2.2

1**Testing procedure The testing was divided into three stages, namely training, testing and gaining user feedback. One participant attended the usability test at a time. During the training stage, the participant was introduced to the authoring tool prototype and all the features were demonstrated. The researcher then encouraged the participant to try out the features on their own e.g.** add **screens to the canvas area, drag and drop icons and videos to the indicated area, and link icon labels with other screens. Afterwards a training exercise was given to the participant to design** a **HIV** counseling **scenario. This applied to all participants. This scenario was given to the participants as a decision** flow graph **diagram, as shown in** Figure 4.2. **During the testing stage, participants were** first **asked to design and**

populate a limited scenario that was specific to their domain. The participants from the pharmacy scenario were asked to design a few screens from the familiar **SignSupport mobile app for medicine dispensing. The participants from the ICDL course scenario were asked to design a specific lesson from the ICDL course book, which can be populated with SASL video instructions. The participants were assigned the tasks specific to their domain to give them a feeling of how they would use the authoring tool to best suit their needs.** This session **was**

1**video recorded. The video recorder was directed to the computer** 34 **screen to record how the participant** interacted **with the authoring tool. Participants were asked to use the think** aloud **strategy to voice out what they**

were

1**doing and all the thoughts that** came **to mind as a result of using the tool. After the testing stage, the researcher conducted a semi-structured interview with each participant. The main questions asked in the interview were: • What did you like/dislike about the authoring tool? • What did you find easy to use when building the scenarios? • What did you find confusing or difficult? • Any features you would like to be changed or added to the authoring tool? Data collected from the trial** ave **summarised and organised according to the four areas of the authoring tool interface, which is stated and discussed in the** results **section below.**

4.2.3 Results

1**All participants finished the training scenario successfully and with no difficulties. They all** repeat- edly **used one screen template with the same icons for the training scenario design. All of them also managed to create and populate the testing scenario design.**

The

1**participants feedback from both the pilot and the actual usability test session is described in accordance with the aforementioned** four **areas of the authoring tool interface features. Buttons area Here**

is the description of

1**feedback from all the participants regarding the available buttons in the button area. All participants understood the use of each button on the interface. 1. Add screen: All participants praised the interface and notification designed for the add screen and its process. The selecting process** was **easy to understand, the background colour of the selected screen template changed as it was clicked, and this assured the participants that the template was indeed selected.**

2. The

1canvas was cleared immediately after clicking the Clear all button; and the XML output was created and saved to a file after clicking the Export to XML button. Two participants were also interested in seeing what the XML output looked like. 35 3. Suggestion on the additional button: Two participants said that there should be an Add picture button that they can use to add pictures to the screens, which works similarly to the Add video for adding videos to the screen.

1Video Area 1. Adding video to the video area: All participants found the video addition process to the video area easy to do. 2. Suggestion of a scroll-bar: One participant suggested that there should be a scroll-bar in the video area, so that multiple videos which will be added to the video area can be seen. 3. Suggestion of video controls: One participant suggested that there should be video control buttons that show on each video in the video area, when the mouse is hovered over them, so that he could click on these video controls to play each video. Icon tabs area All participants could navigate through the icon tabs easily. Names of all the icons appeared when the participants moved the mouse over them. Two participants commented that the tool-tip assured them that they were choosing the icon with the correct semantic meaning to drag to the screens. Two participants struggled to find the home and the exit icons

as they expected to find these icons under the navigation icon tab instead. 4.2.3.1

1Canvas Area 1. Adjustment of the screen and its component sizes: All participants disliked that the compo- nents of the screen remained in the same size and position when the screen was resized. They then had to resize and position each of them one by one. So, two participants suggested that the whole screen and its components should be proportionally resized all at once. 2. The design of the default screen templates: One participant found the default screen tem- plates limited in terms of design. Most of the component areas were fixed in their position, which did not allow the user to modify the default design. One participant mentioned placing the next and back buttons on the sides of the screen, whereby the default screen template did not match this need. Therefore, she suggested having a blank screen template available for users to customise to their own screen design(

s). 36 3.

1Screen links: All participants found the screen linking option easy to understand and use; however, they preferred that the linking process be done with fewer mouse clicks. The linking line should go around the

1screens instead of going underneath them. With the suggested linking line, it would be easier for the user to trace back and recheck each link. In addition, one participant suggested that the line indicating the link should have a message that indicates its origin and the destination of linking. 4. Drag and drop option: All participants said that they liked the drag and drop options for adding icons and videos to the screen. It was observed that all participants missed the icon drop area several times when they were populating the

screens with icons. However, they tried again even though the authoring tool did not give them any warning/notification

Table 4.1 lists features that the participants suggested the authoring tool should have. Partic- ipants made these suggestions during the feedback session, when they were asked about what they would add to the authoring tool. These features will be added to the authoring tool and will also be evaluated for usability in the next iteration. Table 4.1: Recommended features and their motivation Feature Motivation Cut and paste To duplicate screens that are already populated; and modify them for a similar purpose so that there wont be a need to start over all the time. User guide and help function To enable the author to learn to use the author- ing tool independently and to get assistance in the absence of the researcher. Tooltip text on Buttons and mouse-hovers Tooltip text on a

1**button or mouse-hover** option reminds **the** author **what each button or option is used for.**

Scenario design preview This gives authors an idea of how the scenario would look as a real app and also to confirm that the links have been defined correctly. Undo and redo This option helps to recover from errors or to repeat recent events. 37 4.2.4 Results analysis The results indicate

1**that there are several** additional **features required from the authoring tool to improve its usability. The robustness of the authoring tool needs to be thoroughly evaluated before taking it to its potential users** for usability testing **of the**

next iteration. The

1**creation of the three limited domain scenarios (HIV pre-** counseling, **ICDL course lessons and rebuilding a thread of the SignSupport pharmacy app)** demonstrates **that the authoring tool can generalise**

SignSupport scenario creation for limited domain communication between Deaf and hearing people. The

1**reuse of screen templates and data assets (icons and videos),**

combined with the production of an XML file representing a given scenario, appear to indicate

1**that the authoring tool can generalise** the **SignSupport mobile app for multiple signed languages and mobile platforms. The next** iteration **is** for improving **the authoring tool by fixing its errors and** adding **features recommended by the participants.**

Another usability testing exercise will then be conducted

1**with more participants and** one **more limited domain** scenario. **Thereafter, one complete limited domain scenario**

will be built using the authoring tool, with recorded

1**SASL videos with the help of a Deaf person and a SASL interpreter. That**

requires a fully functioning XML parser. The XML parser will enable the authoring tool to produce a mobile app that can be run on a mobile phone. Sign language videos will be recorded separately, and then

integrated into the scenario. 4.3 Third iteration Here we aim to confirm usability and that the participant see the authoring tool as useful to them. The last iteration focused on how the participants navigate through the authoring tool interface when building. 4.3.1 Interface design We addressed the interface design issues that were raised in the previous iteration and we also added a few more features that optimise scenario creation and hints the author as they use the authoring tool. The buttons now have tooltip text, which appears on mouse hover. Tooltip text informs the author about the purpose of the button. The links between the screens are changed from lines to arrows. This enables the author to follow the flow of the scenario by following the arrows from one screen to another. The screens can also be cloned in order to avoid having to recreate similar screens. Other parts of the interface design remained the same, as the participants, 38 in the last iteration, were all satisfied with it. Here is the interface design of the authoring tool in Figure 4.4. Figure 4.4: A screenshot of the new authoring tool interface with text tooltip showing on mouse hover and a clone button that is used to create a copy of the screen. The participant (domain expert) is testing it for usability. 4.3.2 Experiment design We evaluate the authoring tool for usability using the user-based testing method. The aims of our experiment are: • Confirm the authoring tool ease of use. • Understand how the participants view the authoring tool. • Give participants an opportunity to rate usability of the authoring tool • To get recommendations on how we can improve the authoring tool in the future. The experiment consists of three parts, which are the sampling, testing procedure and data collection. 39 4.3.2.1 Sampling Purposive sampling is used again in this iteration. The participants are the computer science researcher, educator and the pharmacist; who participated in the authoring tool evaluation of the previous iteration. We also have two SASL interpreters, who have many years of experience in- terpreting for Deaf people and also working in various projects that empower Deaf people. The experiment consists of three scenarios: HIV pre-counseling, medical dispensing, ICDL course and antenatal care. The HIV pre-counseling scenario is used to demonstrate the authoring tool to the participants, the other three are used for the actual evaluation. For the convenience of this exper- iment, the pharmacist, educator, computer science researcher, the first and the second interpreter will be referred to as participant A, B, C, D and E respectively. 4.3.2.2 Procedure This procedure has training and testing stages; and this session was conducted with each partic- ipant independently. In the training stage, we initially remind the participants what the project is about and what value s/he be adding to the project. The participant signs a consent form as an indication that s/he agrees to participate. Thereafter the researcher explains all the features that the authoring tool has, by adding screens, populating them with videos, text and icons; and linking them while the participant is watching. The participant is then given a chance to tryout each feature that the researcher just demonstrated. The participant is then given an HIV pre-counseling scenario in the form of a printed diagram on paper. The researcher explains the flow of the scenario diagram to the participant before s/he can get to the authoring tool. This helps to ensure that the participant understands the scenario design and its conditional branches. The participant is allowed to ask for help and to ask questions at this stage since the aim here is to teach him/her how the authoring tool works. The testing involves building a different scenario independent of the researcher's help. Here the participant is asked to use the think aloud protocol as s/he is building the scenario. Participant A builds the medical dispensing scenario; B and C build the ICDL course scenario as in the previous session. Participant D and E build the antenatal care scenario. 4.3.2.3 Data collection This experiment is recorded with a video camera, which is pointed to the computer screen as the participant is interacting with the authoring tool. The video camera records the participants 40 voice as s/he uses the think aloud protocol.

> 43**A semi-structured interview** is **conducted with each participant**

when they finish building the testing scenario. Here are some of the

> 1**questions asked in** this **interview: • What did you like about the authoring tool? • What did you**

not like? • What would you like to change? • Can you explain the authoring tool in your own words? • Please rate the authoring tool's usability from 0 - 10 • If the authoring tool was a car, which car would it be, and why? • How does this authoring tool compare with other similar tools that you know? • Would you recommend this authoring tool to other professionals who have contact with Deaf clients? All these questions are asked in order to ensure that the participant understands the purpose of the authoring tool and to get feedback on whether they think the authoring tool is useful. The results are stated and discussed next. 4.3.3 Results The results are divided into two parts, which are the interface design and participants' response to the interview questions. The results of the interface design are presented according to the sections of the user interface. The responses of the participants are presented according to the questions. 4.3.3.1 Interface design • Buttons area: All the participants added the screens successfully by clicking the "add screen" button. The researcher observed that 2 participants clicked on the screen multiple times instead of clicking the checkbox area. Only 3 participants used the "Clone

screen” button to replicate the screens. One of these participants couldn’t see cloned as she expected it to appear next to the original one. This made her pause for some time and search for the cloned screen around the canvas. Adding a video from the computer to canvas, using the ”Add video” button was simple. The only issue was that the videos took long to be loaded to the video area such that the whole authoring tool becomes non-responsive. One partic- ipants suggested that multi-threading should be used so that the user can do other tasks 41 in the authoring tool while the video is loading. The ”clear all” and the ”Export to XML” button worked well when they were used. One participant suggested that there should be a progress bar that appears as the XML file is being created; and lists all the steps taken to create an XML file. This keeps the user informed about the progress of creating an XML file. • Video area: Videos had playback controls (rewind, play/pause and forward) which showed at the bottom centre of each video. One participant suggested that the names of the videos should appear on mouse hovers so that they can remember what the video is about, especially if it’s a signed language video. • Icon tabs: All participants could navigate through the icon tabs and could find their desired icons for the screens that they had. 3 participants praised the drag-n-drop feature for the icons. • Canvas: All participants could easily build and finish building scenarios on the canvas. This includes adding screens and populating them with data and linking them. It was observed that participants could easily move, resize and delete screens that are in the canvas area. Deleting icons, links and videos from the screens was also easy to do, as observed by the researcher. One participant mentioned that she finds it difficult to remember when to left or right click when adding a link or deleting an icon. Removing the link was confusing at times, as the 3 participants try to click on the link arrow in order to delete it. However, 2 participants, saying that they make the scenario design easy to follow, praised the arrows that represent the links. The colour of the canvas area was too similar to the added screens, as said by one of the participants. She then asked that the screens and the canvas area should have distinct colours. One last suggestion for the canvas area was that there should be a space where the authoring tool user can write the title of the scenario. 4.3.3.2 Participants’ view of the authoring tool The participants were asked to define the authoring tool in their own words. The aim of this ques- tion was to reveal their understanding about the authoring tool. Their definitions are represented in Table 4.2. The definitions given by the participants indicate that they understand that the authoring tool aims to empower Deaf people in multiple communication scenarios between themselves and hearing people. Two participants also mentioned, in their definition, that the scenarios created on the authoring tool will be installed on the mobile phone for Deaf people to access. These definitions can be summarised into one definition: The authoring tool helps a domain expert 42 Table 4.2: Participants’ definitions of the authoring tool Participant Definitions A Allow you to independently create a conversation scenario for a Deaf and hearing person. This is achieved by populating screens with information and then link them according to the screen options. B Build customized lessons for computer skills training, so like you can build sign language lessons for Deaf people so that they are appropriate and user-friendly them. C Structure content for Deaf people to consume on a mobile phone. You can also build different communication scenarios for Deaf people. D Used to create multiple scenarios to accommodate Deaf people, e.g. Deaf education or scenarios that raise awareness for Deaf people in any sign language. This tool also benefits hearing people who have Deaf clients in their professional lives. E Create scenarios for the Deaf and hearing people to communicate when the interpreter is not present. The scenarios that the tool creates are stored on a mobile phone and acts as an interpreter on a Deaf persons mobile phone. to create multiple communication scenarios, in multiple sign languages, for Deaf people to access on their mobile phones. This is similar to the research question of this project, stated in Chapter 3. Participants gave the authoring tool a score (0 - 10) for usability, as indicted in Figure 4.5. The average of these scores is 7.1. This clearly means that the participants are

> 39**satisfied with the user interface** design **of the** authoring tool. **The**

general comment was it is simple and very easy to use; and all the features are visible. Below is

> 27**Table 4.3** that **represents the** participants’ **responses when** they **were asked** to famil- iarise **the**

authoring tool with a car or a form of transportation that they know. This

> 40**question was** asked **to** understand **the participants’ perception of the**

authoring tool. 43 Table 4.3: Authoring tool related to cars. Participant Transport Motivation A VW Golf The tool is reliable and will gain popularity as time goes by. B Qhubeka Bicycles Very basic and works really well. C Citi golf Basic tool with normal functions to take you from point A to B. Not fancy just the main purpose. D New avanza All the features are visible and it aims to accommodate multi- ple signed

languages and mobile phones. E Old 5-gear car Easy to use and it does everything as expected. Participant A believes that the authoring tool will be widely known for its reliability and usefulness in the lives of Deaf people. She also mentioned that the authoring tool needs to be advertised to the officials who specialises with content delivery to Deaf people (DeafSA and other Deaf NGOs), she feels that those are the people who will find the authoring tool useful. Domain experts like pharmacists and police might be resistant towards using the authoring tool as they might feel that it is not their ”job” to ensure that Deaf people understand them. Participant B finds the authoring tool very basic but working very well. She says that all the features of the authoring tool makes it so easy to create a scenario in a way that one can create the scenario on the authoring tool, without having to draw it out on paper first. Participant C and E feel that they authoring tool fulfills its purpose but it can be improved by adding text or multimedia formatting features. Participant D likes the idea of accommodating multiple signed languages and mobile phones although she was disappointed that the authoring tool does not produce a real mobile app. 4.3.4 Analysis The participants' feedback seems to be more and more positive as the authoring tool is being tested and improved; from one iteration to another. The interface design of the authoring tool seems to have standardised as well as the participants didn't mention any part that needs to be changed. The next step is to add more features whiles trying to keep the interface design the same. The next section demonstrates the steps needed to create an app by using an authoring tool and an XML parser. 44 4.4 Fourth iteration This iteration uses an authoring tool to design an ICDL exercise app and then using the XML parser to render its XML output into a mobile app. 4.4.1 Process Video recording and editing The authoring tool requires that the domain expert should have the SASL videos already recorded. So the first step was to write down the exercise questions and then record SASL videos for each of them. The videos were recorded with the help of one of the Deaf learners and an educator. The educator read the questions in English text and then signs the question to the Deaf learner, and then the Deaf learner would sign the question to the video camera. The reason we asked an educator instead of a SASL interpreter is that sign language has different signs for one item and so a SASL interpreter might use a sign that the target Deaf learners might not be familiar with. This might lead to confusion and hence affect the learners' grades when they do the exercise. After recording the videos, we needed to edit them and prepare them to be used in the mobile app. Sound was removed completely from the videos as it was not needed. We then changed the videos to black and white so that they take less memory on the mobile device. Lastly, we cropped the videos to a portrait orientation so that they are not squashed but appear clearly on a mobile phone's screen. Each video was given a unique name, which was the keyword of the question e.g. Bold, Underline or Save. Figure 4.6: Signed language video editing The original video in the left was edited to black and white to reduce its size; cropped around to best fit a mobile phone's screen and also had its sound muted. Scenario creation We imported all the ICDL videos to the authoring tool. We then added screens to the canvas area and populated them with icons and videos. Each screen was given a 45 unique name, which was the keyword of the question e.g. Bold, Underline or Save; and had two navigation icons (”back” and ”next”). So the screen that represents current question links its ”back” icon with the screen that represents the previous question; and then links its ”next” icon with the screen that represents question that follows. Each screen also had a text field that states the question that is being signed in the SASL video; in English text. The English text would be helpful to verify that the correct video is used for all the questions. The ”Export to XML” button in the authoring tool to get the XML file that represents the ICDL exercise design, from the authoring tool. We then copied the XML file from our computer onto a mobile phone to be consumed by the XML parser (see appendix D for algorithm). We also copied the necessary SASL videos and the icons to the mobile phone so that the XML parser can find them when generating an app. 4.4.2 Results The XML parser uses fixed file path to get the file and its assets. Every time the researcher needs to launch the generated app, he had to launch it via launching the XML parser. This is because the XML parser rebuilds the app every time one clicks on the XML file. The XML parser does not produce the source code for the rendered application so you cannot edit its interface design. The only thing that can be changed is the XML file. The XML parser also assumes that the scenario screens follow each other sequentially. This might not always be the case because in some scenarios, the author might want to link screens that are far apart but the XML parser will not cater for that. 4.4.3 Analysis XML parser seems to have its own predefined interface for rendering apps and it then uses the XML file as an address to get the necessary assets to be included in the rendered app. It does not give an option for the user to modify the app's interface design or source code. This has potential problems because the rendered app can neither be edited nor installed to other mobile phones because it depends on the XML parser since it is rebuilt every time the XML parser is launched and that the XML parser does not generate source code for the app. 46 4.5 Discussion The interface of the authoring tool improves from one iteration to another. This can be witnessed by the screenshots shown in the interface design of each iteration. The failure of the first iteration was a big learning curve for the researcher and it helped him to prepare better for the other iter- ations. The second and the third iteration stimulated creativity within the participants, as they were able to think of other features that can be added to improve the authoring tool interface. Using the SignSupport mobile app helped the participants understand better the purpose of the authoring tool. Some of the interesting observations were that in the usability session of the first iteration, we had five participants in one room but only three of them gave feedback; and the other two remained silent up to the end of the session. The procedure was changed in the second and the third iteration in a way that each

participant evaluated the authoring tool prototype at their own time. All participants gave feedback freely and even gave recommendations on how the authoring tool can be improved. This can be said to be a proof that conducting usability sessions with one participant at a time is more beneficial than having more then one participant in the same room. Applying the Think aloud protocol would have been impossible as well, if we had more than one user at the same time. Participants were not happy when the authoring tool didn't show them a preview of their scenarios after creating them; or generated an app that they can install on their mobile phones. The three main research questions seemed to have been answered in the results. The multi- ple scenario creation can be witnessed by the success of participants' scenario creation task. The scenarios created had unique content, in videos and text. This can be said to address language independence because all spoken languages can be represented in text and the content of the spe- cific signed language can be recorded into videos. Cross platform independence relies mostly on the XML parser as the authoring tool only produces an XML script. 4.6 Summary Results were not positive in the first iteration but they became better in the second and third iteration. The authoring tool user interface in the first iteration could only accommodate icons and text as the scenario content. The user interface in the second iteration was more visual and 47 could also accommodate video content within scenario creation. The user interface in the third iteration had tooltip text and the links between the screens were arrows instead of lines. The arrows made it easier for the domain experts to follow the scenario flow. The domain experts were mainly hearing people who work or do research for Deaf people. The input from the participants exposes what needs to be improved in the authoring tool interface. The next chapter concludes the research and provides direction for future research. 48 Chapter 5 Conclusion The research conducted herein leveraged multiple approaches to content authoring and generali- sation; and further that software generalisation can improve accessibility and affordability for the ultimate end users. The research aimed at generalising the SignSupport for multiple scenarios, languages and mobile platforms. Four iterative steps were taken (building a prototype, testing it for usability and using feedback, from the participants, to improve the prototype for the next step), guided by Iterative software development and Action research method, in order to achieve this. Generalisation techniques were investigated in related work and some were adopted in carrying out this research and building the authoring tool. Data collection focused on usability and scenario creation; and was done in three iterative steps. Data collected in each step was used to improve the authoring tool for the next iteration. The last step of the research was to design and render an ICDL scenario into a real mobile app. This was done in order to create a step-by-step process on how to design and render a scenario into a real app; using the authoring tool and an XML parser. The results from the first iterative step were that the authoring tool interface was not user friendly for the participants. The authoring tool was then redesigned into a more visual interface, which mainly used point-and-click and drag-and-drop functions. The second iterative step was conducted with hearing domain experts who work with Deaf people. The scenarios created were HIV pre-counseling, medical dispensing and ICDL course. Results showed that the authoring tool can generalise SignSupport for multiple scenarios as all these scenarios were created successfully with the authoring tool. The feedback from the participants was mainly about the interface design that needed to be improved. Some of them also gave recommendations on features that can be added to the authoring tool. The third iterative step was more about acceptance testing and usability. The questions asked to the participants were aimed at ensuring that the participants 49 understand the purpose of the authoring tool and the challenges that it tries to tackle. For example, participants were asked to explain the authoring tool in their own words and to compare it with other similar software that they know. The results from this iteration proved that the participants approve the authoring tool as useful, and sees it as a powerful tool in the future. The fourth iteration demonstrated the steps on building a limited domain scenario app, using the authoring tool and an XML parser.

is: How can we generalise SignSupport for multiple limited domain scenarios, signed and spoken languages; and mobile platforms? This question was then broken into three parts, as presented below. 5.1.1 Limited domain scenarios The current SignSupport mobile apps only accommodated one scenario (medical dispensing and computer literacy training). The approach used in the authoring tool in order to enable multiple scenario creation was to have customisable templates that can be populated with scenario specific content such as text, pictures, icons and videos. These templates can also be linked together in order to define a flow of the scenario. The content differentiates the scenarios from each other but the templates remained the same. 5.1.2 Signed and spoken languages Text, pictures and icons represent the spoken languages in the scenario created; and text is mainly for hearing experts who cannot understand signed language. A scenario designer can represent any spoken language with text. Any signed language can also be recorded into videos for Deaf people to place back later. Representing signed and spoken languages with videos and text; in turn, makes the SignSupport language independent. 5.1.3 Mobile platforms The scenario designed in the authoring tool is converted into XML. XML contains the information about the templates used in the scenario and the data which they were populated with. XML

41**is a cross-platform language that is** commonly **used**

for data transportation and database representa- tion. An XML parser, used in the fourth iteration of the previous chapter, uses the XML produced 50 by the authoring tool to render a platform dependent app. Hence, the XML parser is outsourced in order to accommodate multiple mobile platforms. 5.2 System specifications and limitations This section discusses the specifications of the authoring tool, from the final iteration, and its limitations. 5.2.1 System specifications The authoring tool has default screen templates that the author can add and populate with text, icons or videos; in the canvas area. These templates can also be linked together to define a flow of the scenario. One icon on a screen is linked to another screen to indicate which screen will show next if that icon is clicked, when the scenario has been rendered into a real mobile app. It was observed from SignSupport that the scenario is not always linear and that a users chosen option, in the current screen, determines which screen will show next. One example could be if the user chooses the female icon in the screen that asks for gender, the screen that will show next will ask if the user is pregnant, otherwise a different screen will show. Here is a summary of other system specifications: 5.2.1.1 Button area The button area has five buttons, namely: Add screen, Clone screen, Add video, Export to XML and Clear all. Add screen button is used to add screen templates to the canvas area. When the author clicks on this button, a dialog box appears that shows five different screens, the first four screens were most repeatedly used in the SignSupport mobile app. The fifth screen template is an empty screen that allows the author to design a new screen from scratch, by choosing elements (labels, buttons, radio buttons, text-box, check-boxes etc.) to add onto that screen. Clone screen allows the screens to be duplicated by first selecting the screen, on the canvas, to clone and then clicking the ”Clone screen” button. Add video allows the author to import a video from the computer to the authoring tool. Export to XML button gathers all the content of the scenario in the canvas area and creates an XML file that contains the same content and structure as that scenario. This XML file is then used by the XML parser to render the scenario into a real app, as explained in the fourth iteration of the previous chapter. The Clear all button clears the canvas area by removing all content that the canvas area contains. 51 5.2.1.2 Video area The video area shows a vertical list of all the videos that have been imported from the computer to the authoring tool. Here, each video has its own controls, so the author can play/stop, fast forward or rewind a video. This is useful in ensuring that the correct video was imported. 5.2.1.3 Icons area These are the default icons that the authoring tool comes with. They were taken from SignSupport. The icons are divided into different categories: information, general, navigation, Health and Time. All icons in these tabs are added to the screens through drag-and-drop. The author can also add a new tab or icons if s/he would like them to be in the scenario. All the newly added icons can also be added to the screens in the same way as the default ones. 5.2.1.4 Canvas area This is the area where the scenario creation happens. Screen templates are added and populated in this area. Screens can be removed individually by clicking the button

42**on the top-right-hand corner of the**

screen. Icons and links added to the screens can be removed by right clicking on the icon and selecting an option to remove an icon or link. This area has a vertical and a horizontal scrollbar, which can be used to show extra space on the canvas area if the visible area is not enough for the scenario being created. In order to remove every element on this area, you use the Clear all button, as explained in the button area. 5.2.2 System limits The authoring tool has a number of limitations, although it has proven to be very effective in enabling the author to create a limited domain scenario. Here are some of the limitations of the authoring tool: 5.2.2.1 OS dependent The authoring tool was developed and tested in a windows computer. All the user trails were also conducted in a windows computer. The authoring tool showed negative results when it was installed in another operating systems: the interface of the authoring tool changed and was less user friendly; and the drag-and-drop process was not smooth. It can be concluded from this that the authoring tool works more effectively when running on windows. However, this can be changed in the future by making the authoring tool platform independent. 52 5.2.2.2 Java dependent The

1**authoring tool is** mainly **built using Java programming language,**

videos are played using the library called VLCj

1**and the output of the authoring tool is an XML file.** VLCj enables **the**

authoring tool to play modern multiple video formats because the generic Java media libraries have

limitations. The

1**XML file serves as a database that stores the design information in text.**

This file contains all the data from screen templates like text instructions, icons and video file names. The author needs to install Java in his/her computer before s/he can run the authoring tool. VLCj library is made part of the authoring tool but it also depends on Java. 5.3 Lessons learned 5.3.1 Research with human participants There a number of formal steps that need to be followed when conducting research with human participants. Firstly, the research project needs to get ethics clearance. The ethics clearance is a guidance that ensures that the research results in benefits and minimises harm to the participants. This includes ensuring that the participants are well informed about their role in the research and how their information is going to be used. This helps the participants

34**to make an informed decision** on **whether they** should **participate in**

the research. Ethics clearance also protects the researcher should the participants make a claim against the research. Here are some of the accepted ethical standards: Consent form: informs the participants about the research and whether the research is voluntarily or they will be paid for participating. They are also informed about the risks and benefits of this research to them. The consent form is necessary because the participants have the right to know who could have access to their information and what is their role in the research. Confidentiality: Everyone have a right to privacy, according to the Constitution of South Africa. Participants have the right to know how their information is going to be stored, used and presented in the final report. 53 5.3.2 Generalisation activities Generally, software is built to run on a specific platform and hardware. If people see that software as useful, then they see the need to buy that specific device in order to have access to the software. The challenge comes when one cannot afford to buy that device. This clearly means that person would have no access to that software. So software generalisation helps to increase its accessibility. Software generalisation can be done to achieve a number of goals. Some of the common goals for generalisation include hardware, Operating system and human language independence. Hardware independence is about making software to run on machines with different processing capabilities or created by different manufactures. Operating system and human language independence includes creating software can be installed in different operating systems and usable by end-users who speaks different languages respectively. An example of language independent software is the ATM software. This software enables its target users to choose the language that they best understand in order to do banking. There are number of approaches to achieve generalisation. The same software can have different flavours for different operating systems. This means that a computer with any operating system can install that software and the end-user will have access to all its features. This activity takes time because for the same software needs to be created from scratch for each operating system. Another approach is by creating a web app, which can be accessed by any computer with a web browser. The advantage of web apps is that they are device independent. Their disadvantage however, is that they have limited access to the device's hardware features. The generalisation approach used in this research is creating an authoring tool that a domain expert can use to

1**design a front-end interface for a communication flow between a Deaf and a hearing person; in any limited domain scenario.**

The authoring tool then produces an XML script that can be rendered by an XML parser into a platform dependent app. This approach has two major advantages: it achieves technology accessibility for Deaf people because they just need any mobile device with a bigger screen and high resolution, so that the signed language videos are clear to see. Another advantage of this approach is that it empowers the domain experts and Deaf people. Domain experts benefit in this by communicating better with their Deaf clients, and Deaf people also play a role in the signed language video recording and verification. 54 5.4 Recommendations and future work This section discusses the features of the authoring tool that can be implemented in the future and approaches to conduct research under assistive technologies for Deaf people. 5.4.1 Authoring tool features A scenario preview could be made the same as the PowerPoint slideshow. The author can show which screen should the slideshow begin at. The icon(s) clicked in the currently displayed screen will determine the flow of the scenario preview. This will help the participants confirm their sce- nario design and the links between the screens. Screen templates that the authoring tool provides can be customised in different ways to suit the needs of the author. Having to re-customise the screen template every time you create a sce- nario might have a negative impact on the author. A screen palette can be developed in the future where customised screens can be pulled to, and saved for future use. This will optimise the time an author takes to finish creating the scenario. This also gives a sense of

ownership to the author because s/he can either modify or create new screen templates that suit personal needs. The current authoring tool cannot save a scenario for later retrieval. This makes it difficult for an author who would like to close and continue with the scenario later. A Save and retrieve feature could be implemented to allow the author

46**to save the current state of the** scenario **and**

retrieve it later. An XML file produced by the authoring tool could also be used in order to trace back the state and the content of the scenario when retrieved later. The authoring tool connects to a webcam. The webcam could be used to record signed language videos while a scenario is being designed. A Deaf person or any person that can sign fluently could do this. The recorded signed language videos could also be edited within the authoring tool and then added to the scenario being created. Once the scenario is complete, it can be installed to a mobile phone, together with the recorded signed language videos, and be made available for use immediately. 55 5.4.2 Conducting research for Deaf people The researcher helped with ICDL classes every Wednesdays or Thursdays at DCCT. This helped him to engage with Deaf people and understand the culture and appropriate approach when com- municating with Deaf adults. This is important for a researcher to have this background knowledge about Deaf people as they interact differently than hearing people. One example is that looking at a Deaf person in the face means that they have all your attention. Conducting a research session with a Deaf people requires that you hire a SASL interpreter. The interpreter should be qualified to interpret for Deaf people. It is best to ask the Deaf com- munity to recommend interpreters for you so that you are sure that the interpreter understands all the dialects that your Deaf participants use. Institutions need details of the interpreter before they can pay them and some institutions only pay on a certain date of the month. It is important to get all the necessary details from the interpreter before the day of the research; and also inform the interpreter if your institution pays salaries only on specific dates of the month. The prototype that you create needs to be tested by the actual target users so that you can measure its usability. If your target users are Deaf people then you need to ensure that your prototype interface is visual and has minimal text. It is also important to include the stakeholders in all the steps of your research so that they can give you feedback early. They can help to verify your research plan and help you rehearse your research sessions. This will help you to adjust your plan early and avoid mistakes. Presenting your research plan to a person who has experience working with Deaf people can also give you valuable feedback. 56 Bibliography [1] Avison, D. E., Lau, F., Myers, M. D.,& Nielsen, P. A. (1999). Action research. Communications of the ACM, 42(1), 94-97. [2] Barrett, H. C. (2000). Create your own electronic portfolio. Learning and leading with tech- nology, 27(7), 14-21. [3] Basil, V. R., & Turner, A. J. (1975). Iterative enhancement: A practical technique for software development. IEEE Transactions on Software engineering, (4), 390-396. [4] Bassil, Y. (2012). A simulation model for the waterfall software development life cycle. arXiv preprint arXiv:1205.6904. [5] Blake, E., Tucker, W., & Glaser, M. (2014). Towards communication and information access for Deaf people. SACJ, (54), 10–19 [6] Bourret, R. (1999). XML and Databases. http://www.rpbourret.com/xml/XMLDatabasesProds.htm [7] Burgstahler, S. (2009). Universal design of instruction: Definition,principles, guidelines and examples. Retrieved from www.washington/doit [8] Cavender, A., Trewin, S., & Hanson, V. (2014). Accessible Writing Guide. Special Interest Group on Accessibility in Computing. Retrieved from http://www.sigaccess.org/welcome-to-sigaccess/resources/accessible-writing-guide/ [9] Chininthorn, P., Glaser, M., Freudenthal, A., & Tucker, W. D. (2012). Mobile Communication Tools for a South African Deaf Patient in a Pharmacy Context. In P. Cunningham & M. Cunningham (Eds.), Proc. IST-Africa. Dar es Salaam, Tanzania; Dublin: IIMC International Information Management Corporation. [10] Farkas, K. I., Flinn, J., Back, G., Grunwald, D., & Anderson, J. M. (2000). Quantifying the energy consumption of a pocket computer and a Java virtual machine. ACM SIGMETRICS Performance Evaluation Review, 28(1), 252–263. 57 [11] Glaser, M. (2000). A field trial and evaluation of Telkoms Teldem terminal in a Deaf commu- nity in the Western Cape. In 2nd South African Telecommunication Networks and Applications Conference. Stellenbosch, South Africa: Pretoria: Telkom. [12] Glaser, M., & Tucker, W. D. (2004). Telecommunications bridging between Deaf and hearing users in South Africa. In Conference and Workshop on Assistive Technologies for Vision and Hearing Impairment (CVHI). Granada, Spain. [13] Glaser, M., & Lorenzo, T. (2006). Developing literacy with Deaf adults. In B. Watermeyer, L. Swartz, T. Lorenzo, M. Schneider, & M. Priestley (Eds.), Disability and social change: A South African agenda (pp. 192–205). Cape Town, South Africa: HSRC Press. [14] Gorman, W., Oster, S., & Kniffin, B. (2001). U.S. Patent Application No. 09/917,435. [15] Hansler, S., & Glas, B. (2011). Universal Design in Housing. Age in Action, 26(2), 1. [16] Hayes, G. R. (2011). The relationship of action research to human-computer interaction. ACM Transactions on Computer-Human Interaction (TOCHI), 18(3), 1–5. [17] Hayes, G. R. (2012). Taking action in your research. Interactions, 19(4), 50–53. [18] Heitktter, H., Hanschke, S., & Majchrzak, T. A (2013). Comparing Cross-Platform De- velopment Approaches for Mobile Applications. Web Information Systems and Technologies. Springer Berlin Heidelberg, 120–138. [19] Holzinger, A. (2005). Usability engineering methods for software developers. Communications of the ACM, 48(1), 71–74. [20] Hutchins, E. L., Hollan, J. D., & Norman, D. A.

(1985). Direct manipulation interfaces. Human Computer Interaction, 1(4), 311–338. [21] Ishizaki, K., Takeuchi, M., Kawachiya, K., Suganuma, T., Gohda, O., Inagaki, T., ... & Ogasawara, T. (2003, October). Effectiveness of cross-platform optimizations for a Java just- in-time compiler. In ACM SIGPLAN Notices, 38, 11, (pp 187–204). ACM. [22] Juntunen, A., Jalonen, E., & Luukkainen, S. (2013, January). HTML 5 in Mobile Devices– Drivers and Restraints. , 46th Hawaii International Conference on In System Sciences (HICSS), (pp. 1053–1062). IEEE. [23] King-Sears, M. (2009). Universal design for learning: Technology and pedagogy. Learning Disability Quarterly, 199–201. 58 [24] Lau, Y. K., Cassidy, T., Hacking, D., Brittain, K., Haricharan, H. J., & Heap, M. (2014). Antenatal health promotion via short message service at a Midwife Obstetrics Unit in South Africa: a mixed methods study. BMC pregnancy and childbirth, 14(1), 284. [25] Laufer, L., Halacsy, P., & Somlai-Fischer, A. (2011, May). Prezi meeting: collaboration in a zoomable canvas based environment. In CHI'11 Extended Abstracts on Human Factors in Computing Systems (pp. 749–752). ACM. [26] Looijesteijn, K. (2009). The design of a Deaf-to-hearing communication aid for South Africans. MSc thesis, Delft University of Technology, The Netherlands [27] Mace, R. (1997). What is universal design. The Center for Universal Design at North Carolina State University. Retrieved November, 19, 2004. [28] Motlhabi, M. B., Tucker, W. D., Parker, M., & Glaser, M. (2013a). Improving Usability and Correctness of a Mobile Tool to help a Deaf person with Pharmaceutical Instruction. In Proc. DEV-4 (Article 13). Cape Town; New York: ACM Press. [29] Motlhabi, M. B., Glaser, M., Parker, M., & Tucker, W. D. (2013b). SignSupport: A Limited Communication Domain Mobile Aid for a Deaf patient at the Pharmacy. In R. Volkwyn (Ed.), Southern African Telecommunication Networks & Applications Conference (pp. 173–178). Stel- lenbosch, South Africa: Pretoria: Telkom. [30] Mutemwa, M., & Tucker, W. D. (2010). A mobile Deaf-to-hearing communication aid for medical diagnosis. In D. Browne (Ed.), Proc. Southern African Telecommunication Networks & Applications Conference (pp. 379–384). Stellenbosch, South Africa; Pretoria: Telkom. [31] Myllymaki, J. (2002). Effective web data extraction with standard XML technologies. Com- puter Networks, 39 (5), 635–644. [32] Ngethe, G. G., Blake, E. H., & Glaser, M. (2015). SignSupport: A Mobile Aid for Deaf People Learning Computer Literacy Skills. [33] Nurseitov, N., Paulson, M., Reynolds, R., & Izurieta, C. (2009). Comparison of JSON and XML Data Interchange Formats: A Case Study. Caine, 9, 157–162. [34] Oancea, B., Rosca, I. G., Andrei, T., & Iacob, A. I. (2011). Evaluating Java performance for linear algebra numerical computations. Procedia Computer Science, 3, 474–478. [35] Perron, B., & Stearns, A. (2010). A review of a presentation technology: Prezi. 59 [36] Pressman, R. S. (2005). Software engineering: a practitioners approach. McGraw Hill. [37] Rabin, M. D., & Burns, M. J. (1996, April). Multimedia authoring tools. In Conference Companion on Human Factors in Computing Systems (pp. 380–381). ACM. [38] Sangappa, S., Palaniappan, K., & Tollerton, R. (2002, November). Benchmarking Java against C/C++ for interactive scientific visualization. In Proceedings of the 2002 joint ACM-ISCOPE conference on Java Grande (pp. 236–236). ACM. [39] Signer, B., & Norrie, M. C. (2007, February). PaperPoint: a paper-based presentation and in- teractive paper prototyping tool. In Proceedings of the 1st international conference on Tangible and embedded interaction (pp. 57–64). ACM. [40] Shneiderman, B. (1981, May). Direct manipulation: A step beyond programming languages. In ACM SIGSOC Bulletin (Vol. 13, No. 2-3, p. 143). ACM. [41] Tongco, M. D. C. (2007). Purposive sampling as a tool for informant selection. [42] Tucker, W. D. (2015). Beyond traditional ethics when developing assistive technology for and with Deaf people in developing regions. In M. Hersh (Ed.), Ethical Engineering for In- ternational Development and Environmental Sustainability (pp. 293–324). Springer: London. doi:10.1007/978-1-4471-6618-4 [43] World Wide Web Consortium. (2006). Extensible markup language (XML) 1.1 60 Appendix A Information sheet 1. What is this research project about? • I am going to tell you about a computer-based project, called Authoring tool • This project aims to improve communication

> 1**between a Deaf and hearing person in any limited domain** conversation **scenario.**

• You might be familiar with the mobile application called SignSupport, • The mobile application that helps a Deaf person to communicate with a pharmacist during medical dispensing. • SignSupport only support android mobile platform, South African Sign language and a pharmacy scenario only. • The Authoring tool project aims to expand SignSupport in a way that it can accom- modate multiple limited domain conversation scenarios, mobile platforms and Sign languages. 2. Who is running the project? • We are Computer Scientists from the University of the Western Cape. • You might know Bill Tucker. Hes the project leader. • The student responsible for this particular project is Lindokuhle Duma. 3. What do we want to achieve? • We want to improve SignSupport in a way that it: 61 • Accommodate multiple limited domain communication scenarios. • Support multiple signed languages for illiterate Deaf users in developing regions • Become platform independent to work on a variety of mobile devices. 4. What will we do? • We will design and build an application that runs on the computer. • The application will allow you to design mobile application for handling a

> 1**limited domain scenario between a Deaf and a hearing person, e.g. when a**

Deaf

person is visiting a pharmacist or the department of Home Affairs to get an ID book. • You

2**will have to initially prepare a** conversation **script** dialog aside **then use the** authoring **tool to design the application.**

• The

2**authoring tool will present** you **with screen templates and screen components like image buttons, text boxes, video frames** etc. • **The user**

will then be able to build the application interface using drag-n-drop features. • The video frames, in the application interface, will be used for displaying Sign language videos for conveying information to the Deaf person; and the text will be for conveying information to the hearing person. • You can also connect the screens by clicking on the button and then type the screen number to connect to, in the popup window. • The connection represents the conversation dialog and how you or other end users will interact with the application. • After you have completed designing the application interface, You will then the view the application design in a PDf document. 5. What do we expect of you? • We want Deaf people to help design and evaluate this system. • You will be asked to create simple conversation dialogs for different scenarios, using the system; and then view them in a PDF file. • Participation is of your own choice. • If you agree

23**to participate,** we **will ask you to sign a consent form.**

6. Benefits 62 • This study will

2**improve SignSupport in a way that SignSupport will accommodate** dif- ferent **limited domain communication** scenarios, **multiple signed languages and mobile platforms. • Another benefit from this study**

empowers

2**Deaf people and domain specialist to develop** SignSupport **scenarios together.**

7. Risks AND difficulties • The risk is

2**that the script dialog created from the tool might not cover all parts of the conversation within the limited domain scenario; and this might lead to a** communica- tion **breakdown between a Deaf and a hearing person.**• These **script** dialogs **that** will be **produced by the tool** will **mainly be used for the purpose of this study and** will **not** be used **in the real world**

8. Withdrawal and confidentiality • All videos recorded during the research session will be kept confidential and will be stored on a computer with password, which is known only by the responsible researcher. • Your identity will not be disclosed in public unless we receive permission from you. • Please be informed that you have the right to withdraw from any research session by informing the session leader. • As soon as you withdraw from the session, all material about your information will be destroyed. 9. Dissemination of the study results • All information will be disseminated when the study is completed in the form of several papers at various conferences. • Ultimately the rest of the results will be published in a form of a Masters Thesis. • Deaf participants will be kept informed via several presentations at DCCT at strategic times of the project life. .... Signature Date ..... Name and surname 63 Appendix B Deaf participant consent form I ..... fully understand the Authoring tool project

12**and agree to participate. I understand that I can withdraw from the study at any time,**

and any information collected pertaining my contribution will be destroyed at once. I also

29**understand that** all **information** that **I provide will be kept confidential, and**

that my identity will not be revealed in any publication resulting from the research unless I choose to give permission. I acknowledge that all in-formation attained

24**in this study** or test **will be stored on a computer that**

has a password that is only known by the researcher. Furthermore, all recorded interview media and transcripts will be destroyed after the project is completed.

16**I am** also **free to withdraw from the project at any time. I understand that**

an interpreter will be used for this trial and the information he/she translates will be kept confidential and not repeated. For further information, please do not hesitate to contact: Lindokuhle Duma & Bill Tucker Dept. of

8**Computer Science University of the Western Cape Private Bag X17 Bellville 7535 Email:** 3063344@myuwc **.ac.za**

/ wdtucker@gmail.com ..... Signature (Participant) 65 Appendix C Interpreter consent form I ..... fully understand the Authoring tool project and agree to interpret. I under- stand South African Sign Language and will provide sign language translation. I am bound by Deaf South Africa's (DEAFSA) code of ethics for SASL interpreter to adhere

4**all aspects of the Code of Ethics at all times during and after assignments; keep all assignment-related information strictly confidential and adhere to professional standards of confidentiality;** and **render the message faithfully, always conveying the content, intent and spirit of the speaker using the language most readily understood by the person(s)whom they serve.**

I also pledge that I have explained all the aspects of the research to the participants. For further information, please do not hesitate to contact: Lindokuhle Duma & Bill Tucker Dept. of

8**Computer Science University of the Western Cape Private Bag X17 Bellville 7535 Email:** 3063344@myuwc **.ac.za**

/ wdtucker@gmail.com 67 Appendix D Scenario graph to XML file The screens are made of InternalJFrames and the Canvas area is a JDesktopPane. They can be moved around, by default, by a mouse inside the canvas. The components of the screens are the components of the swing library (JLabel, JTextArea, JPanel) and are used to hold the content of the screens (text, icons and videos). The links between the screens are drawings from the AWT library. The method call repaint is executed every time a screen changes position, in order to redraw the link lines, to point to the new position of that screen. The screens are stored in an array as they are added into the canvas. This makes it easy to access them, as we use the array index to refer to them. The video area is a uses JLabels to store videos when they are imported from the computer. VLCj is a library that is used to play the videos. This library is chosen because it is capable of playing many videos formats. This library also requires that the VLC is installed on the computer which the authoring tool is installed at. The videos are dragged and dropped to the screens as images. Xuggler library capture a video frame at a specific time point and stores it as an image file, with the same name as that video. That is the image file that gets dragged and dropped on behalf of the video.

After the scenario has been created, the XML file is then generated. The screens are accessed using an array index (screen [0] screen[n]), through a forloop. Each of these five screens has their own method, in a class file, which is meant to extract the screens content, as text, and copy it to the XML file, as shown in the figure above. Each screen has an integer variable called screentype, which has a value (1 - 5); and informs the method that should be called

47**in order to extract the content of** that screen. **The**

text content like the title and the video captions are extracted by the 69 getText method. The videos and icons are represented by their file names in the XML file, so their file names get copied to a variable like videoName and iconName, during the drag-n-drop process. The possible changes of Screen type 1 4, are predictable so the content extraction methods of these screens are efficient to get all the content of these screens, regardless of the modifications made to them. Screen type 5 is unpredictable as it is unknown which swing component the author will use and what data type will be stored in those components. The algorithm used to extract content in screen type 5 is that all its components are stored in an array of type Component. This is the parent type of the Swing components so it allows the array to store different Swing components. Initially, the method needs to find the actual instance of the component in order to be informed on how to how to extract the content stored in that component. The content is then extracted from each component using the appropriate methods. screen2XML.jpg Figure D.1: Screen's output in the XML file 70 57 61 65 67 69 4 13 22 37 43 44

Appendix F

Conference paper

Below is the paper that the researcher wrote and published at South African Telecommunication and Network Applications Conference (SATNAC).

Usability of an Authoring Tool for Generalised Scenario Creation for SignSupport

Lindokuhle S. Duma[1], Prangnat Chininthorn[2], Meryl Glaser[3] and William D. Tucker[4]

[1,2,3,4]Department of Computer Science, University of the Western Cape

Robert Sobukwe Road, Bellville 7535 South Africa

Tel: +27 21 959 3010, Fax: +27 21 959 1274

[1]3063344@myuwc.ac.za
[2]pchininthorn@uwc.ac.za
[3]merylglaser@gmail.com
[4]btucker@uwc.ac.za

Abstract— **This paper presents the usability testing results for an authoring tool that generalises scenario creation for a tool called SignSupport. SignSupport is a mobile communication tool for Deaf people that currently runs on an Android smartphone. The authoring tool is computer-based software that helps a domain expert, with little or no programming skills, design and populate a limited domain conversation scenario between a Deaf person and a hearing person, e.g., when a Deaf patient collects medication at a hospital pharmacy or when a Deaf learner is taking a computer literacy course. SignSupport provides instructions to the Deaf person in signed language videos on a mobile device. The authoring tool enables the creation and population of such scenarios on a computer for subsequent 'playback' on a mobile device. The output of this authoring tool is an XML script, alongside a repository of media files that can be used to render the SignSupport mobile app on any platform. Our concern now is to iteratively develop the user interface for the authoring tool, focusing on the domain experts who create the overall flow and content for a given scenario. The current authoring tool was evaluated for usability; for both pharmacy and ICDL course scenarios with purposive sampling. The findings suggest that the authoring tool can generalise SignSupport for multiple limited domain scenarios, mobile platforms and signed languages.**

Keywords— **Mobile Apps, Software Design, XML**

I. Introduction

The authoring tool for generalised scenario creation described in the paper is a computer-based application that helps a domain expert with little or no programming skills to design a front-end interface for a communication flow between a Deaf and a hearing person. The end result is a mobile communication tool called SignSupport that helps a Deaf person with a given communication scenario; with pertinent information relayed to the Deaf user in signed language [1]. The authoring tool produces an Extensible Markup Language (XML) script as an output after the flow of a given scenario is designed by a domain expert. This XML file is consumed by an XML parser to render a mobile application on any given mobile platform [1]. The current version of SignSupport is a mobile application that helps a pharmacist to give comprehensible medical instructions to a Deaf patient during medicine dispensing in the form of pre-recorded South African Sign Language (SASL) videos [2]. These medical instructions are stored on a mobile phone's memory card for the patient to view at any time. The limitations of the SignSupport mobile app are that it only caters for the pharmacy setting scenario, and runs only on an Android mobile platform. Therefore, this authoring tool is aimed at generalising SignSupport to accommodate multiple domain scenarios; for multiple mobile platforms and that can be populated by any language for low literacy end users.

This paper uses Deaf with a capital 'D' to refer to a linguistic and cultural group of people with hearing loss who mainly use SASL as a mother tongue [3]. Deaf people often experience communication barriers while communicating with a hearing person who cannot sign. While many technologies, especially mobile, support voice and text communication, this presents usability difficulties for Deaf people with low functional text literacy. SASL interpreters are very expensive for Deaf people as they are very scarce, and Deaf people battle significantly to communicate with hearing people in the absence of an interpreter. SignSupport is an assistive technology meant to bridge such gaps for a given scenario, and the authoring tool is meant to support scenario designers.

The authoring tool was evaluated for usability with participants who were recruited through an applied purposive sampling method. This is because the scenarios were defined prior to usability testing. Participants who were specialised in these domains or have worked with Deaf people before were recruited. The results show that the authoring tool is capable of creating multiple limited domain scenarios and that it supports a domain expert with little or no programing skills.

The rest of the paper is organised as follows. Section II covers the history of SignSupport. Section III covers the related work. Section IV describes the authoring tool prototype. Section V defines the research methods that guide the research and the prototype implementation. Sections VI presents and discusses the results obtained from the prototype usability testing. Section VII concludes and outlines the next research steps and future work.

II. History of SignSupport

Since 2009, there have been several versions of SignSupport produced through an iterative research process.

1) Version 1 was an internet browser-based mock-up design [3]. The main objective of this mock-up was to help the doctor understand the symptoms of a Deaf patient so that s/he could prescribe medication for the patient. The mock-up presented a set of questions in SASL which the Deaf patient

answered before seeing the doctor. The doctor then viewed the summary of the patient's answers in English text.

2) Version 2 was for the same scenario as version one and was developed as a Symbian mobile app with XHTML [4]. This mock-up required only a mobile phone with a data connection running a browser that supported Small web format. However, the doctor and Deaf patient scenario was found to be too wide and complicated to be constructed as an application [5].

3) Version 3 was then shifted into a more constrained communication domain between a pharmacist and a Deaf patient [5]. Its aim was to help the Deaf patient understand the medical instructions as prescribed by a pharmacist. This mock-up was designed with two types of interfaces: one for the Deaf patient to input background information prompted by SASL videos and icons and one for the pharmacist to view the Deaf patient's background information and dispense medication. The Deaf interface also relayed that information in pre-recorded SASL videos.

4) Version 4 was a redesign by a multi-disciplinary and trans-university team and then implemented as an Android app [6]. The construction of this version is now in the process of usability testing in a public hospital pharmacy.

The SignSupport approach was also used to create a prototype to aid with self-paced learning for an International Computer Driving License (ICDL) course. That prototype was developed in parallel with the more recent pharmacy prototypes. Our research team sees the potential of SignSupport for additional contexts, i.e. other limited domain communication scenarios. Therefore, this paper proposes that an authoring tool for SignSupport can provide a solution to assist domain experts to design a communication flow that can meet specific needs for Deaf end users.

III. Literature Review

This section discusses the literature review that has been consulted in order to formulate this study. We used the ideas and techniques that that were found in this literature review, as references when building and evaluating the authoring tool. This ensured that the authoring tool suits the needs of the domain experts and that it provides the greatest user experience when in use.

A. Authoring tools in general

An authoring tool allows a user with little or no programming experience to design and implement complex applications, mainly web applications [7]. Authors can create these complex and attractive applications by merely clicking and defining relationships between objects, e.g., text, pictures and videos. An authoring tool also allows an author to preview a design to see how it will look when it is a complete app. Examples of well known authoring tools include Dreamweaver and Microsoft Front Page. The structure and the features of the existing authoring tools were used as a reference to inform the interface design of the SignSupport authoring tool. One typical similarity amongst authoring tools is that they have multiple distinct templates that authors can customise directly or by changing the templates' source code. Templates help authors to work faster as they are generic and can be adapted to any form within the intended use of that authoring tool.

B. Language independence

Language independent techniques involve developing applications that accommodate different human languages. This means that software assets (text, icons and videos) can be easily changed or replaced. Language independent software accommodates different data formats and conforms to software engineering principles such as software reuse [8]. The SignSupport authoring tool will have screen templates that can be modified and reused limitlessly with different data assets. Pictures and videos have different data types so this authoring tool will use generic libraries that can read most popular data formats.

C. Design for all

This method is also known as universal design, as it entails creating software that can be used by all people effectively, without the need for adaptation or specialised resources [9]. Principles of this method include creating useful designs that accommodate a wide range of individual preferences and abilities, error tolerance and low physical effort [9]. The aim is also to accommodate Deaf domain experts, with low text and functional literacy, as authoring tool users in the future. The techniques that are used to achieve this goal include direct manipulation via drag-and-drop of objects in a WYSIWYG editor [10]. The authoring tool has tooltips, help options and a user guide for novice users. This enables any author to learn to use the authoring tool quickly and also to have assistance within the authoring tool in case s/he needs it. These helper options can also be adapted to in signed language videos, as tutorials that demonstrate certain functions for Deaf authors.

IV. The Authoring Tool for SignSupport

This section describes the features available in the authoring tool. The authoring tool helps a domain expert to design an interface for a limited domain conversation flow between a Deaf and a hearing person. The software presents the domain expert with screen templates. These screen templates can be populated with text, icons and/or signed language videos (that would be provided by native signed language speakers, not the domain experts, unless they are Deaf, of course). The screens can also be linked to each other in a graph structure. The link between each screen indicates the user interaction on the real mobile app i.e. which screen will show next when the end user clicks a given button. Figure 1 presents a screenshot of the authoring tool interface while a domain expert is using it. The interface of the authoring tool is divided into four areas:

1) The button area contains four buttons, namely 'Add template', 'Clear all', 'Add video' and 'Export to XML'. When the domain expert clicks on the 'Add template' button, a window pops up and presents all the default screen templates of the authoring tool. The user then chooses the template that s/he would like to add to the canvas by clicking the checkbox on the left side of the preferred template and then clicks the 'OK' button at the bottom of the window. The selected screen template then appears on the canvas as active. The 'Clear all' button clears all the screen templates from the canvas. The 'Add video' button helps the domain expert add videos from the computer as an asset within the authoring

tool. The videos line up vertically as they are being added from the computer to the video area. The 'Export to XML' button invokes a method that reads all the screen templates, their assets and connections to each other; and then produces an XML file that includes all of this information. Since the XML file contains only text, it only stores an asset's information such as filename and relative path, in text. An XML parser on a target device consumes the XML file in order to render a platform-specific mobile app.

2) The video area contains a series of sign language videos that are added by the domain expert from the computer. The sign language videos can be recorded before or after the scenario based conversation flow is designed. Then each video is given an English tag/filename in a way to help the scenario domain expert include the correct video in the right place [11]. Videos in this area can be played (and paused) by clicking on them. They can also be dragged and dropped to other screens.

3) The icons tab area contains icons in various categories. The domain expert can also add new tabs of icons. The icons can be dragged and dropped to the screens, and linked to other screens and/or actions.

4) The Canvas area is where the selected screen templates appear when added. The domain expert populates a screen by typing in the intended message and dragging and dropping icons or a video to the indicated areas in the screen. If the domain expert drops an icon into a wrong area, s/he can right click on that area and select the 'remove icon' option and the icon will be removed. To connect the screen to another, the domain expert clicks on the icon that represents a button from one screen, confirms that s/he wants to connect that icon to another screen at a prompt dialog pop-up, then selects the screen to connect with. A blue link line will appear in the Canvas, between these two screens. This link gets updated and redrawn as the screens are moved from one position to another.

The authoring tool is built using the Java programming language, and the output of the authoring tool is an XML file. The XML file serves as a database that stores the design information in text. XML is supported by a variety of technologies in various platforms [12]. This in turn helps us to achieve platform independence for scenarios designed from the authoring tool.

V. METHODS

This research follows an action research methodology [13]. This approach is cyclic in nature and involves an iterative series of problem definition, planning, implementation, observation and reflection phases. The aim of action research is to improve or change both technological and social systems for the better [13]. We hope the SignSupport mobile app proves to be a useful communication tool for both Deaf people and pharmacists. Deaf people also feel that they need communication tools for other scenarios which they can install on their various mobile phones, hence the need for an authoring tool. The development of the authoring tool is based on an iterative prototyping and usability testing method. The first step is to iteratively implement core features of the authoring tool and then evaluate them through usability testing before adding more features. Testing our prototype for usability helps us to confirm user requirements, uncover software bugs and to accommodate new features that end users may suggest after trying out the prototype. The domain experts are the main drivers of the authoring tool implementation. Since the whole research is guided by action research methodology, it is important that they are included throughout the research process.

User-based testing was employed for the usability experiment [14]. The goal of the experiment was to evaluate how effective the authoring tool can be, in enabling a domain

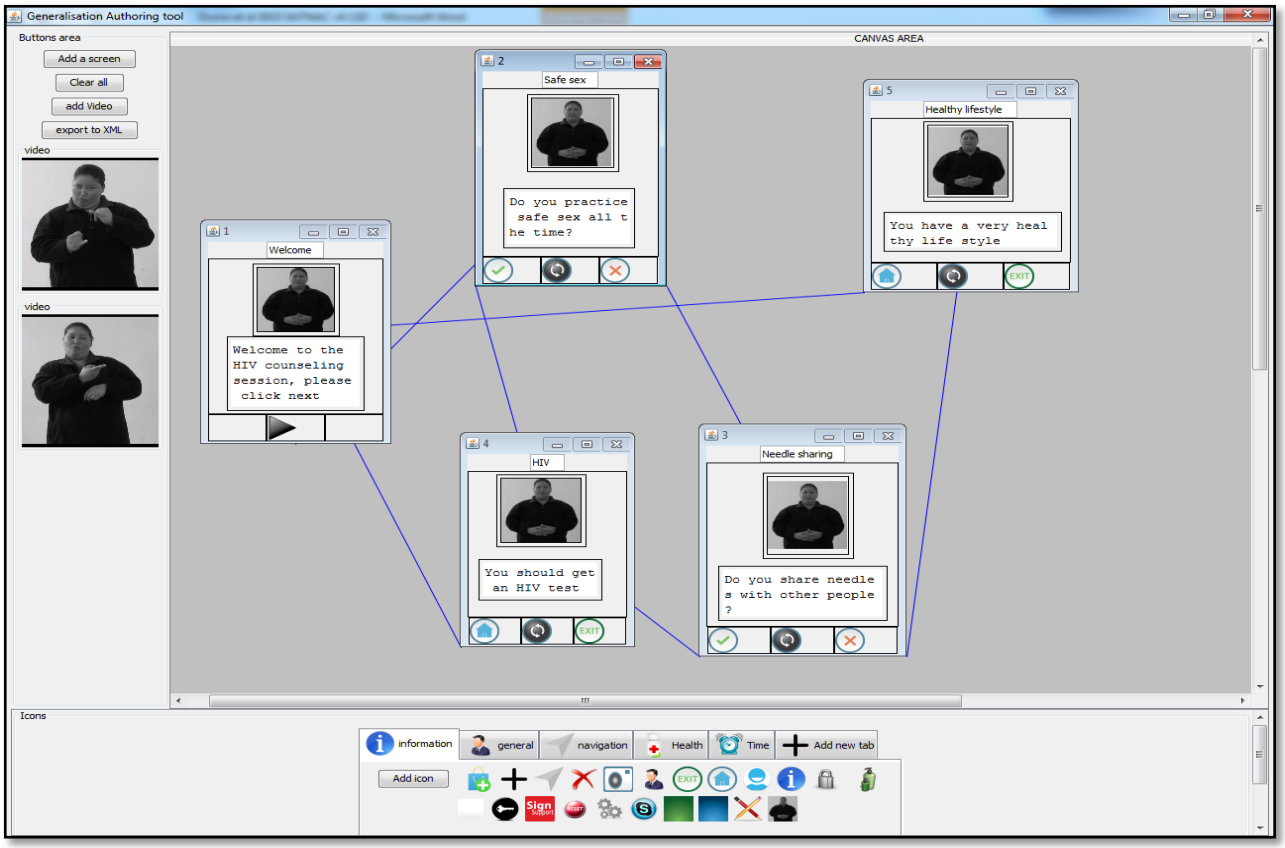

Figure 1: A screen shot of the Authoring tool's interface while a 'user' (domain expert participant) is testing it.

expert with little or no programming skills, to create a limited domain conversation scenario. The main focus is how the domain expert searches for icons from the icon tabs, adds and removes icons and videos from the screen templates, recovers from errors; reuses screen templates in the same scenario and reuses the same icons in different screen templates are also observed. The domain expert was also be encouraged to give suggestions on how the prototype can be improved to better suit their needs.

A. *Sampling*

Participants were selected through a purposive sampling method. Purposive sampling is purposefully selecting participants in terms of the qualities/skills they have [15]. The communication flow design scenarios for the experiment had already been defined, as for the medication dispensing process and the (ICDL) course. The pilot session of the authoring tool was evaluated with the aid of an industrial design engineer. Participants recruited were a pharmacist for the pharmacy setting scenario, a Deaf educator and a computer science researcher for the ICDL course scenario for the actual usability testing sessions. All participants have unique work experience with Deaf people. The industrial design engineer designed the SignSupport mobile app and the pharmacist is involved in the co-design and co-testing of the previous and current versions of the SignSupport mobile app with Deaf users. The Deaf educator gives ICDL lessons in SASL every Thursday at a local Deaf community; and the Computer science researcher works on another authoring tool for creating ICDL lessons only. All our participants are the potential users of this authoring tool and hence ideal candidates for the experiment.

B. *Testing procedure*

The testing was divided into three stages, namely training, testing and gaining user feedback. One participant attended the usability test at a time. During the training stage, the participant was introduced to the authoring tool prototype and all the features were demonstrated. The researcher then encouraged the participant to try out the features on their own e.g. add screens to the canvas area, drag and drop icons and videos to the indicated area, and link icon labels with other screens. Afterwards a training exercise was given to the participant to design an HIV counselling scenario. This applied to all participants. This scenario was given to the participants as a decision flow graph diagram, as shown in Fig 2.

During the testing stage, participants were first asked to design and populate a limited scenario that was specific to their domain. The participants from the pharmacy scenario were asked to design a few screens from the familiar SignSupport mobile app for medicine dispensing. The participants from the ICDL course scenario were asked to design a specific lesson from the ICDL course book, which can be populated with SASL video instructions. The participants were assigned the tasks specific to their domain to give them a feeling of how they would use the authoring tool to best suit their needs. Each participant was asked to speak out about everything they see, think, and act while using the authoring tool. All the 'think-out-loud' [13] messages were accounted as part of their feedback. This process was then followed for the HIV scenario for all of the participants.

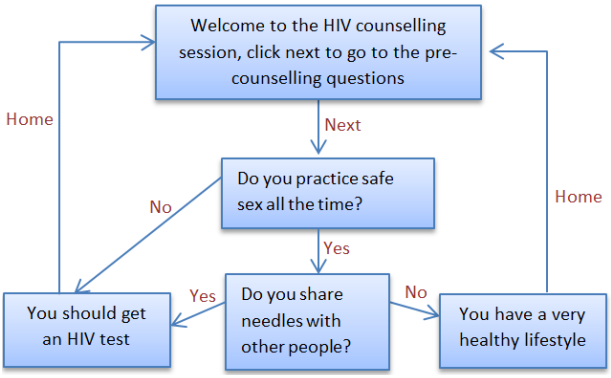

Figure 2. HIV pre-counselling scenario used for the training stage for all participants

During the feedback stage, additional questions regarding the available features and the use of the authoring tool were asked to the individual participants for additional clarification.

C. *Data collection*

Qualitative methods were used to collect data. The usability session was video recorded. The video recorder was directed to the computer screen to record how the participant interacted with the authoring tool. Participants were asked to use the think aloud strategy to voice out what they were doing and all the thoughts that came to their mind as a result of using the tool. After the testing stage, the researcher conducted a semi structured interview with each participant. The main questions asked in the interview were:

- What did you like/dislike about the authoring tool?
- What did you find easy to use when building the scenarios?
- What did you find confusing or difficult?
- Any features you would like to be changed or added to the authoring tool?

D. *Data analysis*

Data collected from the trial was summarised and organised according to the four areas of the authoring tool interface, which are stated and discussed in the section below.

VI. Results and Discussion

All participants finished the training scenario successfully and with no difficulties. They all repeatedly used one screen template with the same icons for the training scenario design. All of them also managed to create and populate the testing scenario design. The participants' feedback from both the pilot and the actual usability test session is described in accordance with the aforementioned four areas of the authoring tool interface features.

A. *Buttons area*

Here is the description of feedback from all the participants regarding the available buttons in the button area. All participants understood the use of each button on the interface.

1) 'Add screen': All participants praised the interface and notification designed for the 'add screen' and its process. The selecting process was easy to understand, the background colour of the selected screen template changed as it was clicked, and this assured the participants that the template was indeed selected.

2) The canvas was cleared immediately after clicking the "Clear all" button; and the XML output was created and saved to a file after clicking the "Export to XML" button. Two participants were also interested in seeing what the XML output looked like.

3) Suggestion on the additional button: Two participants said that there should be an 'Add picture' button that they can use to add pictures to the screens, which works similarly to the 'Add video' for adding videos to the screen.

B. Video Area

This section describes feedback obtained about the video area.

1) Adding video to the video area: All participants found the video addition process to the video area easy to do.

2) Suggestion of a scrollbar: One participant suggested that there should be a scrollbar in the video area, so that multiple videos which will be added to the video area can be seen.

3) Suggestion of video controls: One participant suggested that there should be 'video control buttons' that show on each video in the video area, when the mouse is hovered over them, so that he could click on these video controls to play each video.

C. Icon tabs area

All participants could navigate through the icon tabs easily. Names of all the icons appeared when the participants moved the mouse over them. Two participants commented that the tooltip assured them that they were choosing the icon with the correct semantic meaning to drag to the screens. Two participants struggled to find the 'home' and the 'exit' icons as they expected to find these icons under the navigation icon tab instead.

D. Canvas Area

This section presents the feedback regarding the screen's components when the screens were already added to the canvas area.

1) Adjustment of the screen and its component sizes: All participants disliked that the components of the screen remained in the same size and position when the screen was resized. They then had to resize and position each of them one by one. So, two participants suggested that the whole screen and its components should be proportionally resized all at once.

2) The design of the default screen templates: One participant found the default screen templates limited in terms of design. Most of the component areas were fixed in their position, which did not allow the user to modify the default design. One participant mentioned placing the 'next' and 'back' buttons on the sides of the screen, whereby the default screen template did not match this need. Therefore she suggested having a blank screen template available for users to customize to their own screen design(s).

3) Screen links: All participants found the screen linking option easy to understand and use; however, they preferred that the linking process be done with fewer mouse clicks. The linking line should go around the screens which lie between the two indicated screens instead of going underneath them. With the suggested linking line, it would be easier for the user to trace back and recheck each link. In addition, one participant suggested that the line indicating the link should have a message that indicates its origin and the destination of linking.

4) Drag and drop option: All participants said that they liked the drag and drop options for adding icons and videos to the screen. It was observed that all participants missed the icon drop area several times when they were populating the screens with icons. However they tried again even though the authoring tool did not give them any warning/notification

The table below lists features that the participants suggested the authoring tool should have. Participants made these suggestions during the feedback session, when they were asked about what they would add to the authoring tool. These features will be added to the authoring tool and will also be evaluated for usability in the next iteration.

TABLE 1

SUGGESTIONS OF ADDITIONAL FEATURES OBTAINED FROM THE PARTICIPANTS

Feature(s)	**Motivation**
Zoom in and out	Helps authors to their whole scenario design outline without having to scroll up or down. This feature will also enable the author to focus on one screen or part of the screen at a time
Copy and paste	To duplicate screens that are already populated; and modify them for a similar purpose so that there won't be a need to start over all the time.
User guide and help function	To enable the author to learn to use the authoring tool independently and to get assistance in the absence of the researcher.
Tooltip text on buttons and mouse-hovers	Tooltip text on a button or mouse-hover option reminds the author what each button or option is used for.
Scenario design simulation	This gives authors an idea of how the scenario would look like as a real app and also to confirm that the links have been defined correctly.
Undo and redo	This option helps to recover from errors or to repeat recent events.

Despite all the recommendations for improvement, all participants felt that the authoring tool could assist and empower their capacity greatly to create a communication flow to use with a Deaf counterpart. All participants praised the authoring tool, saying that it was easy to use and it saved

a lot of time since most of the usability session tasks were completed within few minutes, by just clicking and dragging items to the screens in the canvas. They also mentioned that they would like to use it in the future.

VII. CONCLUSION AND FUTURE WORK

The results indicate that there are several additional features required from the authoring tool to improve its usability. The robustness of the authoring tool needs to be thoroughly evaluated before the taking it to its potential users for usability testing of the next iteration. The creation of the three limited domain scenarios (HIV pre-counselling, ICDL course lessons and rebuilding a thread of the SignSupport pharmacy app) demonstrates that the authoring tool can generalise SignSupport scenario creation for limited domain communication between Deaf and hearing people. The reuse of screen templates and data assets (icons and videos), combined with the production of an XML file representing a given scenario, appear to indicate that the authoring tool can generalise the SignSupport mobile app for multiple signed languages and mobile platforms.

The next step is to improve the authoring tool by fixing its errors and adding features recommended by the participants. Another usability testing exercise will then be conducted with more participants and more limited domain scenarios. Thereafter, one complete limited domain scenario will be built using the authoring tool, with recorded SASL videos with the help of a Deaf person and a SASL interpreter. Then that scenario can be evaluated with Deaf participants. That requires a fully functioning XML parser. The XML parser will enable the authoring tool to produce a mobile app that can be run on a mobile phone. Note that signed language videos will have to be recorded separately, then integrated into the scenario. Another possible future effort would be that the authoring tool connects to a webcam. The webcam could be used to record sign language videos while a scenario is being designed. A Deaf person or any person that can sign fluently could do this. The recorded sign language videos could also be edited within the authoring tool and then added to the scenario being created. Once the scenario is complete, it can be installed to a mobile phone, together with the recorded signed language videos, and be made available for use immediately.

ACKNOWLEDGMENT

We thank the Deaf Community of Cape Town for their involvement. Thanks also to George Ng'ethe, Marshalan Reddy, and Edwin Blake at UCT for their collaboration on this project. We also thank Telkom, Cisco, Aria Technologies and the THRIP (Technology and Human Resources for Industry Partnership) initiative of the South African Department of Trade and Industry for financial support via the Telkom Centre of Excellence (CoE) programme. THRIP funding (project TP13072623839) is managed by the National Research Foundation (NRF). Any opinion, findings and conclusions or recommendations expressed in this material are those of the authors and therefore the NRF/THRIP does not accept any liability in regard thereto.

REFERENCES

[1] Blake, E., Tucker, W., & Glaser, M. (2014). Towards communication and information access for Deaf people. SACJ, (54), 10–19

[2] Motlhabi, M. B., Glaser, M., Parker, M., & Tucker, W. D. (2013). SignSupport: A Limited Communication Domain Mobile Aid for a Deaf patient at the Pharmacy. In R. Volkwyn (Ed.), Proc. *SATNAC* (pp. 173–178). Stellenbosch, South Africa; Pretoria: Telkom.

[3] Looijesteijn, K. (2009). The design of a Deaf-to-hearing communication aid for South Africans. MSc thesis, Delft University of Technology, The Netherlands.

[4] Mutemwa, M., & Tucker, W. D. (2010). A mobile Deaf-to-hearing communication aid for medical diagnosis. In D. Browne (Ed.), Proc. *SATNAC* (pp. 379–384). Stellenbosch, South Africa; Pretoria: Telkom.

[5] Chininthorn, P., Glaser, M., Freudenthal, A., & Tucker, W. D. (2012). Mobile Communication Tools for a South African Deaf Patient in a Pharmacy Context. In P. Cunningham & M. Cunningham (Eds.), Proc. *ST-Africa*. Dar es Salaam, Tanzania; Dublin: IIMC International Information Management Corporation.

[6] Motlhabi, M. B., Glaser, M., Parker, M., & Tucker, W. D. (2013). SignSupport: A Limited Communication Domain Mobile Aid for a Deaf patient at the Pharmacy. In R. Volkwyn (Ed.), Proc. *SATNAC* (pp. 173–178). Stellenbosch, South Africa; Pretoria: Telkom .

[7] Gorman, W., Kniffin, B., & Oster, S. (2001) Web page authoring tool. *U.S. Patent Application 09/917,435*.

[8] Pressman, R. S. (2005). Software engineering: a practitioner's approach. McGraw Hill, 306-307.

[9] Mace, R. (1997). What is universal design. The Center for Universal Design *at North Carolina State University*. Retrieved Retrieved November, 19, 2004.

[10] Shneiderman, B. (2003) Designing the user interface. Pearson Education.

[11] Motlhabi, M. B., Tucker, W. D., Parker, M. B., & Glaser, M. (2013, December). Improving usability and correctness of a mobile tool to help a deaf person with pharmaceutical instruction. In Proc. DEV (Article 13). ACM.

[12] Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E., & Yergeau, F. (1998). Extensible markup language (XML). *World Wide Web Consortium Recommendation REC-xml-19980210. http://www. w3. org/TR/1998/REC-xml-19980210*, 16.

[13] Hayes, G. R. (2012). Taking action in your research. *Interactions*, 19(4), 50–53.

[14] Jaspers, M. W. (2009). A comparison of usability methods for testing interactive health technologies: methodological aspects and empirical evidence. *International journal of medical informatics*, *78*(5), 340-353.

[15] Tongco, M. D. C. (2007). Purposive sampling as a tool for informant selection. *A Journal of Plants, People, and Applied Research, Vol. 5, pp. 147-158.*

Lindokuhle S. Duma received a BSc Honours in Computer Science in 2013 from the University of the Western Cape (UWC) and is presently studying towards an MSc with Bridging Applications and Network Group (BANG) at the same institution. His research interests include Deaf communication tools, authoring tools and cross-platform solutions. Duma is a Telkom bursar.

Prangnat Chininthorn received her Master's in 2011. She carries on her work to improve communication problems between Deaf and hearing people in health context through her PhD studies in the collaboration of Delft University of Technology and University of the Western Cape.

Meryl Glaser has an MSc in Human Communication specializing in Deaf People from the City University in London. Her research interests include Deaf education, communication, SASL and Deaf adult literacy, both text and ICT.

William D. Tucker is an Associate Prof. in Computer Science at UWC and leads the BANG research team there. His main research interests include community-based ICT4D projects and network technologies.