

UNIVERSITY OF THE WESTERN CAPE

Video Annotation Wiki for South African Sign Language



A thesis submitted in fulfillment for the
degree of Master of Science

in the
Faculty of Science
Department of Computer Science

February 2011

Declaration of Authorship

I, Jameel Adam, declare that this thesis titled, ‘ Video Annotation Wiki for South African Sign Language ’ and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly 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:

“Golden apples are just apples.”

My Dad



UNIVERSITY OF THE WESTERN CAPE

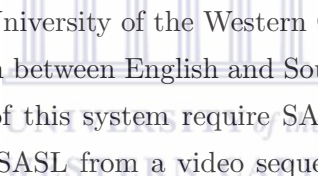
Abstract

Faculty of Science

Department of Computer Science

Master of Science

by Jameel Adam



The SASL project at the University of the Western Cape aims at developing a fully automated translation system between English and South African Sign Language (SASL). Three important aspects of this system require SASL documentation and knowledge. These are: recognition of SASL from a video sequence, linguistic translation between SASL and English and the rendering of SASL. Unfortunately, SASL documentation is a scarce resource and no official or complete documentation exists. This research focuses on creating an online collaborative video annotation knowledge management system for SASL where various members of the community can upload SASL videos to and annotate them in any of the sign language notation systems, SignWriting, HamNoSys and/or Stokoe. As such, knowledge about SASL structure is pooled into a central and freely accessible knowledge base that can be used as required. The usability and performance of the system were evaluated. The usability of the system was graded by users on a rating scale from one to five for a specific set of tasks. The system was found to have an overall usability of 3.1, slightly better than average. The performance evaluation included load and stress tests which measured the system response time for a number of users for a specific set of tasks. It was found that the system is stable and can scale up to cater for an increasing user base by improving the underlying hardware.

Acknowledgements


The author wishes to firstly acknowledge the SASL project supervisor Mr James Connan for his advice, guidance, suggestions and financial support. Sir your wisdom and patience is truely admirable.

I also would like to thank my family, Soraya, Yusuf, Saliem and especially my beloved mother Mariam for supporting and encouraging me to pursue this degree. Without their support it would have been impossible to complete this thesis.

Lastly I would like to thank all my friends, especially Mehrdad for his assistance and guidance during the compilation of this thesis.



Contents

Declaration of Authorship	i
Abstract	iii
Acknowledgements	iv
List of Figures	viii
List of Tables	xi
Abbreviations	xii
 <p>UNIVERSITY of the WESTERN CAPE</p>	
1 Introduction	1
1.1 Background	1
1.2 Research Objectives	3
1.3 Research Question	4
1.4 Thesis Outline	5
2 Sign Language Notation Systems	6
2.1 Sign Language	6
2.1.1 Definition of Language	6
2.1.2 Background and History of Sign Language	7
2.1.3 Sign Language Misconceptions	7
2.1.4 Overview of Sign Language Structure	8
2.2 Sign Language Notation Systems	9
2.2.1 Stokoe Notation (SN)	9
2.2.2 The Hamburg Notation System (HamNoSys)	10
2.2.3 Sutton Sign Writing System (SSW)	12
2.3 Comparison Between Sign Language Notation Systems' Parameters	14
2.3.1 Movement	14
2.3.2 Hand Shape	15
2.3.3 Location	16
2.3.4 Orientation	17
2.3.5 Non-Manual Grammatical Signals (NMGS)	18
2.3.6 Result of Comparison	21

2.4	Conclusion	22
3	Video Annotation Systems	23
3.1	Video Annotation	24
3.1.1	Structure of a Video Annotation System	24
3.1.2	Requirements of a Video Annotation System	26
3.1.2.1	Coding the Data	26
3.1.2.2	Analyzing and Interpreting the Data	26
3.1.2.3	User Interface and Device Control	27
3.1.2.4	Displaying the Data	27
3.2	Video Annotation Systems	28
3.2.1	Anvil	28
3.2.2	E-Lan	30
3.2.3	SignStream	33
3.2.4	Vannotea	35
3.2.5	Summary	39
3.3	Conclusion	40
4	Knowledge Management	42
4.1	Knowledge Management Systems (KMSs)	43
4.1.1	Groupware Systems (GS)	43
4.1.2	Expert Systems (ES)	44
4.1.3	Document Management Systems (DMS)	44
4.1.4	Decision Support Systems (DSS)	44
4.1.5	Semantic Network Systems (SNS)	45
4.1.6	Database Management Systems (DBMS)	45
4.2	Groupware Systems	45
4.2.1	Wiki Technology	46
4.2.2	DSpace	47
4.2.3	Cyn.in	48
4.2.4	Chisimba	49
4.2.5	Drupal	50
4.2.6	Summary of Section	51
4.3	Chisimba Wiki	52
4.3.1	Wiki Sub-Systems	52
4.3.1.1	Category Sub-System	54
4.3.1.2	Parser Function Sub-System	55
4.3.1.3	Flagged Revisions Sub-System	55
4.3.2	Chisimba Architecture	55
4.3.2.1	The MVC Paradigm	55
4.3.2.2	The Chisimba Component Structure	57
4.4	Conclusion	60
5	Design and Implementation	61
5.1	Summary and Formulation of Implementation Specifications	61
5.2	Methodology	62
5.3	Development of the System Functionality	63

5.3.1	Configuring the Chisimba Wiki for Video Support	63
5.3.2	Implementation of Video Annotation Support	66
5.3.3	Implementation of Support for Annotation in the Sign Language Notation Systems	69
5.3.3.1	Adding SLNS Subtitles	69
5.3.3.2	Retrieving and Playing Back SLNS Subtitles	72
5.4	The User Interface	73
5.4.1	The Initial User Interface	73
5.4.2	Refinement of the User Interface	76
5.5	Conclusion	81
6	System Testing and Analysis	82
6.1	Usability Testing	82
6.2	Performance Testing	87
6.2.1	Load Testing	88
6.2.2	Stress Testing	91
6.3	Conclusion	93
7	Conclusion	94
7.1	User Interface Refinement	95
7.2	Mobile Support	95
7.3	Concluding Remarks	95
A	Appendix	96
A.1	Performance Testing	96
	Bibliography	102

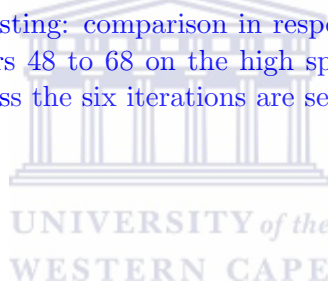


List of Figures

2.1	Stokoe Notation: Tab, Dez and Sig [66].	10
2.2	Stokoe Notation: Tab, (Tabula) refers to the sign location [55].	10
2.3	Stokoe Notation: Dez, (Designator) refers to handshape and orientation [55].	11
2.4	Stokoe Notation: Sig, (Signification) refers to motion and action, this figure represents the Sig movement symbols [55].	11
2.5	Stokoe's Notation: Representation of a written passage from the childrens story "Goldilocks and the three bears" read from left to right [64].	12
2.6	HamNoSys: Using the sign language parameters: handshape, hand position, location, and movement modifiers [65].	12
2.7	HamNoSys: NMGS representation in HamNoSys [65].	12
2.8	HamNoSys: Iconic symbols for various locations and shapes of the human body [58].	13
2.9	Hamburg Notation: Representation of the words "Ball", "House" and "Box" [65].	13
2.10	Sutton SignWriting : The word "hello" depicted with SSW [45].	13
2.11	Sutton SignWriting : A question written in SSW [45].	14
2.12	Sutton SignWriting: The basic handshape schematics [45].	16
2.13	SN and SSW: The ASL representation of the word "Coffee". SN uses a diacritic, where as SSW uses physical arrangement for descriptive representation [45].	17
2.14	Stokoe Notation: The Original ASL(SN) depicted on the left and the BSL adaptation with two orientation parameters [45].	19
2.15	Sutton Sign Writer: Orientation of the hands with shading and gap between finger and palm to indicate if the hands are parallel to wall or floor [45].	19
2.16	Sutton Sign Writer: Rotation of hands parallel to wall points "up and down" and hands parallel to floor points "forward and backwards" [45].	20
2.17	Sutton Sign Writer: Denoting the use of pronouns in Sign Language [45].	20
2.18	Sutton Sign Writer: Denoting the use of Non Manual Gesture Signalling (Facial Expressions) [45].	21
3.1	The two elements of video annotation namely: video content and meta-data	25
3.2	A simple annotation meta-data file with annotations and temporal alignments	25
3.3	Anvil : Video Annotation Interface and Controls	30
3.4	E-Lan: Video Annotation Interface and Controls [31].	32
3.5	SignStream: Video annotation interface content player [21].	34
3.6	SignStream: Video annotation interface content player [20].	35

3.7	Vannotea: The client-side user interface of the video content and annotation panes of the Vannotea system [61].	38
3.8	Vannotea: The system design and architecture of the components related to the function as the complete Vannotea system [61].	39
4.1	Wikipedia: The user interface of the Wikipedia application [73].	46
4.2	DSpace: The web-based client user interface of the DSpace application [75]	47
4.3	Cyn.in: The web-based client user interface of the Cyn.in application [12].	48
4.4	Chisimba: The user interface of Chisimba	50
4.5	Drupal: The web-based client user interface of the Drupal application [18]	51
4.6	Encapsulation of the three domain layers and processes.	57
4.7	MVC: Using the MVC platform and helper modules	58
4.8	Software Architecture Pattern: Chisimba Model View Controller (MVC). .	59
4.9	The architecture of the Chisimba Wiki system	59
5.1	JWplayer: The user interface of the JW FLV player [72].	64
5.2	JWplayer: Chisimba Wiki page embedded with the JW FLV player. . . .	64
5.3	Chisimba framework: The user interface of the file manager module. . . .	65
5.4	Modified Wiki main page with file manager link.	65
5.5	Button: The "Browse button" is used to select a file using the file manager.	66
5.6	Chisimba framework: The Select file link used in the user interface of the file management module.	67
5.7	Controls for annotating a subtitle.	67
5.8	Table relationship between video content page and the subtitles.	67
5.9	JW FLV player: The user interface of the video player with the additional subtitle pane area.	69
5.10	Subtitles list: Displayed in a tabular form containing links and information of annotated subtitles.	70
5.11	SLNS subtitle addition: (a) Popup for adding Stokoe; (b) Popup for adding HamNoSys.	70
5.12	SWIS: The user interface with canvas and glyphs.	71
5.13	Flow diagram: Interaction process between JW FLV player, Dojo and SWIS.	72
5.14	The home page of the system and instructions depicted to enter the website.	74
5.15	The main page containing the "browse" button used to upload the video files.	75
5.16	The Wiki content page used to insert page content and descriptions. . . .	75
5.17	An example of a selected uploaded video file before the user saves the information.	76
5.18	Prototyping Model: The initial user interface of the system.	76
5.19	Prototyping Model: This model is used to refine the user interface of the system.	77
5.20	Prototyping Model: The second iteration of the quick upload area from the main page of the user interface.	77
5.21	The final refined Wiki video content page user interface.	79
5.22	The final refined Wiki main page user interface showing only the refined quick upload area.	79
5.23	The final refined Wiki content description page user interface.	80

6.1	The average usability score of each sub-task across all 10 users.	85
6.2	Results of load testing: response time for users 1 to 26 on the lower specification machine.	89
6.3	Results of load testing: response time for users 1 to 41 on the higher specification machine.	90
6.4	Results of load testing: response time for users 42 to 68 on the higher specification machine.	90
6.5	Results of stress testing: response time at varied number of concurrent users on the higher specification machine. The system is below the stress capacity at 46 users but exceeds it at 47 users.	92
A.1	Results of load testing: comparison in response times across the six test iterations for users 1 to 23 on the lower specification machine. The response times across the six iterations are seen to be very similar.	99
A.2	Results of load testing: comparison in response times across the six test iterations for users 1 to 23 on the high specification machine. The response times across the six iterations are seen to be very similar.	100
A.3	Results of load testing: comparison in response times across the six test iterations for users 24 to 47 on the high specification machine. The response times across the six iterations are seen to be very similar.	100
A.4	Results of load testing: comparison in response times across the six test iterations for users 48 to 68 on the high specification machine. The response times across the six iterations are seen to be very similar.	101



List of Tables

3.1	Chapter summary of video annotation system evaluation	41
4.1	Summary of Groupware KMS evaluation.	53
4.2	Components of the MVC model: Model, View and Controller [33].	56
5.1	Fields in the Subtitles table and their descriptions.	68
5.2	Key issues highlighted in the user interface of the video content page in the first iteration and resolution of these issues.	78
5.3	Key issues highlighted in the user interface of the main page in the first iteration and resolution of these issues.	78
5.4	Key issues highlighted in the user interface of the Wiki content description page in the first iteration and resolution of these issues.	78
5.5	Key issues highlighted in the second iteration and resolution of these issues.	78
6.1	The profile of the test subjects of the usability evaluation.	83
6.2	The score scale used to measure the ease of use of the system's interface.	84
6.3	The sub-tasks used for evaluating the system.	84
6.4	Results of the usability evaluation of the system for all ten users.	86
6.5	Results of the stress testing of the system.	92
A.1	Results of the load tests iterations for the higher specification machine	96
A.2	Results of the load tests iterations for the lower specification machine	98

Abbreviations

ASL	American Sign Language
BSL	British Sign Language
CMS	Content Management System
CSS	Cascaded Style Sheets
DBMS	Data Base Management System
DMS	Document Management System
DSS	Decision Support System
ES	Expert System
FSL	French Sign Language
FLV	Flash Live Video
GS	Groupware Systems
HamNoSys	Hamburg Notation System
HMM	Hidden Markov Model
HTML	Hyper Text Markup Language
ISL	Irish Sign Language
JW FLV	Jeroen Wijering Flash Live Video
KB	Knowledge Base
KM	Knowledge Management
KMS	Knowledge Management System
NN	Neural Network
MVC	Model View Controller
NID	National Institute for the Deaf
NMGS	Non Manual Gesture Signalling
SASL	South African Sign Language
SL	Sign Language

SLNS	S ign L anguage N otation S ystem
SN	S tokoe N otation
SNS	S emantic N etwork S ystem
SSW	S utton S ign W riting
SWIS	S ign W riting I mage S erver
SVM	S upport V ector M achine
TT	T imed T ext
VAWSASL	V ideo A nnotation W iki S outh A frican S ign L anguage
WDF	W orld D eaf F ederation
WYSIWYG	W hat Y ou S ee I s W hat Y ou G et
XML	e Xtensible M arkup L anguage



Chapter 1

Introduction

1.1 Background

According to a census conducted by the National Institute for the Deaf (NID) in 2002 [15], there were approximately 0.8% deaf people and 2.4% hard of hearing people in South Africa of which the total population size is estimated to be 50 million people. [24]. World wide, these figures are even higher, with some organizations such as World Federation of the Deaf (WDF) estimating figures as high as 650 million disabled people, which consists of a percentage of deaf people estimated to 360 million. This is according to a census conducted in 1996 by [14] and the survey conducted by WDF [1].

Most deaf and hard of hearing people in South Africa communicate using South African Sign Language (SASL) as their first and only language. Contrary to popular belief, there is no universal sign language. Each country has its own distinct sign language orthography such as French Sign Language (FSL), Irish Sign Language (ISL) and SASL. Additionally, it is erroneously believed that sign languages are signed representations of spoken languages. This is not the case. Sign languages have been proven to be fully fledged languages that are distinct from spoken languages [64]. Thus, SASL is a language consisting of phonological, morphological and syntactical properties that are distinct from English and all other official spoken languages in South Africa. It is a language of its own.

The implication of this is that deaf people are not necessarily literate in spoken languages. Research has shown that 30 percent of deaf adults have poor literacy skills in spoken languages [1, 24]. While it has been shown that it is possible for a deaf person to learn to read and write in a spoken language orthography, the deaf individual would not have the same proficiency of a hearing person who has been taught in the same spoken language

orthography. Therefore, deaf people suffer from a disability in communicating in spoken languages.

There is an acute lack of educators proficient in SASL. As such, there are few educational resources for deaf people whose first and only language is SASL [24, 34]. This leaves deaf people with limited professional and communication skills that are required to get access to employment opportunities. The majority of deaf adults in South Africa are unemployed [24]. Most deaf people reside on a low income level resulting in poverty [34].

Another setback that deaf people suffer from is difficulty in carrying out everyday tasks. For example it may be difficult for a SASL speaking deaf person to buy groceries, use public transport and interact with hearing people who are not fluent in SASL in general.

In order to bridge the communication barrier, interpreters have been used to facilitate communication between deaf people and hearing people. Interpreters are skilled hearing individuals who are fluent in both SASL and spoken languages and extensive training is required to attain this ability. The interpretation process is a demanding activity and requires the interpreter to simultaneously and attentively listen, sign and speak [16]. It is no surprise, therefore, that the services of these interpreters are expensive and are not affordable to most deaf people [16, 24, 34]. The employment of a SASL interpreter could cost on average 500 South African Rand (ZAR) for below four hours of service [16]. This cost increases steeply as the number of hours increases and is, for example, as high as 4,000 (ZAR) for four hours of service.

The *Integration of Signed and Verbal Communication: South African Sign Language Recognition and Animation* project (henceforth referred to as the SASL project) at the University of the Western Cape aims to develop a fully automated translation system between English and South African Sign Language (SASL). This system will supplement the services of SASL interpreters at a greatly reduced cost. It will also eliminate human factors such as fatigue arising from the demanding process of interpretation as well as cater for privacy by eliminating the use of the interpreter.

The system will be able to capture English audio, carry out speech recognition to convert it into text and translate the text into a sign language notation system which is used to animate a 3D Avatar to perform SASL. Conversely, the system will be able to capture SASL video, process the video to extract relevant SASL features in the form of a sign language notation system, translate the sign language into English text which is then rendered as English audio using a speech synthesizer.

Several components in this system require extensive information of SASL in order to function. The gesture recognition systems use Artificial Intelligence to recognize gestures. Training such systems requires example SASL input which is not readily available.

Translating between SASL and English requires in-depth information on the structure of both languages. Although such information is available for English, this is not the case for SASL. Very limited documentation exists on this language, most of which is proprietary and costly [16]. Producing animations of SASL signs also requires example SASL videos from which to work.

This research aims at creating an online SASL knowledge base. This system will enable users to upload SASL videos and annotate them with the appropriate sign language commentary in a collaborative manner. These SASL videos can be used as input to the SASL gesture recognition systems, as templates to animate SASL signs as well as to build knowledge on SASL to be able to translate between SASL and English.

The greatest advantage of basing the system online is the use of collaboration and user generated content. The knowledge base of this system allows users to collaborate, discuss and provide commentary. This system further ensures that it is accessible across the Internet and users in remote locations can easily access the system.

Online collaboration and user generated content systems have become very popular [6, 53]. A prominent example is Wikipedia. A significant advantage of online-based systems is the provision of enhanced accessibility which, in turn, enhances collaboration. Users in various locations and of varied backgrounds are able to contribute content as well as discuss and provide commentary on the content. Eventually, an expansive sign language knowledge base would develop.

1.2 Research Objectives

Based on the motivation provided in the previous section, the aim of this research is to provide the SASL project with an annotation system that meets the following requirements:

1. provides video annotation functionality.
2. caters for sign language annotation of videos.
3. is collaborative.
4. is online.
5. provides a collaborative knowledge management system for the knowledge collected.

6. provides an acceptable level of usability from the users' perspective and performance.

The usability of the system will be evaluated by users using a rating system. The performance of the system will be evaluated in terms of the stress and load endurance of the system.

In order to obtain a system with the necessary requirements mentioned, the following tasks need to be achieved:

- Determine which sign language notation system is most advantageous for use with video annotation.
- Acquire an annotation system that satisfies the requirements enumerated. This objective can be broken into the following sub-objectives.
 - Survey the annotation systems that currently exist.
 - If such systems exist, determine whether or not these annotation systems meet the requirements enumerated.
 - If no such systems exist or they do not meet the requirements, implement an annotation system with the requirements enumerated.
- Evaluate the annotation system in terms of performance and usability.

WESTERN CAPE

1.3 Research Question

Can an online collaborative sign language video annotation knowledge management system be developed that has an acceptable usability and performance level?

This broad research question has been broken down into sub-questions for the purpose of the investigation of this research.

- Which sign language notation system is most advantageous for use in video annotation?
- Can an annotation system that meets the requirements be acquired? This question can be broken down into the following sub-questions:
 - What annotation systems currently exist?
 - Do these annotation systems meet the requirements enumerated?
 - If no suitable systems exist, can we implement an annotation system meeting the requirements?

- Does the system developed achieve acceptable usability and performance levels?

1.4 Thesis Outline

The remainder of this thesis is organized as follows:

Chapter 2 – *Spoken and Sign Language* – discusses the three most popular sign language notation systems – Sutton SignWriting, HamNoSys and Stokoe – that can be used to annotate sign language video and attempts to select the most advantageous notation for video annotation in this research. It puts these sign language notation systems into context by discussing their sign language origins and background. It also provides a discussion of the structure of each notation system and performs a comparison between these systems in terms of their inherent advantages and disadvantages. This answers research question number 1.

Chapter 3 – *Annotation Systems* – is a survey of existing sign language video annotation systems and carries out an evaluation of their functionalities and features with respect to the system required by this research. The advantages and disadvantages of each system are enumerated. This survey queried whether or not an existing system could be extended or a new annotation system was to be developed.

Chapter 4 – *Knowledge Management Systems* – discusses the various types of knowledge management systems that can host a sign language video annotation tool. The advantages and disadvantages of these systems are enumerated and are used to select a suitable knowledge management system for use in this research.

Chapter 5 – *Design and Implementation* – details the design and implementation of the online sign language video annotation knowledge management system developed. It provides a detailed description of the various components and their integration into one unified system.

Chapter 6 – *System Testing and Analysis* – describes the experimental design used to evaluate the research question put forth. It details the methods used to test the system. It also provides the results obtained, an analysis of those results and the implications of the results on the research question being tested.

Chapter 7 – *Conclusion* – concludes based on the analysis of the results obtained from testing the system. Additionally, it provides directions for future work.

Chapter 2

Sign Language Notation Systems

In this chapter the three most popular sign language notation systems is discussed namely, Stokoe, HamNoSys and Sutton SignWriting. A comparison was performed of these systems in order to select the most suitable system for video annotation. In section 2.1.1, an introduction to language is provided as a background to sign language. Section 2.1.2, in turn, provides a background, history and introduction to sign language which forms the base for sign language notation systems. The five parameters that fully characterize any sign language gesture sequence are introduced and explained. Section 2.2 provides an overview on each sign language notation system. Subsequently, section 2.3 performs a comparison between these notation systems with respect to each of these parameters leading to a conclusion of the most suitable sign language notation system for use in this research.

2.1 Sign Language

2.1.1 Definition of Language

The term “language” can be defined as a system for encoding and decoding information [64]. Human languages are used to communicate in human culture and can be thought of as an individual’s internal understanding which is processed into an encoding of words. These words are produced using one or more bodily organs and are transmitted using one or other medium. They are then received by a target communicating party using one or more bodily organs and are decoded into information that lead to an internal understanding. [44, 54, 59, 64].

Languages are not inherited through genetics in humans. The words of a language do not carry an inherent meaning. Any word can be arbitrarily associated. For example, the

spoken word mist may refer to the phenomenon of small water droplets being suspended in air whereas it translates to manure in German. The word ape refers to a specific primate in English whereas it refers to a bee in Italian [25]. Thus, language is a system of symbolically representation and medium of conveyance of an internal understanding. Therefore, it is an encoding and decoding of an internal understanding [54, 64].

2.1.2 Background and History of Sign Language

The majority of people use spoken languages to communicate. However, approximately 364 million deaf people worldwide are unable to use spoken language to communicate because of their hearing disability. Therefore, deaf people have to rely on an alternative form of communication, namely sign languages. Sign languages are visual gestural languages that enable deaf people to communicate with other deaf people, when communication can not be achieved using spoken languages.

Historically, sign languages were developed and used by deaf people out of the need to communicate with other deaf people. For example, in 1760 L'Epee, a French lawyer, observed two deaf sisters communicating with each other using a visual gestural language that they had developed on their own [64]. L'Epee undertook the study of sign languages based on this observation. He later developed a complete sign language for the deaf in France now known as French Sign Language (FSL).

In contrast to spoken languages that are perceived using the ears and performed using the mouth, tongue, lips and vocal chords, sign languages are perceived using the eyes and performed using the upper body, face, hands, arms and head.

2.1.3 Sign Language Misconceptions

There are various misconceptions about sign languages. Some of these misconceptions are described and clarified below.

A major misconception is that there is only one sign language used by all deaf people. There are many different sign languages and these vary from country to country similar to spoken languages that vary from country to country. Example are French Sign Language (FSL) from France, Irish Sign Language (ISL) from Ireland, Greek Sign Language (GSL) from Greece, Japanese Sign Language (JSL) from Japan, and South African Sign Language (SASL) from South Africa.

Another misconception is that sign languages are visual gestural representations of spoken languages or that sign languages are direct derivatives of spoken languages. As a

result of this misconception it is assumed that deaf people are capable of reading and writing in spoken languages. In 1960, Stokoe proved that sign languages are full-fledged languages of their own, and have complete linguistic characteristics, as do spoken languages [64]. Sign languages are not derivatives of spoken languages. Although they are influenced by spoken languages, Stokoe showed that they are largely independent of spoken languages. Therefore deaf people are not automatically literate in spoken languages.

Finally, following from the previous misconception, it is erroneously believed that sign languages are written using the writing systems of spoken languages and that they do not have writing notations of their own. There are a variety of writing notations that can be used to denote sign language such as Stokoe, HamNosys and Sutton SignWriting. Section 2.2 covers these notation systems in detail [45, 64].

2.1.4 Overview of Sign Language Structure

Sign language gestures are of two types: manual and non-manual gestures. Manual gestures include the use of the arms and hands and to make a variety of handshapes, orientations and movements. Stokoe and other researchers [17, 36, 45] have characterized manual gestures using the following elements: hand shape, orientation, location and movement. Manual gestures are used to display a concept akin to sentences in spoken language. and facial expression [64].

Non-manual gestures include the use of the shoulders, head, body and facial features such as the eyebrows and lips to produce a variety of facial expressions, head motions and body positioning and movement. Non-manual gestures support and supplement manual gestures and signal emphasis and emotion of the concept akin to the tonality of voice in spoken languages. Examples include signalling confusion by furrowing the eyes and shrugging the shoulders and expressing joy by smiling and nodding the head.

Similar to spoken languages, sign language words consist of smaller units called phonemes. Phonemes are the smallest units of sound in spoken languages. For example, the letter ‘e’ is associated with multiple sounds such as the /e/ sound in ‘egg’ or the /i/ sound in ‘e-mail’. Each of these sounds is considered a separate phoneme that can be used to construct multiple words. Similarly, sign language words are comprised of phonemes. The contrast between phonemes in spoken languages and sign languages is that phonemes in spoken languages are sound units, whereas phonemes in sign languages are gesture units. These gesture units comprise varied hand shapes, locations, orientations and movements.

For example the flat hand touching the forehead and moving away from the forehead in an upward direction is the SASL sign for “hello” whereas the same flat hand touching the chin and moving away in the same upward direction is the SASL sign for thank you. Therefore, the flat hand and upward motion are two gesture units or phonemes in SASL.

Researchers [17, 45, 64] have defined five parameters to fully characterize any sign language gesture sequence. Four of these parameters make up the manual component of the gesture and are movement, handshape, location and orientation. The final parameter which defines the non-manual component is called the non-manual grammatical signalling (NMGS) parameter. Any sign language gesture sequence is characterized by a distinctive handshape, orientation and location followed or preceded by a movement. A facial expression of other non-manual gesture may also be used for emphasis. In section 2.3 a comparison for each Sign Language Notation System is explained in section 2.2 with respect to each of these parameters.

2.2 Sign Language Notation Systems

There are three major sign language notation systems namely Stokoe, HamNoSys and Sutton SignWriting. In this section a detailed description of each sign language notation is provided and a subsequent comparison is carried out between the notation systems.

2.2.1 Stokoe Notation (SN)

The Stokoe Notation (SN) was invented by Stokoe in 1960 and was published in his book “Sign Language Structure” [64]. The SN characterizes sign language words using three phonemic components: a Tabula (tab), a Designator (dez) and a Signation (sig). The tab specifies the location at which the sign takes place relative to the body. The dez specifies the shape and orientation of one or both hands in the sign. The sig specifies the motion involved in the sign. Figure 2.1 is a simple SN representation of the American Sign Language (ASL) word “don’t know” and shows the tab, dez and sig components of the SN.

As seen in the figure 2.1 SN uses symbols from both the Latin and English alphabets as well as other symbols invented by Stokoe. Figure 2.2, 2.3 and 2.4 summarize major taxonomy symbols used in SN to represent the tab, the dez handshape, the dez orientation and the sig. It is written from left to right as [tab][dez][sig]. Figure 2.5 is a SN representation of the common childrens story “Goldilocks and the three bears” in ASL.

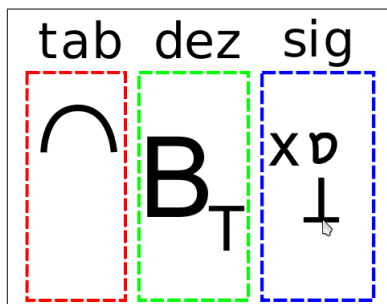


FIGURE 2.1: Stokoe Notation: Tab, Dez and Sig [66].

The SN does not support the transcription of non-manual gestures at all. Stokoe reasoned that non-manual gestures were too complicated to characterize and transcribe. He stated that it was necessary to first establish more research on manual gestures and only later focus on non-manual gestures and their parameterization [64].

SN was developed specifically for ASL and has been modified to suit some other sign languages [22].

There is an SN font that is available as a true type font [55] that can be used to type and render SN. As of 2009 web-browsers support the rendering of this font making it suitable for the development of the system required by this research. Figures 2.2, 2.3 and 2.4 provide the ASCII and Unicode equivalents of the SN font symbol set.

Unicode	ASCII	Stokoe	Description
∅	0	Q	neutral location
∅	Q	h	face, or whole head (symbol is superimpose ^ and v)
n	P	u	forehead, brow, or upper face
u	T	m	eyes, nose, or mid face
u	U	l	lips, chin, or lower face
3	}	c	cheek, temple, ear, or side face
∏	N	k	neck
[]	[]	[shoulders, chest, trunk
2	7	i	upper arm
√	J	j	elbow, forearm
∅	9	a	inside of wrist
p	6	b	back of wrist

FIGURE 2.2: Stokoe Notation: Tab, (Tabula) refers to the sign location [55].

2.2.2 The Hamburg Notation System (HamNoSys)

The Hamburg Notation System (HamNoSys) was invented in 1985 at the University of Hamburg in Germany as a phonetic transcription notation for sign language. HamNoSys is based on and extends SN and follows the same structure as SN. Figure 2.6 illustrates the written structure of HamNoSys. Figure 2.9 is a depiction of the ASL words ‘ball’, ‘house’ and ‘box’ in HamNoSys. HamNoSys provides an extended and more intuitive approach than SN in the representation of handshapes. It uses symbols for hand shapes

Unicode	ASCII	Stokoe	Description
A	A	A	fist (as ASL 'a', 's', or 't')
B	B	B	flat hand (as ASL 'b' or '4')
5	5	5	spread hand (as ASL '5')
C	C	C	cupped hand (as ASL 'c', or more open)
E	E	E	claw hand (as ASL 'e', or more clawlike)
F	F	F	okay hand (as ASL 'f'; thumb & index touch or cross)
G	G	G	pointing hand (as ASL 'g' 'd' or '1')
H	H	H	index + middle fingers together (as ASL 'h', 'n' or 'u')
I	I	I	pinkie (as ASL 'i')
K	K	K	thumb touches middle finger of V (as ASL 'k' or 'p')
L	L	L	angle hand, thumb + index (as ASL 'l')
3 (3)	3 (3)	3 (3)	vehicle classifier hand, thumb + index + middle fingers (as ASL '3')
O	O	O	tapered hand, fingers curved over thumb (as ASL 'o' or 'm')
R	R	R	crossed fingers (as ASL 'r')
V	V	V	spread index + middle fingers (as ASL 'v')
W	W	W	thumb touches pinkie (as ASL 'w')
X	X	X	hook (as ASL 'x')
Y	Y	Y	horns (as ASL 'y', or as index + pinkie)
8 (8)	8 (8)	8 (8)	bent middle finger;

FIGURE 2.3: Stokoe Notation: Dez, (Designator) refers to handshape and orientation [55].

Unicode	ASCII	Stokoe	Description
D \wedge	D \wedge	D \wedge	moving upward
D \vee	D \vee	D \vee	moving downward
D \mathbb{N}	D \mathbb{W}	D \mathbb{r}	moving up and down
D $>$	D $>$	D $>$	to the dominant side
D $<$	D $<$	D $<$	to the center or non-dominant side
D \cong	D \mathbb{z}	D \mathbb{z}	side to side
D \top	D \mathbb{t}	D \mathbb{t}	toward signer
D \perp	D \mathbb{f}	D \mathbb{f}	away from signer
D \dagger	D \mathbb{m}	D $\mathbb{=}$	to and fro
D α	D \mathbb{a}	D \mathbb{a}	supinate (turn palm up)
D β	D \mathbb{b}	D \mathbb{b}	pronate (turn palm down)

FIGURE 2.4: Stokoe Notation: Sig, (Signification) refers to motion and action, this figure represents the Sig movement symbols [55].

that visually resemble the actual handshape. Figure 2.8 summarizes symbols for major handshapes in HamNoSys.

HamNoSys also provides extended symbols for the representation of additional motions and orientations. Figures 2.8 and 2.6 summarize major symbols for the representation of motions and orientations in HamNoSys. Another feature of HamNoSys is that, as of version 4, it provides a partial character set for the representation of some non-manual gestures. Figure 2.7 summarizes some of the symbols used in HamNoSys to represent non-manual gestures.

HamNoSys is actively developed and has undergone several development iterations between versions 1 through 4. Such development extends the features of the notation system. It can be used to represent any sign language.

Similar to SN, there is a HamNoSys font that is available as a true type font that can be used to type and render HamNoSys [49, 58]. Figures 2.6, 2.8 and 2.7 provide the ascii and unicode equivalents of the HamNoSys font symbol set.

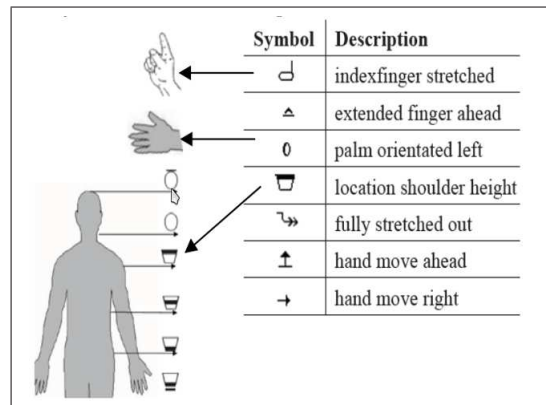


FIGURE 2.8: HamNoSys: Iconic symbols for various locations and shapes of the human body [58].

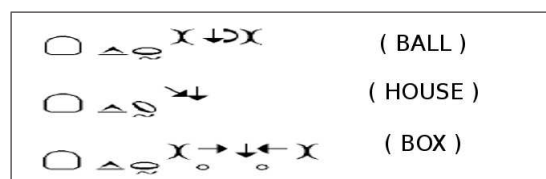


FIGURE 2.9: Hamburg Notation: Representation of the words "Ball", "House" and "Box" [65].

decision to develop the SSW notation arised from the interest of researchers to develop a written notation for sign language. Sutton modified and customized the DanceWriting notation to cater for the representation Sign Language.

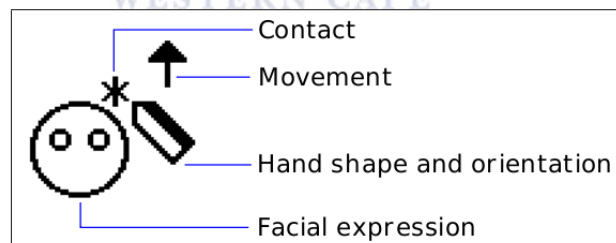


FIGURE 2.10: Sutton SignWriting : The word "hello" depicted with SSW [45].

The SSW notation uses a schematic approach for the notation. This approach does not restrict the SSW notation to a specific and limited alphabet. The SSW notation writing format significantly differs from the writing format approaches of SN and HamNoSys. It is a pictographic notation that uses intuitively shaped glyphs and pictograms to represent sign language as a notation system. The notation displays a visual representation of a sign language gesture. For example, figure 2.10 is a SSW representation of the SASL word 'Hello' and, as seen, visually depicts the hand location, shape, contact and motion as well as the facial expression. The pictograph is known as a sign box and can be filled with many different symbols representing various facial expressions, hand shapes,

orientations, locations and motions. Most symbols resemble the actual object they are representing.

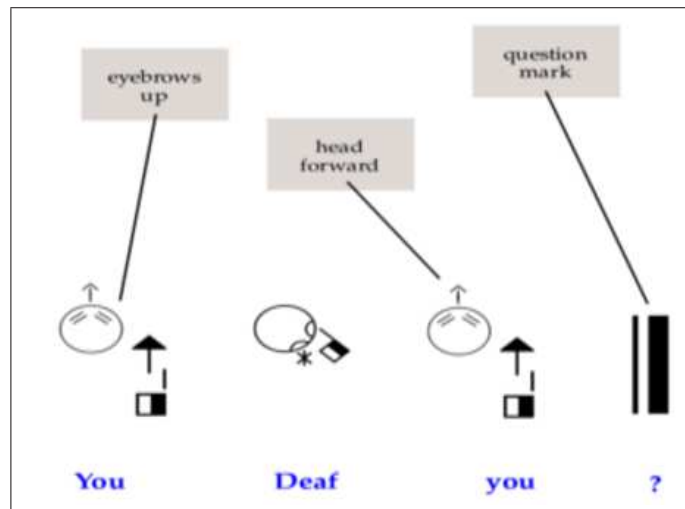


FIGURE 2.11: Sutton SignWriting : A question written in SSW [45].

Figure 2.11 depicts the SignWriting symbols used with a combination of facial expressions, hand shapes, orientations, locations and motions in order to formulate the visual written representation of the question ‘You Deaf You ?’. SSW has evolved with the help of deaf people since its invention in 1974. As such, it has been made suitable for the representation of any sign language. Although SSW notation has no official Unicode font, there is a software application in the form of a SignWriting Image Server (SWIS), which can be installed and used to build and render SSW pictographs.

2.3 Comparison Between Sign Language Notation Systems’ Parameters

This section aims to compare each of the sign language notations with respect to the sign language parameters used to characterize sign language sequences, namely: movement, handshape, location, orientation and non-manual grammatical signalling (NMGS).

2.3.1 Movement

There are three methods of representing the movement parameter in sign language notations:

- Specifying the beginning position and the subsequent movement from which the end position can be inferred.

- Specifying the beginning position and explicitly specifying the end position.
- Specifying the beginning position, the subsequent movement, and the end position.

SN and HamNoSys use the first method of movement representation thus specifying the beginning position and the subsequent movement, but leaving the end position open to interpretation. In contrast SSW allows for the use of all 3 methods to represent movement. The provision of all three representation methods by SSW emerged from requests by SSW users. Users claimed that such a provision increased the flexibility of the notation and made it easier to read the notation [44, 45]. However, this provision increases the complexity and ambiguity of SSW since there are many ways of representing a single gesture [44].

SN, HamNoSys and SSW all make use of arrows to indicate movement. However, where SSW provides arrows to represent extended movements such as spiral motions, SN and HamNoSys break movements into segments and represent them using a series of arrows. While this makes SSW simple to use and less cumbersome, it makes it difficult to process these compound movement arrows for use with the animation of an avatar. SN and HamNoSys' movement representation can be animated with greater ease since it is segmented but is more cumbersome [17, 44, 50]. Furthermore, HamNoSys and SN differ in this respect in that SN provides only a small number of arrows and can only represent a limited set of complex motions whereas HamNoSys provides a rich set of movement arrows to represent any complex motion. Again, the trade off lies between complexity and flexibility. While HamNoSys is the least ambiguous of the three notations in terms of the representation of movements and the most suitable for the animation of an avatar, it is the most cumbersome and complex. SN is less complex than HamNoSys but less flexible and cumbersome. SSW is the most ambiguous and most difficult for use with avatar animation but is the most intuitive and least cumbersome.

2.3.2 Hand Shape

There are two approaches that are used to represent handshapes namely: the taxonomic and schematic approach. The taxonomic approach associates handshapes with arbitrary symbols such as associating the letter 'B' with a flat palm handshape. The schematic approach represents handshapes using schematics shaped like the actual physical arrangement of the hand. An example is the handshape in figure 2.12 that represents a flat palm.

HamNoSys and SSW use the schematic approach whereas SN uses the taxonomic approach to represent handshapes. The advantage of the schematic approach is that the

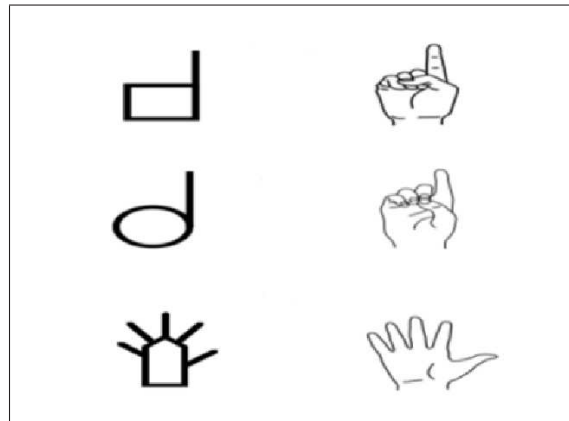


FIGURE 2.12: Sutton SignWriting: The basic handshape schematics [45].

handshapes that are represented are intuitive and easy to identify and it is easy to extend the number of handshapes by adding extra schematics as required. In contrast, the taxonomic approach is not intuitive since symbols representing handshapes are arbitrarily associated and it is difficult to represent a large set of handshapes and/or extend the number of handshapes since the alphabet used to represent handshapes is finite. In this respect, HamNoSys and SSW are more advantageous than SN. SN has been criticised for the limited number of handshapes it is capable of representing [45].

There is no difference in the approach taken to represent handshapes in SSW and HamNoSys, both notations use the schematic approach to represent handshape [44].

2.3.3 Location

Similar to the representation of handshapes, the representation of location is achieved using the taxonomic and schematic approaches. SN follows the taxonomic approach to represent location whereas SSW and HamNoSys use the schematic approach to represent location.

SN uses symbols from the Latin alphabet to represent positions and locations of the body. These symbols are arbitrarily associated to indicate location. For example, the chin is represented by an upside down ‘U’, ‘[’ and ‘]’ indicates the shoulders and a ‘C’ represents the cheek. More symbols are illustrated in figure 2.2. SN only represents 12 distinct location positions with 12 symbols. Sign language researchers later discovered that more unique locations existed that needed to be represented, Corina listing 36 new locations [43] and Johnson listing an additional 56 locations [42].

SSW and HamNoSys represent locations on a person by means of schematics that approximately resemble those locations. For example, the head is represented by means

of a circle in both SSW and HamNoSys. SSW, being pictographic, represents all other locations in relation to the head and allows room for interpretation thereof. HamNoSys, not being pictographic, explicitly defines the location and has a set of symbols that are used to represent various locations such as the torso, which is represented by a trapezium to represent the trapezoid shape of the human torso, the elbow, which is represented by two perpendicular lines in the shape of an ‘L’ representing the fore arm and upper arm in a right angle position as shown in figures 2.6, 2.8 and 2.7.

SN attempts to take the simplistic route to sign language notation but is restrictive in representing locations in sign language. Its method of representing locations is also unintuitive. SSW does not define the location explicitly and is relatively ambiguous in this regard but it is intuitive. HamNoSys is less ambiguous and explicitly defines location but this makes the notation more complex.

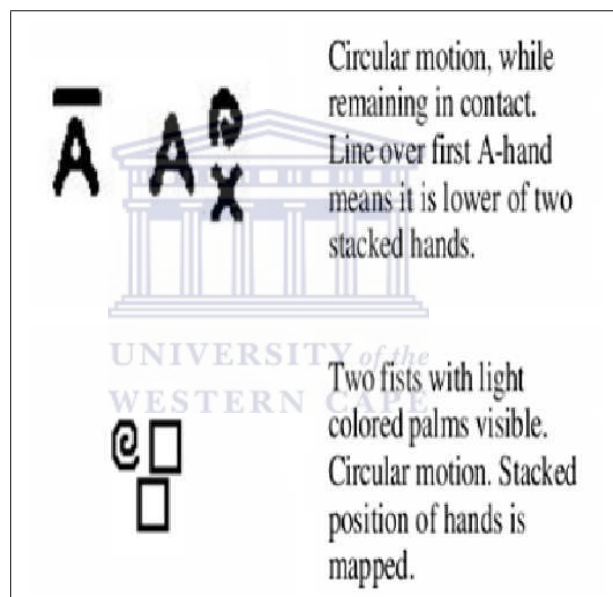


FIGURE 2.13: SN and SSW: The ASL representation of the word "Coffee". SN uses a diacritic, where as SSW uses physical arrangement for descriptive representation [45].

2.3.4 Orientation

Stokoe argued against orientation as a parameter, stating that it could be fused with the hand shape parameter such that handshape symbols simultaneously represent the orientation. Linguists later argued against this method and decided that orientation was to be a separate parameter [44, 45, 64]. This view became standard and SN was modified to integrate the orientation as a separate parameter. Thus the left to right ordering formulae had now included subscripts to indicate the orientation parameter. The ordering of the modified version included two subscript orientation parameters with

the first indicating the direction in which the fingers pointed and the second indicating the orientation of the arms. Similar to other parameters, this parameter is represented using arbitrary symbols, that is, a taxonomic approach. Figures 2.2, 2.3 and 2.4 lists many such symbols and their implied orientations.

HamNoSys represents orientation using an oval shape placed after the handshape. The oval can be placed with its long radius aligned diagonally, vertically, or horizontally to indicate the rotation of the palm in the axis perpendicular to the palm. Furthermore, the oval can be unshaded, shaded or half-shaded indicating the back of the palm facing the viewer, the palm facing the viewer and the edge of the palm facing the viewer respectively. Thus, HamNoSys is able to express complex orientations. Figure 2.9 depicts more complex representations of sign language, which use the orientation parameter.

In contrast, SSW uses an intuitive schematic approach to represent the orientation parameter. The orientation of the hands is indicated on the schematic of the hand(s) by means of a system of shading. An unshaded hand indicates the palm of the hand facing the viewer, whereas a darkly shaded hand indicates the back of the hand facing the viewer. A hand that is half shaded indicates the edge of the hand facing the viewer. Figure 2.15 illustrates three hand orientations, one unshaded, one half-shaded and one shaded. Different rotations can be indicated visually as seen in figure 2.17. Furthermore, a distinction is made between the hands aligned parallel to the floor and parallel to the wall, indicated by a gap between the palm and fingers for the alignment parallel to the floor. This can be seen in figure 2.16. The combination of shading, rotation and alignment to either the wall or floor makes it possible to represent almost any hand orientation in an intuitive manner.

SN is not intuitive in its representation of the orientation parameter and can only represent a select few orientations but it is simplistic. SSW is intuitive and can represent a greater number of orientations but is still limited in this regard. HamNoSys can represent an extensive set of orientations but suffers an increased complexity as a result.

2.3.5 Non-Manual Grammatical Signals (NMGS)

The Non-Manual Grammatical Signals (NMGSs) parameter has recently been added to the parameters used to characterize sign language. The NMGSs are also known as non-manual gestures and make use of motions of the mouth, eyes, lips, brows and shoulders like shrugging to indicate that the person does not know what has been asked. The NMGSs are necessary for speaking in sign language without which the understanding of sign language would be impossible [37, 41–44]. For example NMGSs are used in non verbal signalling, questioning, grammatical descriptors and adverbs.

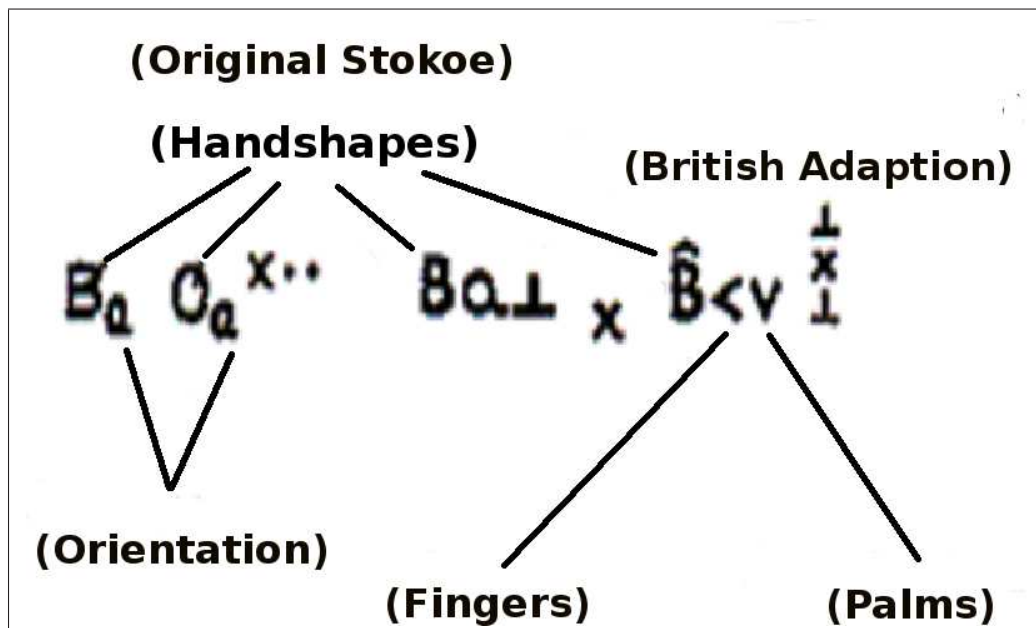


FIGURE 2.14: Stokoe Notation: The Original ASL(SN) depicted on the left and the BSL adaptation with two orientation parameters [45].

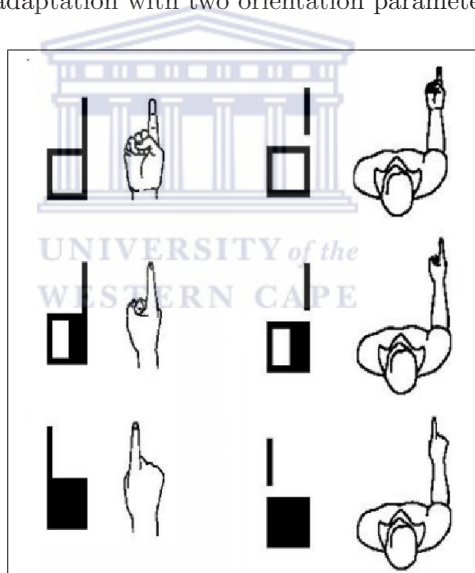


FIGURE 2.15: Sutton Sign Writer: Orientation of the hands with shading and gap between finger and palm to indicate if the hands are parallel to wall or floor [45].

Stokoe realised that NMGSs are an integral part of sign language. He stated that an analysis of NMGSs is feasible only after the basic concepts of sign language structure were examined [64]. For this reason, SN completely ignored NMGSs and does not include an NMGS parameter at all. The remainder of this discussion therefore focuses on HamNoSys and SSW only.

- Facial Expression
- Eye gaze

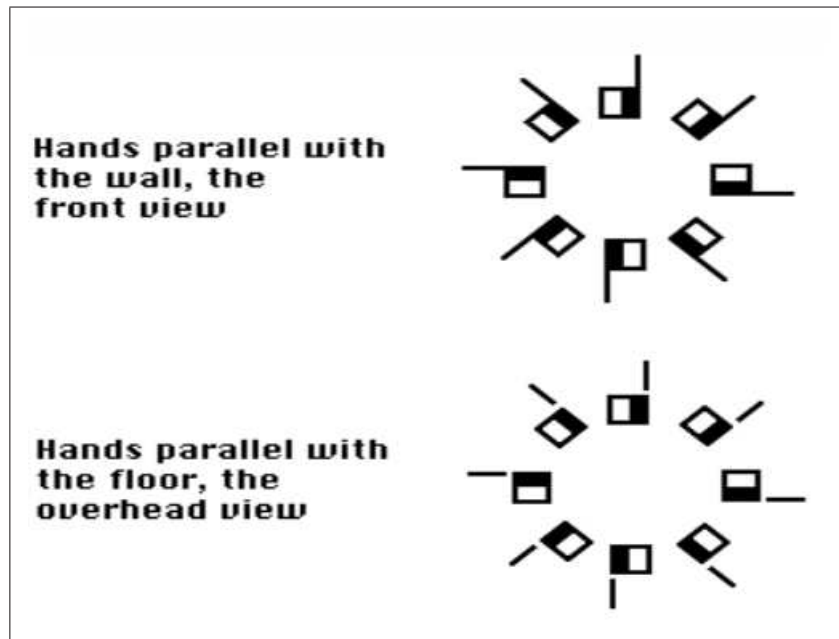


FIGURE 2.16: Sutton Sign Writer: Rotation of hands parallel to wall points "up and down" and hands parallel to floor points "forward and backwards" [45].

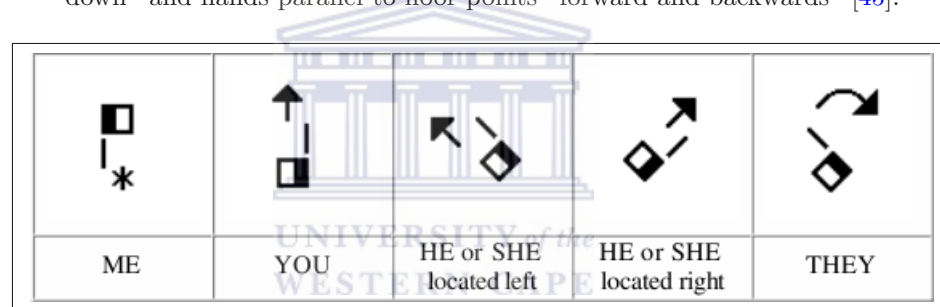


FIGURE 2.17: Sutton Sign Writer: Denoting the use of pronouns in Sign Language [45].

- Lip movements
- Body posture
- Shoulder movements
- Head position

SSW represents the face, eyes, lips and head as schematics, as shown in figure 2.18. Facial expressions are shown intuitively on the facial circle with features representing different configurations of facial features. For example different mouth shapes are indicated by means of line such as a upward arch representing a smile and a downward arch representing a sad mouth. Different configuration of the eyebrows are represented using double lines above the eye region orientated and shaped differently. For example, outward-downward double lines indicate a frown. Figure 2.18 illustrates four different facial expressions in SSW.

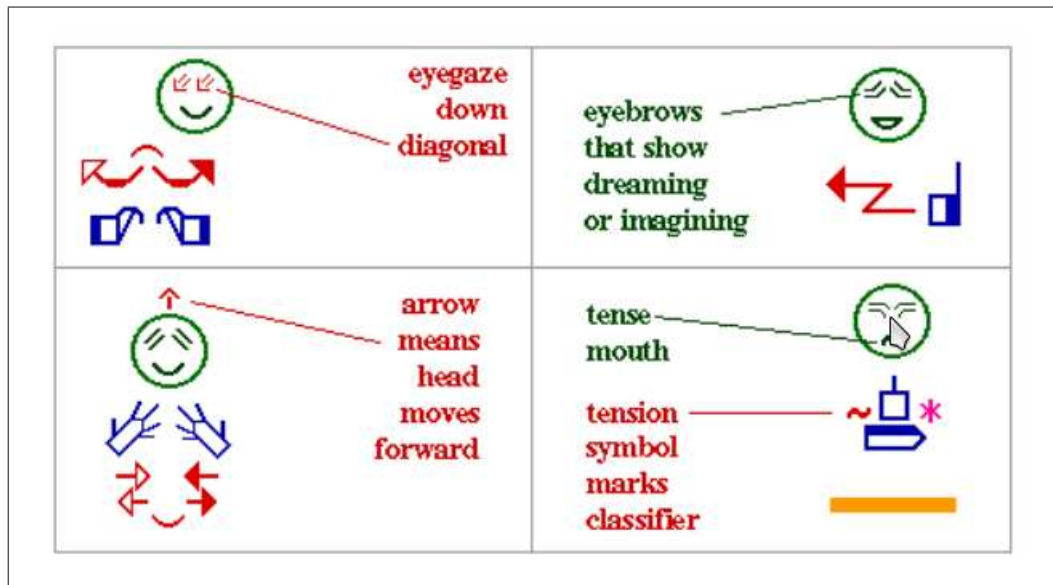


FIGURE 2.18: Sutton Sign Writer: Denoting the use of Non Manual Gesture Signalling (Facial Expressions) [45].

As of version 4, HamNoSys partially supports NMGSs [13, 26]. Symbols are provided to represent a reduced set of NMGSs, and do not cater for all types of NMGSs. Symbols are provided to represent the eyes, mouth, nose, head and chin. The shoulders are indicated using the trapezium shape icon representing the torso, shaded on the top section. NMGSs can be specified by articulating motion, location and orientation parameters onto these symbols. Therefore, it is not capable of expressing many NMGSs such as various mouth shapes and eye configurations such as squinting.

SN has an advantage of being simplistic, but does not cater for the NMGS parameters. SSW can represent an extensive set of NMGSs in an intuitive and concise manner. SSW's pictographic format ensures that even the most complicated NMGSs can be represented in a non-cumbersome way. However, it remains ambiguous such that exact interpretations of NMGSs remains open to interpretation. The addition of NMGSs adds additional capabilities to HamNoSys. However, being non-pictographic, the addition of NMGSs further complicates and congests HamNoSys. It also remains limited in its capabilities thereof and many expressions such as eye gaze, movement or squinting can not be shown using the current versions of HamNoSys.

2.3.6 Result of Comparison

SSW is intuitive and is able to represent any sign language but is ambiguous and difficult to animate avatars with. HamNoSys can also represent any sign language and is less ambiguous than SSW and suitable for the animation of avatars but is very complex and

not intuitively represented. SN is simplistic but has a restricted symbol set and does not include the NMGS parameter in its notation representation. It can not represent all sign languages.

The advantages and disadvantages of each notation make them more suitable than other notations for particular applications. For example, SN may be more suitable for limited sign language representation purposes such as representation of a simple phrase-book, SSW for moderately large multi-lingual dictionaries and HamNoSys for applications that require detailed transcription of sign language with very little ambiguity such as in the animation of avatars. Therefore, it is not possible to promote any one notation as the best notation since this is application dependent.

Therefore, this research will necessarily have to incorporate all three notation systems into the video annotation wiki for South African Sign Language.

2.4 Conclusion

In this chapter, the three most popular sign language notations were discussed. An overview of each notation was provided as well as a contextual background of sign language notations. Subsequently, each notation system was discussed in more detail with respect to the five parameters used to characterize sign language. A comparison was carried out between the representation of these parameters between the three notation systems to conclude that neither one of the notation systems was necessarily better than the other but that each was suitable to particular applications.

It is therefore concluded that it is necessary to provide annotation in all three notation systems.

Chapter 3

Video Annotation Systems

Annotation is a process of attaching comments, explanations, references, notes, remarks and links to time segments in a media item¹. There are many tools that are capable of annotating media items, but this research is restricted to those tools that specifically support the annotation of video media items in satisfaction of the first requirement mentioned in Chapter 1. Video annotation technologies transform static video content into segments and units to which are attached descriptions or annotations.

This chapter begins by explaining the concept of video annotation and the conceptual design of a system that carries out video annotation. Also, in order to potentially acquire and use an existing video annotation system, the chapter conducts a survey of the most prominent video annotation systems. These are: Anvil, E-Lan, SignStream and Vannotea. An overview of each video annotation system is provided and the discussion specifically focuses on whether or not each video annotation system meets the requirements mentioned in Chapter 1 and extended in Chapter 2. For the sake of completeness, the requirements are re-iterated here. The annotation tool is one that:

1. Provides video annotation functionality.
2. Caters for sign language annotation of videos.
3. Is collaborative.
4. Is online.
5. Provides a collaborative knowledge management system for the knowledge collected.

¹Images, audio and/or video.

6. Provides an acceptable level of usability (from the users' perspective) and performance.

Based on the findings of chapter 2, the system must additionally be able to support annotation in all three sign language notation systems. Additional relevant features and functionalities of these video annotation systems are also mentioned as an extended basis of understanding. It is stated ahead of time that no studies have been conducted to test the stress performance, load performance or the usability, from the users' perspective, of any of these systems. However, a study has been conducted that analyzes some of these systems from an expert perspective [57]. A brief summary of this analysis is provided, where applicable, as an indication of the usability and performance of the system in question. However, this summary can not and is not used to draw firm conclusions as to the usability or performance of the system.

Section 3.1 explains the conceptual design of a video annotation system with respect to the functional and user requirements as a base for the implementation of this research. Section 3.2 discusses each of the following video annotation systems: Anvil, E-Lan, SignStream and Vannotea, respectively, in Subsections 3.2.1 through 3.2.4. Subsection 3.2.5 then summarizes the evaluation of these video annotation systems with respect to the requirements of this research. The chapter is then concluded.

3.1 Video Annotation

This section discusses video annotation in terms of its conceptual structure in Section 3.1.1 and the requirements of video annotation system from a user's perspective in Section 3.1.2.

3.1.1 Structure of a Video Annotation System

Video annotations consist of two elements: content and meta-data [4, 5]. The content is the actual video data that is to be annotated and the meta-data contains the actual annotations and their temporal alignments. The meta-data is mostly stored in a file of a particular format and many standard formats exist. Examples include the MPEG-7 profile, the AS format and the TT format.

A video player displays the content to the viewer and extracts annotations from the meta-data file and displays them at the times and for the durations they are indicated to appear. The video player must either have standardized support for the annotation

format being used or it must be custom-configured to use a format that is not standard. Figure 3.1 depicts a conceptual video annotation system. Figure 3.2 depicts a simple annotation meta-data file with annotations and temporal alignments.

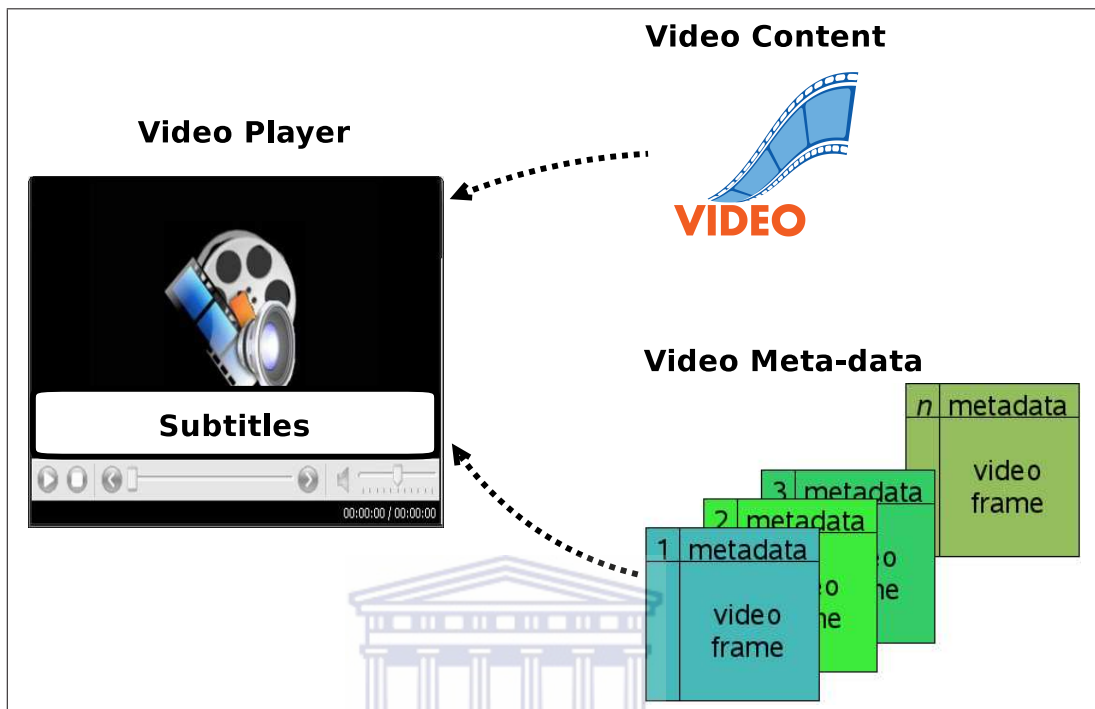


FIGURE 3.1: The two elements of video annotation namely: video content and meta-data

```

<tt xml:lang="en">
-<body>
  -<div xml:lang="en">
    <p begin="00:00:01.0" end="00:00:10.0">We wish to demonstrate</p>
    <p begin="00:00:10.0" end="00:00:15.0">the South African Sign Language for</p>
    <p begin="00:00:15.0" end="00:00:20.0">the word "Hello"</p>
    <p begin="00:00:20.0" end="00:00:25.0">"Hello"</p>
  </div>
</body>
</tt>

```

FIGURE 3.2: A simple annotation meta-data file with annotations and temporal alignments

Players mostly fall into one of two categories: stand-alone players and embedded players. Stand-alone players are applications that are installed onto an operating system and primarily work off client. Examples are VLC Player, MPlayer and Windows Media Player. Embedded players are web-based players that are integrated into the web-browser. Examples are JW FLV Player and Flow Player. While it is possible to embed stand-alone players into web-browsers, additional plugins are required to be installed on the client operating system and they are not primarily designed to work in this way.

A video annotation system that is designed to run as a stand-alone application will most likely make use of a stand-alone player whereas a web-based video annotation system will more likely make use of an embedded player.

3.1.2 Requirements of a Video Annotation System

It has been previously mentioned in this chapter that video annotation is an activity related to providing video media with commentary. Harrison & Baecker discusses the user requirements needed to design video annotation and analysis systems based on studies and surveys that were conducted [27]. These requirements are divided into four categories namely: coding the data, analyzing and interpreting the data, user interface and device control, and displaying the data. The following sections briefly discuss these requirements.

3.1.2.1 Coding the Data

A video annotation system, first and foremost, needs to be able to provide support for attaching commentary to various video segments. This process is referred to as coding the (video) data. Coded data necessarily consists of two parameters that attach commentary to video segments. These are:

- Marking the occurrence of the event.
- Marking the points at which the interval starts and ends.

Users must be able to specify these two parameters in the process of annotation.

3.1.2.2 Analyzing and Interpreting the Data

In the process of annotating a video and the coding of the data, the user needs to be able to carry out continual analyses of the data. This is done in order to view and rectify the annotations on the video. Various types of analyses can be performed. The list below describes a few general capabilities that can cater for various types of analyses required.

- Play the next annotation.
- Play the previous annotation.
- Group up annotations.

- Play a group of annotations.
- Repeatedly play an annotation or group of annotations.
- Search the annotations according to keywords.
- Display basic quantitative data such as durations of annotations.
- Time series and interaction analysis.
- Export the annotated video.

3.1.2.3 User Interface and Device Control

The video annotation system requires that a set of requirements be met by the user interface and device control. These are as follows :

- It must provide a video player to display the video to the user.
- The interface must be customizable.
- It must provide feedback mechanisms to guide the user.
- It must provide tool functionality to provide the user with an interface between other task requirements of the system.
- It must have a consistent user interface.

The video player must also cater for the following requirements:

- Play back of video at high speed.
- Play back of video at regular playing speed.
- Play back of video frame by frame.
- The ability to pause the video at any frame.

3.1.2.4 Displaying the Data

The user must be able to display the annotated data. The video annotation tool must provide a variety of customizable formats of displaying the data including the following:

- Video and annotations.

- Video only.
- Annotations only.
- Time line of events.
- Specific annotation segments only.

3.2 Video Annotation Systems

3.2.1 Anvil

Kipp developed Anvil as a gesture research tool [7, 40]. Anvil is a general-purpose video annotation system. It is used in various research areas such as Human-Computer Interaction, anthropology and linguistics. It is able to annotate a single video segment with multiple annotation elements. Each annotation element can contain text and/or audio and/or video and/or hyperlinks. It does not provide any support for sign language notation annotation [38] and, as such, does not meet the requirements of this research in this respect. It provides a graphical chart that displays the annotations of an annotated video according to the timed sequence. It uses QuickTime and AVI for capturing input video formats. It uses the MPEG-7 standard to encapsulate the video content output [38, 39].

The MPEG-7 profile is a multimedia content description standard, which was intended for building semantic relationships between multimedia content and description. This semantic relations in MPEG-7 provides faster searching and multilingual support. The profile is an XML document that contains time sequences associated with the multimedia content and description.

Anvil can be downloaded from the Internet on a link which location is provided by Kipp on request. It is free for use for non-commercial applications. It is a stand-alone application that can be installed on most operating systems such as Unix, Mac and Windows. Anvil requires the Java Media Framework to be installed on the operating system. Once the installation is complete the application can then be used. As such, it is not an online application and does not meet this requirement.

Anvil is a single-user application. It does not support the collaborative sharing of information, therefore it is not a collaborative system. It has a individualised non-collaborative knowledge management system. Therefore, it does not meet the requirements in these criteria.

An expert analysis of Anvil [57] indicates that it is very flexible since annotations can be refined at any time during the process of annotation. It allows for the deletion, addition and editing of annotations without restarting the activity. Another usability advantage is the visualization of annotations which allow the annotator to browse and notice patterns in the annotation time line display. It supports the exportation of annotations to statistical packages such as SPSS and Excel. It can also import audio transcriptions, but poorly supports the transcription of audio.

The expert analysis indicates [57] that it has a moderate learning curve which is neither advantageous nor disadvantageous to the usability of the system. It has a moderately easy-to-use interface layout.

However, the analysis indicates that it has several usability issues [57]. These include technicalities in the installation of the system on to a PC, since the system depends on Java and the Java Media Framework software. Also, it requires that a specific video codec be installed in order for the system to be usable. Limitations of the size of video files that can be loaded are another drawback of the system. Anvil only caters for 30 minutes of video. The Anvil application often crashes [57].

The Anvil user interface consists of simple windows that are designated for specific tasks. These tasks include an annotation board, element display, element edit and main window for Anvil. Figure 3.3 depicts the Anvil user interface with associated video device controls 3.3.

The following functionalities are provided by Anvil [38]

- Create project
- Create annotations
- Modifying track elements
- Change layout
- Search
- Import files
- Bookmarks
- Adjust layout

The creation of annotations using Anvil's user interface requires the user to open a video file. Once the video file has been opened the user can switch to any point on the

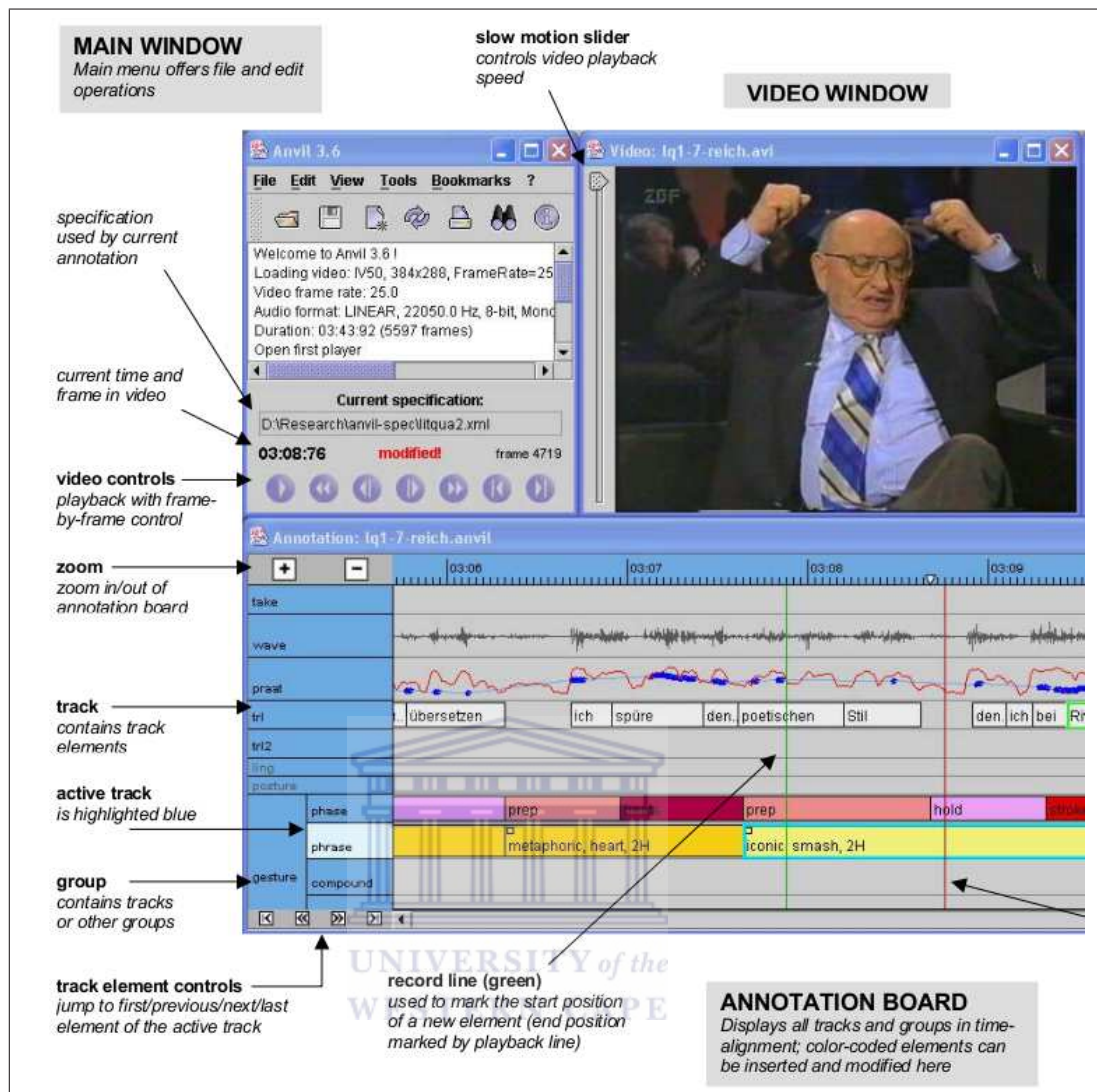


FIGURE 3.3: Anvil : Video Annotation Interface and Controls

timeline using the seek bar. The user can then insert annotation elements by repeating this procedure. The user can also delete or edit the information contained inside annotation elements or the elements themselves. The default system user interface can be customized by the user into a layout that is deemed appropriate.

To summarize, Anvil has many advantages but does not meet many of the requirements of this research and can not be acquired for use.

3.2.2 E-Lan

E-Lan (EUDICO Linguistic Annotator) is a multimedia² annotation tool that is used for documenting video content and researching the documented video content for linguistics.

²Video, audio and/or images.

Similar to Anvil, it can annotate video content with text, video, audio and hyperlinks.

Prior to version 2.4, E-Lan only supported the annotation of MPEG and MPG video formats and the WAV audio format. As of version 2.4, this restriction has been lifted and it supports a wide variety of multimedia formats. It uses the E-Lan Annotation Format (EAF), which is a custom format, to encapsulate the video annotation of content [31]. The EAF format is an XML-formated document similar to that of the MPEG-7 profile mentioned in Section 3.2.1. This format can only be rendered using E-Lan.

E-Lan supports sign language notation annotation but only in the form of the Unicode equivalent of Stokoe. Therefore it does not meet the requirement of supporting all three sign language notations.

E-Lan is freely available and can be downloaded from the E-Lan website [30]. It is a stand-alone video annotation system developed using Java. Hence, it is a platform-independent system and can be deployed on Windows, Mac OSX , Unix and Linux operating systems. Prior to the installation of E-Lan on any operating system, it is required to first install Java and the Java Media Framework. Therefore the E-Lan system is not an online system and does not meet the requirement for this research in this respect.

Similar to Anvil, E-Lan is a single-user application and does not provide multi-user or collaborative support. It has an individualised non-collaborative knowledge management system. Therefore, it does not meet the requirements in these criteria.

According to an expert analysis of E-Lan [57], it has many advantages that enhance usability. It supports a Unicode character set which allows a diverse set of characters for annotation. It provides a high level of flexibility with respect to the number and type of annotations that can be assigned to one time segment. It has an easy-to-use interface layout which can be configured according to the annotators preference and has a shallow learning curve. It can use 4 windows to display 4 different video streams. Also, it can export annotation data to a statistical package such as Excel.

However it has several usability issues. It suffers from technicalities related to the installation of the system similar to the Anvil system installation. The search function in E-Lan is restricted to that of a textual domain. It suffers from reduced functionality on Mac systems, resulting in a detachment of the video window. The export function has a limitation in that it can not extract segments as well as the annotation data. Instead, the whole video and all the annotation data is exported.

Figure 3.4 depicts the E-Lan user interface. Below the video preview area a video control tool-bar resides, with necessary icons and buttons depicted. The video control tool-bar

is an important part of the video annotation tool due to the interaction that occurs between the annotation and video preview areas. In order to annotate video content using E-Lan, users have to navigate the controls for viewing the video content and then attach an annotation to the right hand side annotation area depicted in figure 3.4.

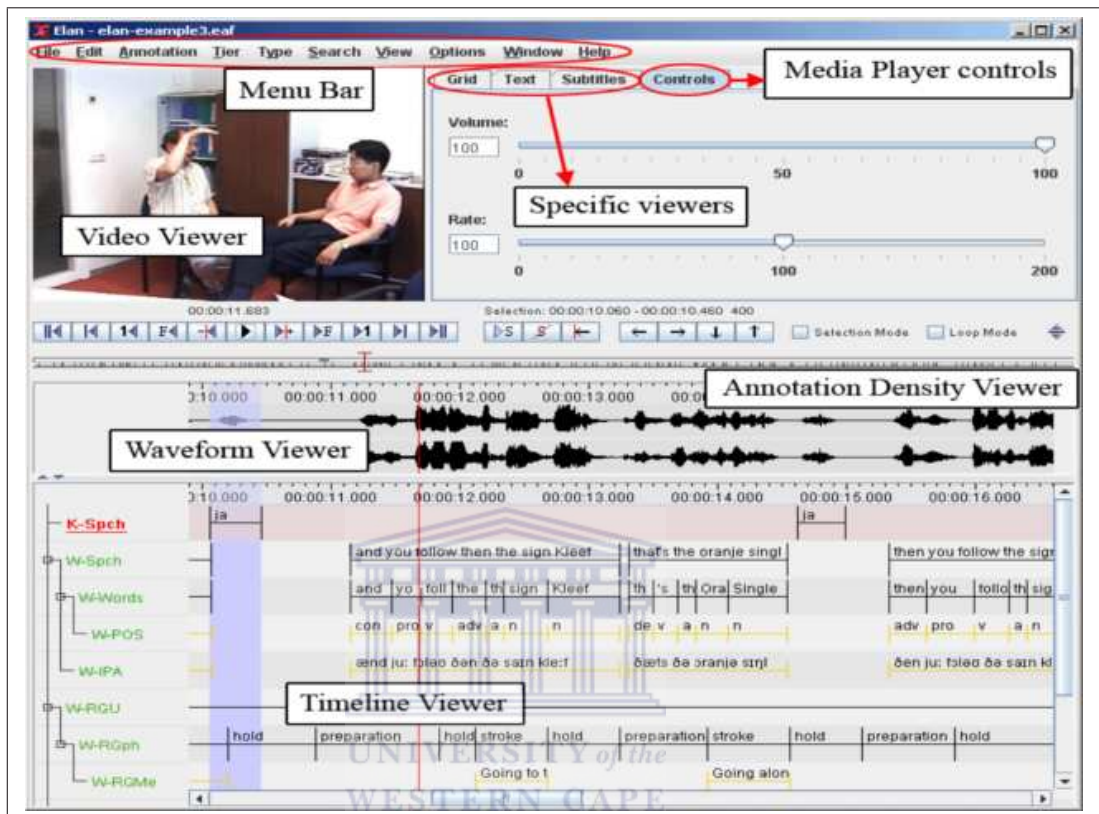


FIGURE 3.4: E-Lan: Video Annotation Interface and Controls [31].

E-Lan supports the following:

- Displaying a speech and/or video signal, together with their annotations.
- Time-based linking of annotations to media streams.
- Linking of annotations to other annotations.
- An unlimited number of annotation tiers as defined by the users.
- Different character sets.
- Exportation as tab-delimited text files.
- Importation of E-Lan files.
- Search options.

To summarize, E-Lan has many advantages but does not meet the requirements of this research.

3.2.3 SignStream

SignStream is a database research tool for the analysis of linguistic data and visual gestural languages captured on video. It was developed as part of the American Sign Language linguistic research project [51, 62] but it caters for general purpose video annotation as well. SignStream has a database of sign language phonemes (explained in Chapter 2) that can be applied when users annotate sign language video. It also supports textual annotation of video. Once users have completed an annotation, the content can be viewed using the SignStream system. Figure 3.5 depicts the user interface of the SignStream video annotation system.

SignStream was developed specifically to support sign language notation annotation for sign language research [20] but only supports the Sutton SignWriting notation system. It does not support the other two sign language notation systems mentioned in Chapter 2. Therefore it does not meet the requirement of this research. SignStream version 2.2 was specifically developed for Mac OS X and does not support Windows, Unix or Linux systems. A trial version of the system is available for download at the SignStream website [63]. It is a stand-alone application. Prior to installation it requires QuickTime4 and version 8.1 of the Mac OS X operating system.

Similar to E-Lan and Anvil, SignStream it is a single-user application and does not provide multi-user or collaborative support. It has an individualized non-collaborative knowledge management system. Therefore, SignStream does not meet the requirements of being an online collaborative knowledge management system.

SignStream has an intuitive interface for displaying [51, 62], manipulating and annotating video. It can display both start and end points of annotated content. It provides a database of defined fields and values which are, however, editable and new fields and values may be created by the user. It provides a visual representation of time aligned annotation relations among segments. It allows direct manipulation to video and audio segments corresponding to utterances or to specific items that have been coded within utterances.

SignStream has a moderate learning curve [21]. It supports only the QuickTime video codec and does not allow playback of any other formats. It can only be used on a Mac OS X operating system, hence it is not platform independent. The search functionality supports text and phonemic searches of annotated videos. It does not use an XML based

schema and can not be used with other research video annotation systems such as E-Lan or Anvil.

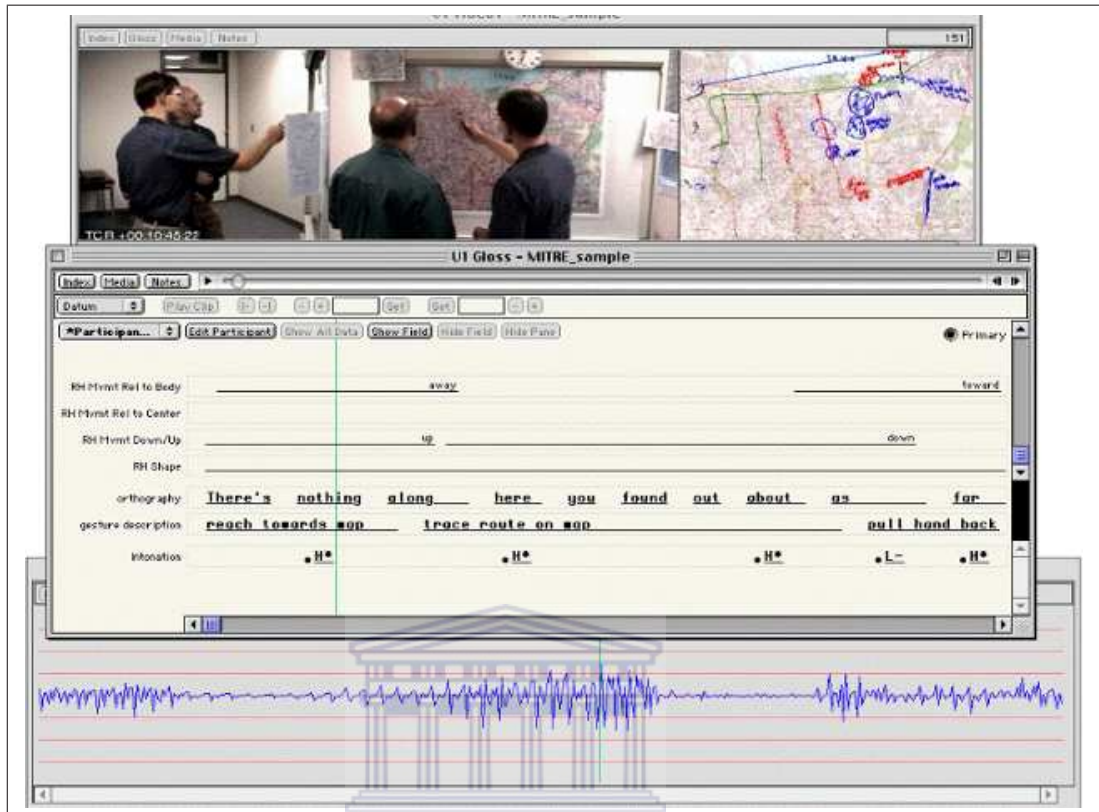


FIGURE 3.5: SignStream: Video annotation interface content player [21].

The following lists the common characteristics and activities of the SignStream system [20, 51, 62].

- Utterances may contain separate panes for distinct participants in a conversation; all information is time-aligned across participant panes. Distinct utterances (potentially from different SignStream databases) can be viewed on-screen at the same time.
- The user is able to incorporate multiple movies into a single database and allows potentially overlapping segments of video to be coded in distinct utterances.
- An integrated search capability is provided. This provides a script facility, making it possible to save and play subsets of utterances in a given order based on the search criteria.

The SignStream application provides many features and advantages but does not meet the requirements of this research.



FIGURE 3.6: SignStream: Video annotation interface content player [20].

3.2.4 Vannotea

Vannotea is a collaborative video annotation and analysis tool. Vannotea has a broad spectrum of applications and thus qualifies this system as a general purpose video annotation technology. Vannotea was developed to document video content relating to medical science and techniques associated with surgery. It supports the annotation of video content with text, audio, hyperlinks and additional video. However, it does not provide any support for annotation with any sign language notation system. Therefore it does not qualify based on this research requirement. It supports many video formats, such as mkv, flv, mpeg and mp4 as inputs for video annotation [29].

Vannotea uses the client-server model. The Vannotea system architecture is represented in [61]. The system is an online-based application and can easily be accessed using a web-browser on any operating system. Therefore it meets this research requirement of being an online system. An advantage of this is that the user is not required to carry out any installation of the software. It has been developed specifically for collaboration. It allows multi-user access to a common set of videos stored in an online knowledge base repository. Client collaborators can discuss and annotate such video content in a collaborative

manner. Therefore it meets the research requirements of being a collaborative online knowledge management system.

However, Vannotea is a proprietary system and is not freely available. It has an online demonstration version that can be accessed at the following website [67]. Therefore, it can not be used with this research.

The Vannotea system is similar to the system that this research requires. As a result some of the underlying technologies that it employs are examined. Figure 3.8 depicts the architecture of the Vannotea system. Vannotea was developed using multiple systems namely: Annotea, Jabber, Shibboleth, Vic/Rat and XACML. These underlying technologies are open source systems that are integrated to form Vannotea. Annotea is a book-marking annotation server that stores and organises generic annotation data. It was developed by the W3C as part of the Semantic Web initiative which was extended to support annotation of fine-grained contexts within multimedia objects. The Jabber system provides the instant messaging required for the real-time application sharing and event logging. Shibboleth is a middle-ware initiative that enables identity management and secure access to web resources shared amongst organizations. The Vic/Rat are videoconferencing tools that enable the recording of separate H.261 streams from participants and their conversion to tiles within a single MPEG movie. XACML (eXtensible Access Control Markup Language) is an XML-based language for defining and enforcing access control policies.

The Vic/Rat provides an interface with which to perform real-time streaming of video and audio for video conferencing. Jabber is an instant messaging protocol that can be used as a control mechanism to handle the discussion system over multiple chat clients in the Vannotea system. As a result Vannotea is able to support real time discussions and annotations of video content. Real time support is not currently a requirement of this research. Therefore, the Vic/Rat and the Jabber protocol technologies are not applicable to this research based on the requirements mentioned.

Shibboleth and XACML are tools used in systems in large organisations that require a high level of security. The Shibboleth middle-ware system manages the distribution of user security details. It is able to copy the user security from one domain to another. XACML is used to build the user access policies. Both Shibboleth and XACML are tools that allows users to access the system distributed across different domains. The scalability of security using the Shibboleth and XACML middle-ware is not a current requirement of the system of this research. As a result it is not included in this research. However it can be considered for future development of distributed user access across the system.

The Annotea server is an open source web-server used to capture meta-data from Vannotea. It is a storage system which uses the application logic to reference the stored annotation data. Annotea uses an RDF meta-data schema to store annotation data. The knowledge management system chapter discusses the use of these meta-data schemas. For the purpose of this research Annotea is a usable system, but is not the only system that can be used to develop the system this research intends to build. The implementation of the system required in this research is explained in Chapter 5.

Therefore, the Vannotea system architecture and the tools used within the system do not exactly suit the needs and requirements of the system that this research aims to achieve. Therefore, the exact Vannotea configuration is not used to develop the system required for this research. However, this research attempts to adopt the general client-server architecture of Vannotea shown in figure 3.8.

Vannotea is not a freely available system. It is built using the Microsoft .NET Framework Version 2.0 and several other tools such as Windows Media Player 10 and Quicktime 7.1.

Vannotea provides support for high quality video. Also, the use of the Jabber instant messenger for real-time discussions makes collaboration more flexible, and allows content to be shared and associated with annotations. The system provides security to users and allows access control permissions applied to each user and the user's content. It supports videoconferencing tools that can be used to stream discussions and associate them with annotations. It also caters for users with low bandwidth.

The disadvantage of the Vannotea system is that it consists of several other underlying systems that have been mentioned. These individual systems are each subject to failure. If this occurs, the Vannotea system could suffer a failure. This fact makes Vannotea vulnerable. Testing and evaluation of the Vannotea system is a cumbersome activity due to the scale of the system [10]. It does not support the display of annotation information on the timeline and does not allow the videoconferencing video as an annotation to media items.

The following are functions within the system :

- Browse : existing multimedia repositories with web front end
- View : various formats of videos, images, 3D objects, web documents, etc.
- Compare : by viewing/playing multiple media files/documents simultaneously
- Annotate : by highlighting regions and attaching notes, links, local files, etc.

- Search : the private, group or public annotation data to retrieve or analyse content
- Secure : and share personal notes
- Collaborate : in real-time through the integrated Jabber messaging architecture
- Record : event logs and audio/video from conferencing tools (Vic/Rat, Google Talk)
- Play : back previously recorded, collaborative discussion sessions

The Vannotea user interface is depicted in figure 3.7. The user interface consists of 3 sections, namely, content, video and annotation. The interface is similar to that of Anvil and E-Lan and displays all relevant information to the user relating to the individual activities.

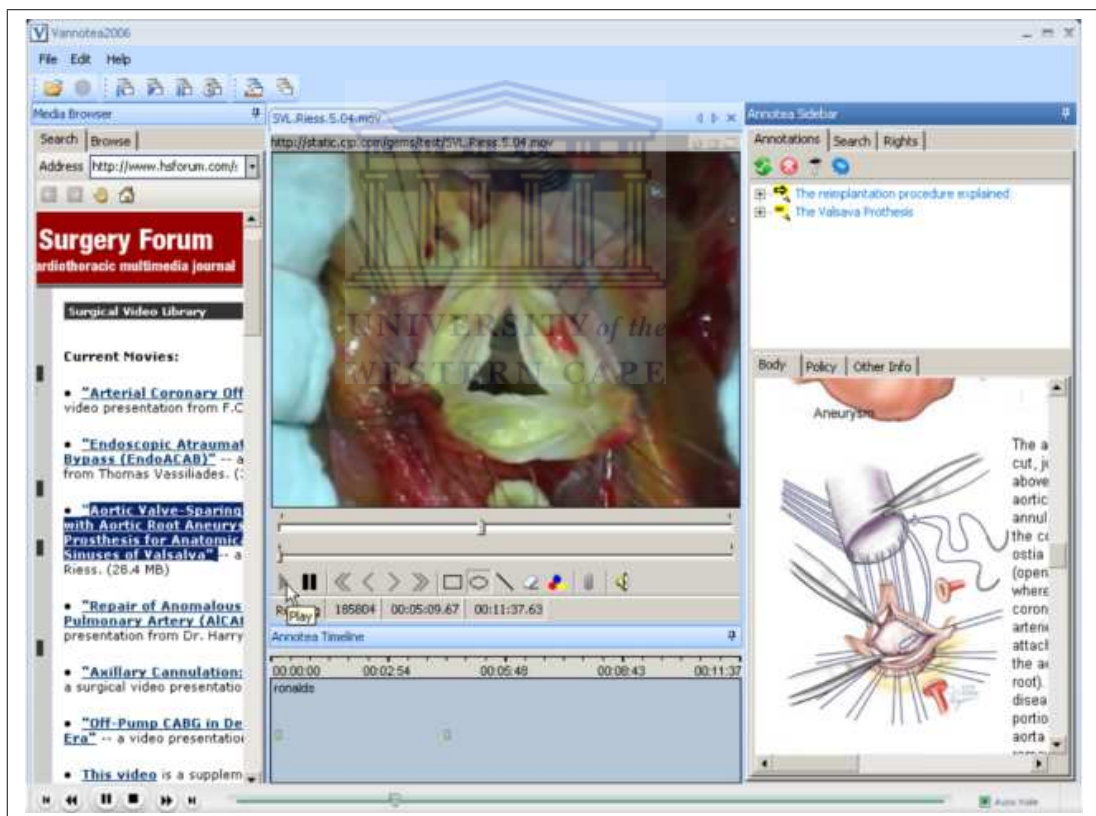


FIGURE 3.7: Vannotea: The client-side user interface of the video content and annotation panes of the Vannotea system [61].

Although the Vannotea system satisfies more of the requirements of this research than other video annotation systems, it does not satisfy any of the requirements with respect to the support needed for SLNSs.

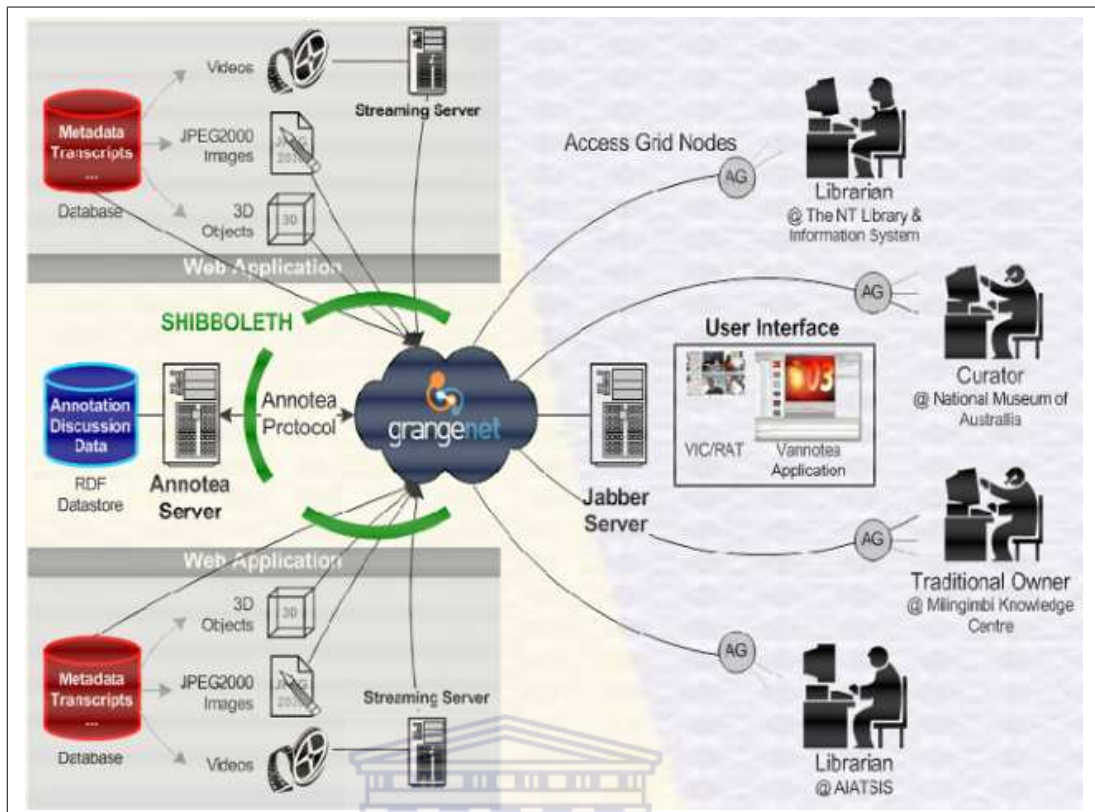


FIGURE 3.8: Vannotea: The system design and architecture of the components related to the function as the complete Vannotea system [61].

3.2.5 Summary

Table 3.1 summarizes the evaluation of the video annotation systems discussed in this chapter with respect to the requirements of this research. It consists of the video annotation systems that were evaluated with respect to satisfying the stated requirements mentioned in Chapter 1. It should be noted that the requirement of being a video annotation system has been omitted from this table since all of the systems evaluated were annotation systems of this type. Also, the requirement of an acceptable usability and performance has been omitted from this table since it was not possible to conclude on the satisfaction of this requirement due to a lack of studies in this respect.

It should also be noted that the term ‘Individualised’ stated in table 3.1 indicated the the video annotation system implemented an arbitrary method for storing the annotation information, not specifically a knowledge base. Conversely, the term ‘Public’ has been used to indicate the use of a knowledge base in this regard.

It is observed from table 3.1 that none of the examined video annotation system completely satisfy the needs of this research and therefore cannot be acquired for use as is.

3.3 Conclusion

In this chapter the most relevant video annotation systems were investigated. An overview of each video annotation system was provided and the discussion specifically focused on whether each video annotation system met the requirements mentioned in Chapter 1 and extended in Chapter 2. Additional details were also provided as a further basis of understanding. The investigation concluded that none of the video annotation systems completely satisfies the requirements of this research. Some systems satisfied more requirements than others but were still found to be unsuitable to the needs of this research. The Vannotea video annotation system architecture was found to be generally suitable as a template model for the system of this research.

It is therefore concluded that it is necessary to develop a video annotation system that meets the requirements mentioned in Chapter 1.



Requirements	Anvil	E-Lan	SignStream	Vannotea
Sign language annotation.	No	Yes	Yes	No
Collaborative.	No	No	No	Yes
Online.	No	No	No	Yes
Knowledge management system.	Non-collaborative	Non-collaborative	Non-collaborative	Collaborative

TABLE 3.1: Chapter summary of video annotation system evaluation

Chapter 4

Knowledge Management

In the previous chapter, a survey of the existing video annotation systems was conducted. It was concluded that none of these systems satisfied the requirements of this research completely and that it was necessary to implement a video annotation system for this research that satisfies these requirements. In order to develop the system required by this research, it was necessary to acquire an online collaborative knowledge management system that is able to host the video annotation component, as well as capture, store and organize the information from this component. This chapter aims to survey prominent existing knowledge management systems (KMSs) that satisfy the requirements of being online and collaborative so as to potentially acquire one of them for use.

In addition, it is desirable for the KMS to not enforce a strict user registration policy so that any user can contribute content. This is to decrease potential participation barriers and encourage deaf people to participate. It must, however, provide a mechanism to protect the KMS from vandalism. To the same end, the KMS should be popular and have a large user base in order to enhance familiarity of the system to common users and further encourage its use. It should be easily extensible. To this end, development and extension of the KMS in question should be a well documented topic and many resources should exist in this respect. It is highly desirable that the KMS be written in PHP since this is the web development language of choice of the SASL group.

Section 4.1 discusses the categories of KMSs. It explains the characteristics of each category and shows that only one category – Groupware Systems – are online collaborative KMSs in satisfaction of the requirements. Therefore, section 4.2 focuses on Groupware Systems. It enumerates four prominent systems belonging to this category that can be used in this research. It evaluates these systems in terms of the additional requirements mentioned in the previous paragraph and shows that the Wiki technology provides the most suitable user policies and content handling mechanisms for this research and that

the Chisimba framework is most suitable in terms of developer resources available. It explains a method of using the Chisimba Wiki plugin thus combining both technologies to make use of the advantages of both. Section 4.3 focuses on the structure of the Chisimba Wiki. It explains the Wiki subsystems that provide the required user policies and content handling mechanisms. It also explains the structure of the underlying Chisimba framework.

4.1 Knowledge Management Systems (KMSs)

A KMS is a system that uses various strategies to organise and store knowledge in an information repository. It is a special Information System (IS) developed by IS researchers specifically to manage and store information in organisations. The aim of KMSs is to capture and pool knowledge within an organisation into one big repository. This makes the knowledge available for analysis and facilitates the innovation of new concepts and discovery of patterns and trends using the existing knowledge. This provides increases in performance and intelligence. According to Wagner [70], Knowledge Management Systems (KMS) are used by organizations to identify, create, represent and share knowledge from experience over time.

For the purposes of the current research, the intention is to collect and pool SASL knowledge from deaf people outside the organisation into one big repository. The knowledge is made available for future analysis in order to extract the patterns and structure of SASL, which can then be used as bases of knowledge and inputs to other translation technologies developed by the SASL Project, as mentioned in Chapter 1.

KMSs are divided into seven different categories. These are: Groupware Systems (GS), Expert Systems (ES), Document Management Systems (DMS), Decision Support Systems (DSS), Semantic Network Systems (SNS) and Database Management Systems (DBMS). The following sub-sections provide a description of each of these types of KMSs.

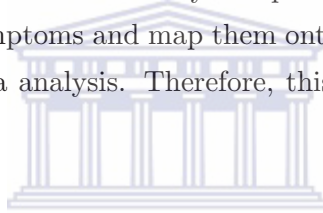
4.1.1 Groupware Systems (GS)

Groupware Systems are software systems that provide co-operative and collaborative support to the users of these systems and allows users to create, share and discuss information and content. Information is dynamically captured from and shared between users. Examples of GSs are Wiki Technology, DSpace, OpenKM and Drupal. These are Open Source online tools and are collaborative by virtue of their GS classification.

Therefore, they meet the requirements of this research and are discussed further in Section 4.2. Video Conferencing tools and instant messengers are also grouped as GSs but are outside of the scope of this research.

4.1.2 Expert Systems (ES)

Expert Systems are KMSs that use captured knowledge from a knowledge repository as input for analysis. The analysis results in a model of the data that maps the data onto a set of outcomes and can be generated using various Machine Learning techniques such as the use of Neural Networks (NN), Hidden Markov Models (HMMs) and Support Vector Machines (SVMs). The model can then be used to classify new data by mapping this data onto one of these outcomes. For example, ESs in medical science can use existing data in the form of various symptoms and build a data model that maps these symptoms onto various diseases. This model can then be used to diagnose a new patient. This is done by feeding the symptoms observed by this patient into the ES which uses the data model to classify these symptoms and map them onto one of the diseases. This research does not perform any data analysis. Therefore, this type of system does not apply to this research.



WESTERN CAPE

4.1.3 Document Management Systems (DMS)

Document Management Systems are software systems that process and string together text and images into a single data unit called a document. The document is usually stored on the user's files system in a location specified by the user. The DMS does not manage the storage of the document but only the structure of information in the document itself. Two example of DMS software suites are Microsoft Word and Microsoft Excel. DMSs do not cater for the collaborative data collection or storage and management of large volumes of data. Therefore they do not apply to the needs of this research.

4.1.4 Decision Support Systems (DSS)

Decision Support Systems are systems that carry out analysis of large volumes of collected data within an organisation. The analysis is used to identify various trends that are prevalent within the data and identify correlations between various factors. For example, a DSS can be used within an organisation to identify trends and correlations between the number of days absent from work and variables such as age, time of year and gender. This information can be used for the decision making processes by senior

management staff members. Similar to ESs, DSSs perform an analysis of the data, which is not the focus of this research. Therefore they do not apply to this research.

4.1.5 Semantic Network Systems (SNS)

Semantic Networks are systems that express data entities in the form of a directed or undirected network graph and establish relations between these entities. This results in a realization of the underlying meaning in the data and generation of knowledge from the data. An example of a system that uses a semantic network is WordNet which contains an extensive English dictionary with the definitions of the words and additionally contains relations between various words that have the same meaning and are commonly referred to as synonyms. The focus of SNSs are generally varied from the focus of this research. Therefore they are not considered or discussed any further.

4.1.6 Database Management Systems (DBMS)

Database Management Systems are systems that manage the creation and storage of data. They consist of data structures that can manage fields of various type. A query language allows users to interface with the DBMS to carry out a range of tasks including the insertion of new entries into the database and the retrieval of data that matches a certain set of criteria. Example of these systems are MySQL, Postgress, MySQL Lite and Oracle. They mostly provide Command Line Interfaces (CLIs) and require expert knowledge. For this reason, they are used in combination with other KM software that provide an easy-to-use graphical interface that serves as a bridge between the underlying CLI and the user and provides user-friendly methods of carrying out required tasks. For example, GSs use an underlying DBMS to store information. DBMSs on their own are not user-friendly but are used in this research in combination with GSs.

4.2 Groupware Systems

Specific GSs are online and collaborative and meet the requirements of this research as mentioned in the previous section. This section discusses four prominent GSs that are online and collaborative in order to potentially select one for use in the implementation of the system of this research. The two important factors that are considered are extendibility and popularity (user base). Extendibility will determine the degree of ease in modifying the system to incorporate video annotation capabilities. It is also desirable

to provide an interface that is familiar to as many common users as possible, hence, the system should be popular.

The following subsections discuss the following online collaborative GS systems: Wiki technology, DSpace, Cyn.in, Chisimba and Drupal.

4.2.1 Wiki Technology



FIGURE 4.1: Wikipedia: The user interface of the Wikipedia application [73].

Wikis are web-based applications that allow users to collaboratively create and edit content that is displayed on the site. They are built on an underlying framework that provides such functionality called Wiki technology. The underlying Wiki technology was developed by the MediaWiki foundation, which is responsible for on-going development of additional features requested by the Wiki communities [60]. Figure 4.1 is a screen shot of a Wiki interface. It provides an advanced user interface and built-in functionality to carry out a range of tasks involving management of online content. Any user on the Internet is allowed to edit and create content. The disadvantage of this approach is that it leaves the Wiki open to vandalism. In order to combat this destruction, Wiki developers have incorporated a selective semi-authentication system inside Wiki in the form of the Flagged Revisions sub-system explained in Section 4.3.1.3. Only pages that are repeatedly vandalised or are suspected to be prone to vandalism are authenticated

by an administrator before the changes made are reflected. A WYSIWYG editor is provided to provide an intuitive and easy-to-use interface for these purposes. Wikis use Apache, PHP and MySQL. The most prominent example of a website that makes use of Wiki technology is Wikipedia.

Wiki technology is free and open source and can be acquired for use from the MediaWiki website at [47]. Wikis are very popular [28]. For example, Wikipedia has a large user base of over 13 million registered users [74]. The number of anonymous users may increase this amount several times. Numerous other sites such as SourceForge and GoogleCode also use Wiki technology as documentation knowledge bases for software projects hosted. The extension of Wiki technologies is a well-documented topic with numerous online resources and developer documentation available [46]. This makes it easy to add video annotation functionality.

The success of the collaborative model of Wiki technology led to the creation of Wiki plugins for many other GS systems. These plugins provide Wiki functionality. GSs with such plugins qualify as Wiki technology.

4.2.2 DSpace

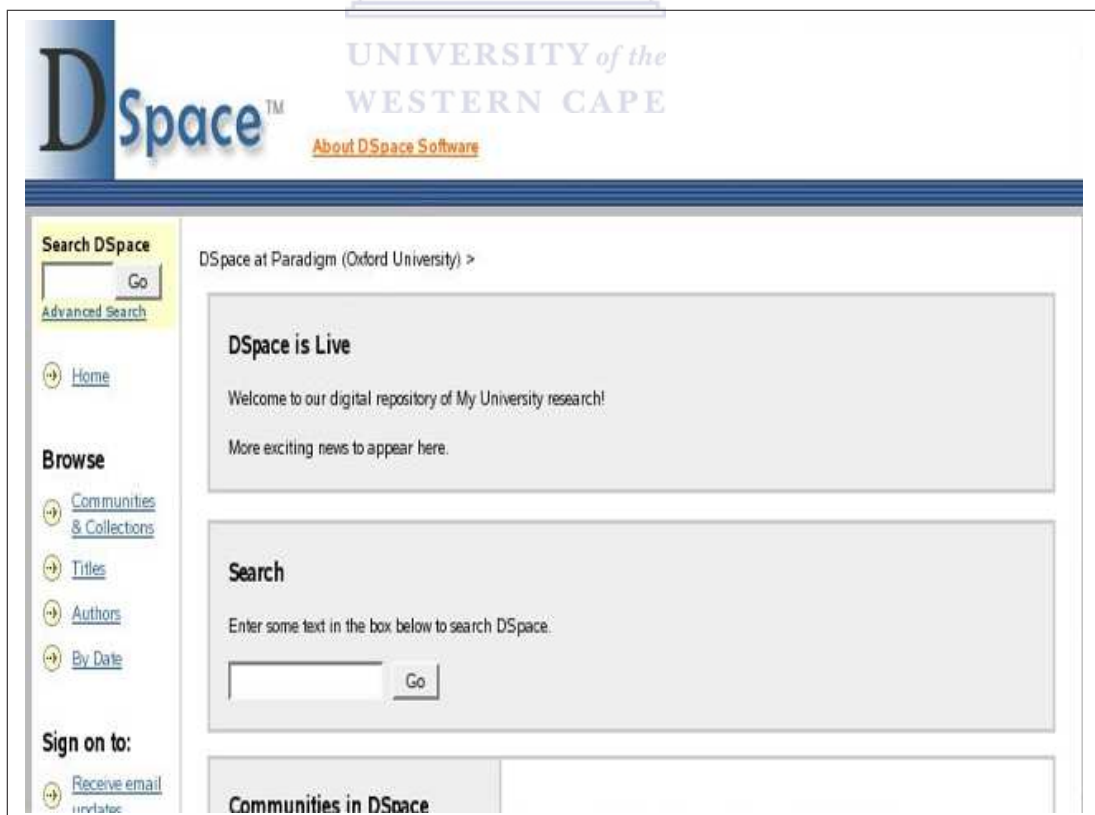


FIGURE 4.2: DSpace: The web-based client user interface of the DSpace application [75]

DSpace is a digital media repository developed by the Massachusetts Institute of Technology (MIT) and HP Labs. It provides a system that can manage digital media items such as electronic documents, images, text, audio and video. It was primarily aimed at serving as a KMS repository for institutions such as universities and public digital libraries. It can also be applied and customized to cater for organisational requirements. It provides an interface to allow uploading of various media types as shown in Figure 4.2. Users may only upload information once they have been registered. This adds complexity to the process but reduces risks of vandalism. DSpace is written in the Java programming language and uses Java Server Pages (JSPs) to interface web clients with the DSpace KMS. It has support for databases such as PostgreSQL and Oracle.

It is open source and is freely available for download at the DSpace website [19]. It is very popular amongst research and educational institutions such as Harvard University, the University of California and the NASA Langley Research Centre. Since its release in 2002, DSpace has been installed in approximately 800 such institutions internationally [75]. It is also customizable although this topic is far less well documented than Wikis.

4.2.3 Cyn.in

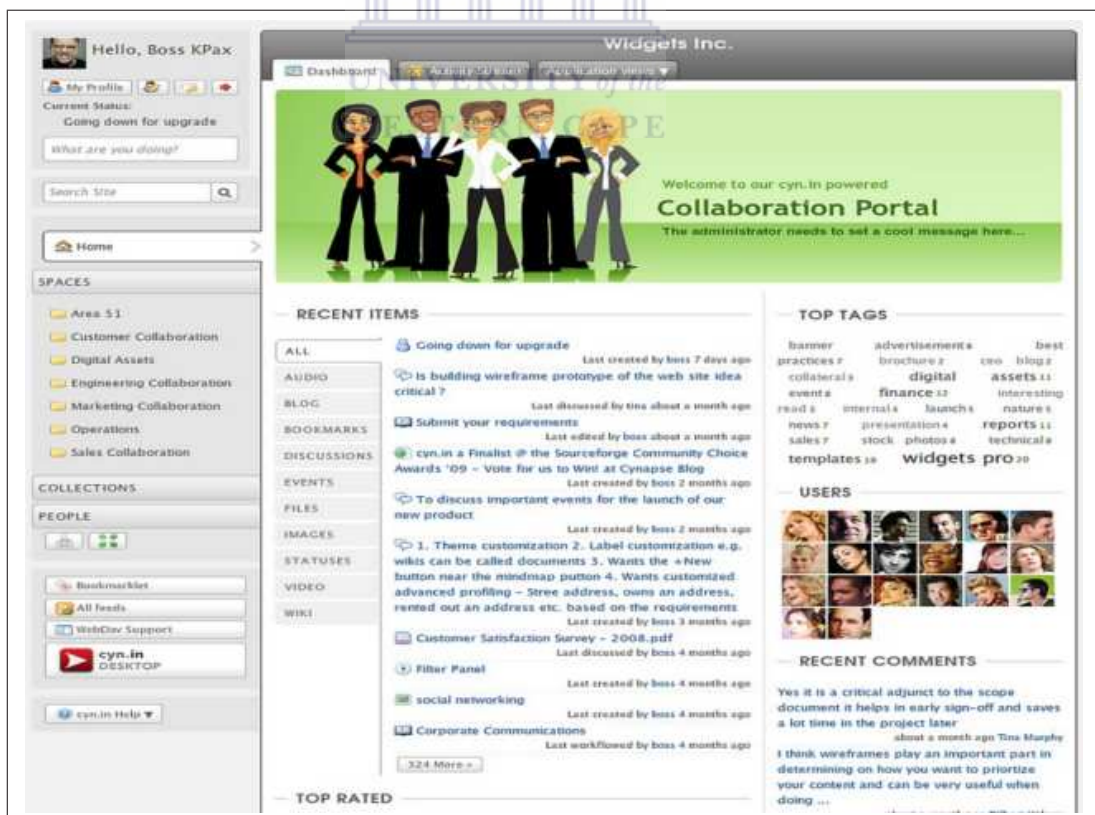



FIGURE 4.3: Cyn.in: The web-based client user interface of the Cyn.in application [12].

Cyn.in is an online content management system (CMS) framework. It allows users to collaboratively store, retrieve and organize content. It was developed by the company Cynapse. Figure 4.3 is a screen shot of the interface of a web application that is built on the Cynapse technology. Like DSpace, only registered users are able to upload digital media to the Cyn.in repository. It is written in Python and uses Apache and MySQL as server infrastructure.

It has two deployment editions: the free open source edition and the enterprise edition which is a commercial product. The community edition is a free and open source software that is available at [12]. It is claimed to be highly popular with 200,000 installations of the system worldwide, of which 4000 installations were of the open source edition [68]. Therefore, its enterprise edition is far more popular than its open source edition, with far more features, capabilities and support. Many modules are provided that can be installed into the system. These module are used to provide additional functionality such as blogging, discussion forums and special file management. There are few developer resources available for the open source edition and this makes extension a difficult task.

4.2.4 Chisimba



Chisimba is a framework created by the Free Software Innovation Unit at the University of the Western Cape. It primarily functions as a CMS and has been used as a KMS. It can act as a repository that provides the functionality for users to collaboratively store, organise and retrieve information. The standard Chisimba installation consists of the Chisimba core, which has basic CMS functionalities common to many CMS systems. Figure 4.4 is a screenshot of the interface of a typical Chisimba default installation. Chisimba enforces a user registration policy. Only registered users may contribute content to the site. The advantage and disadvantage of this approach have been mentioned. However, a Chisimba Wiki plugin has been created which transforms Chisimba into a Wiki. The resulting system provides all the same functionality and open user policy as Wiki technology. Chisimba is written in PHP and uses the MySQL database to store information.

It is free and open source software and can be obtained at the African Virtual Open Initiatives and Resources (AVOIR) website [3, 9]. It is not as popular as other GSs Internationally. It is used by 16 African higher education institutions as well as several Asian- and US-based institutions. As such, the default interface may not be automatically familiar to users. However, the Wiki plugin transforms the interface to resemble the Wiki interface and the resulting interface meets the familiarity requirements of this

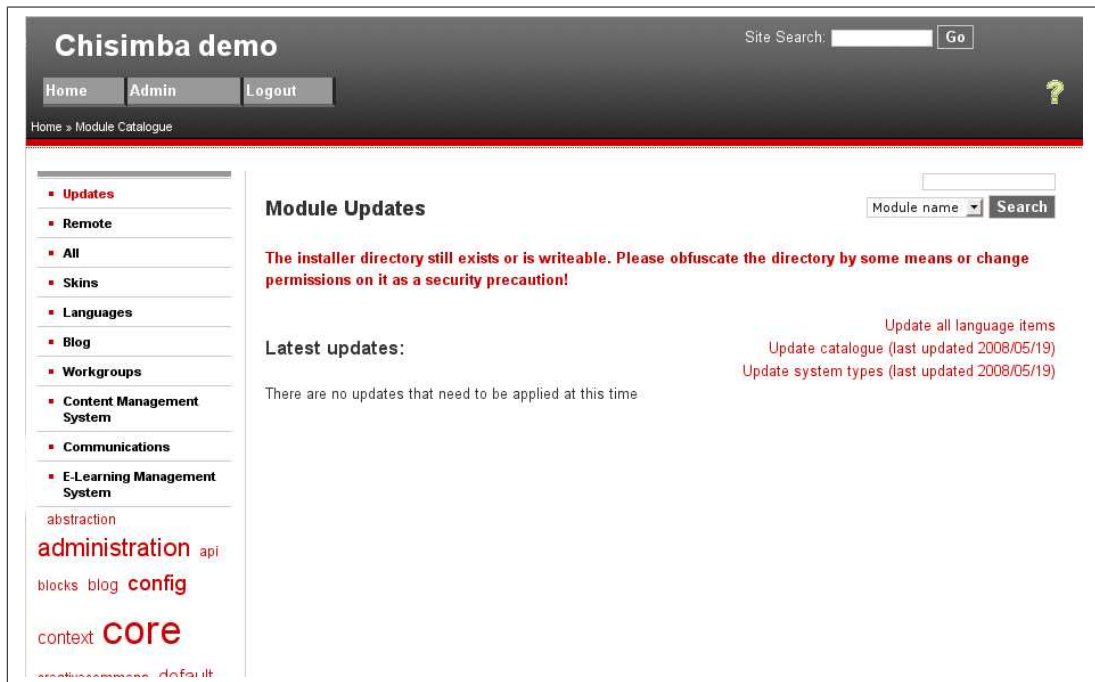


FIGURE 4.4: Chisimba: The user interface of Chisimba

research. Also, having been developed at the University of the Western Cape, the developer resources available to this research are extensive and include documentation and personal access to some of the developers of the system, which is not the case with any of the other GS systems. This simplifies the extension of the system to include video annotation and, given the existence of the Wiki plugin, makes the system very suitable to the needs of this research.

4.2.5 Drupal

Drupal is a framework which primarily functions as a CMS and has been used as a KMS and collaborative business application for organisations as well. It resembles Chisimba in many ways. It also acts as a repository that provides users with collaborative information storage, organisation and retrieval functionality. The standard Drupal installation consists of the Drupal core. Figure 4.5 is a screen shot of the interface of a typical Drupal default installation. Drupal enforces a user registration policy. Only registered users may contribute content to the site. The advantage and disadvantage of this approach has been mentioned. It also provides a Wiki plugin that can provide Wiki functionality to the system. It is written in PHP and uses the MySQL or PostgreSQL databases to store information.

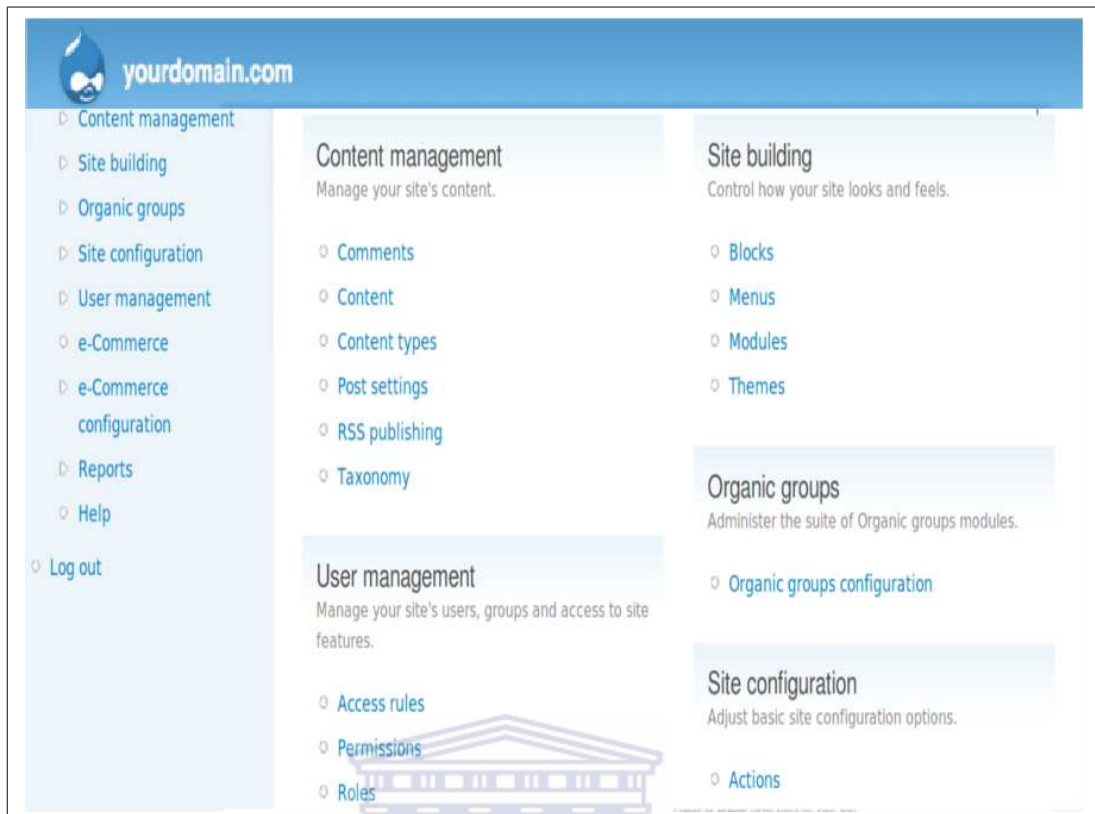


FIGURE 4.5: Drupal: The web-based client user interface of the Drupal application

It is a free and open source software available at the Drupal website at [18]. It is a popular technology used by many corporations, non-profit organisations and institutions. It is estimated that, as of 2010, 7.2 million websites use Drupal as the underlying framework. It has a large user base of over 83 000 users registered on the official Drupal community and developer websites. The Drupal framework is highly modularised and numerous extensions exist that can easily be downloaded and installed into Drupal providing a variety of specialised functionalities. It has an extension repository consisting of over 5800 freely available extensions and modules. The Drupal framework supports modules such as Blogs, Video Logs (Vlogs) and Conference Managers. Drupal is easily customizable and numerous online developer documentation and resources are available. There are also user guides for general users.

4.2.6 Summary of Section

Table 4.1 summarizes the evaluation of the KMSs discussed in this section in terms of the additional requirements set forth in the beginning of this chapter. It is apparent that the Wiki technology best satisfies these requirements and serves the interests of

this research. Therefore, it is adopted for use and is used as an implementation base for the video annotation functionality.

Wiki technology is the most suitable system to this research in terms of its user contribution policies and popularity base. Chisimba is the most suitable system to this research in terms of developer resources and extendibility. In order to combine the best of both of these systems, this research employs the Chisimba Wiki plugin. This provides a familiar interface, suitable user contribution policies as well as the vast developer resources available.

4.3 Chisimba Wiki

It has been explained that the use of the Chisimba Wiki plugin provides the best of, both, the Wiki technology and Chisimba. The user policies as well as the content generation and handling mechanisms employ the Wiki model while the underlying structures are that of Chisimba. This section explains the overall structure of the resulting system. Section 4.3.1 explains the Wiki sub-systems that provide Wiki functionality to users. Section 4.3.2 explains the architecture and structure of the underlying Chisimba system.

4.3.1 Wiki Sub-Systems

Wiki technologies have three underlying sub-systems that are used to handle and manage the content created by Wiki contributors. These subsystems are:

1. The Category Sub-System
2. The Parser Function Sub-System
3. The Flagged Revisions Sub-System.

These sub-systems were not initially part of the Wiki technology but were gradually introduced into the technology based on requests made by Wiki user communities. The development of these sub-systems has been on-going since Wikipedia was launched in 2001. Sections 4.3.1.1 through 4.3.1.3 explain the function of the Category, Parser Function and Flagged Revisions Sub-Systems, respectively.

Systems	User registration	Popularity Base	Extensible	Documentation	Language
Wiki technology	Semi-Authenticated	Common Users	Yes	Sufficient	PHP
DSpace	Strict	Research Institutions	Yes	Sufficient	Java
Cyn.in	Strict	Business Enterprises	Yes	Insufficient	Python
Chisimba	Strict	Few Research Institutions	Yes	Vast	PHP
Drupal	Strict	Common Users	Yes	Sufficient	PHP

TABLE 4.1: Summary of Groupware KMS evaluation.

4.3.1.1 Category Sub-System

The Category sub-system deals with linking information. It allows users to create content groups called categories. It was introduced into the Wiki technology in 2004 [60]. Existing content can be associated with one or more categories which link all content associated with it according to a certain common criterion. For example, content items entitled “South Africa”, “Botswana” and “Namibia” could be linked together by a category called “Southern African Countries” which links these countries based on their location in Africa. These articles may also belong to many other categories based on other characteristics such as “Countries of the World”.

Prior to the Category sub-system being introduced into the Wiki technology, the latter had very limited functionality with respect to linking and grouping information. Its addition made it possible to provide the user with related content and perform a thorough traversal of a subject matter if required.

The Category sub-system handles the linking of information in two ways using three features, namely: the created links feature and the full text search feature. Each feature is briefly described below.

The created links feature provides users with the ability to manually create links in Wiki content pages. It allows users to interlink content in two ways. The first way involves linking keywords in a content page to a Wiki content page about that keyword. For example, a page about a zoo may contain the keyword “Elephant”. The user can use the created links feature to convert this keyword into a link to a Wiki content page about Elephants. This is done by inserting a double square bracket around the keyword as in “[[Elephant]]”. The second way involves associating the content page with a category. In the previous example involving the three Southern African countries, the line “[[Category: Southern African Countries]]” is inserted into the content pages of Namibia, Botswana and South Africa would associate these content pages with that category. In both cases, the created links feature parses the information inside the double square brackets and carries out the intended task.

The full text search feature provides the user with the ability to perform a textual search of Wiki content. This feature allows users to search the content of the entire site including meta-data and files. Various techniques are used in full text searches in order to increase performance and accuracy. These range from simple techniques such as keyword searches to more advanced techniques such as proximity and fuzzy logic searches. The latter techniques attempt to provide search results that are relevant based on meaning and relationships.

4.3.1.2 Parser Function Sub-System

The Parser Function sub-system provides users with functionality to re-use existing content and display stylised content without having to interface with the underlying Cascading Style Sheet (CSS) template. It was introduced in April 2006. Prior to its release, the Wiki technology allowed users to style content by editing the underlying CSS. CSS templates were continuously re-used and re-edited leading to cumbersome CSS rule layouts with numerous clashes and numerous different CSS templates – 200,000 at the time. In parsing this CSS, browsers suffered a reduction in loading time.

Starling announced the introduction of the Parser Function sub-system in April 2006 into the Wiki software. Users would not be allowed to edit the underlying CSS template any longer but the Parser Function sub-system would allow users to interface with the CSS in a controlled manner. Users would have to use keywords to style information such as displaying mathematical expressions. The sub-system also allowed users to insert additional items into content, such as geo-coding services and widgets, without manipulating the CSS directly. This standardized the look-and-feel of the Wiki.

4.3.1.3 Flagged Revisions Sub-System

The Flagged Revision sub-system provides security against vandalism. As mentioned earlier, Wikis allow any user to edit the content of any page without requiring registration. This made the Wiki subject to potential vandalism and required that a mechanism be put in place to control it. Rather than force users to register and erode the “Wikiness” of the system, the Flagged Revisions sub-system allows Wiki administrators to flag Wiki pages that might be subject to vandalism or have been repeatedly vandalised for revision. These Wiki pages can still be edited by any user, but need to be reviewed by Wiki administrators for approval. Other pages continue to follow the Wiki policy. Rather than restrict users, restrictions are placed on sensitive content.

4.3.2 Chisimba Architecture

The Chisimba architecture is based on the MVC (Model-View-Controller) paradigm and is built on a Client/Server architecture.

4.3.2.1 The MVC Paradigm

Chisimba makes use of the MVC application model which is employed by many CMSs. This model defines three components that are used to represent different functional

Component	Description
Model	Encapsulates and abstracts storage and retrieval of raw data
View	Encapsulates and abstracts the display of user interface elements
Controller	Constructs views based on user navigation and retrieves data input from the View and updates the Model

TABLE 4.2: Components of the MVC model: Model, View and Controller [33].

aspects of the system. These are: the Model, the View and the Controller [8, 33, 33, 56]. Each of these components are described below and summarized in table 4.2.

The Model is an abstraction of the raw data in the database. Raw data may be arranged in multiple tables and may have no direct meaning. The Model constructs meaningful data structures from the underlying data pertaining to the system. These structures can be read from and written to and the changes to the underlying database tables are abstracted. For example, in an e-commerce system, raw data may be used to construct structure such as a shopping cart, customers, orders and products for sale.

The View is concerned with ways to display data from the Model to the user and present methods to collect input where necessary. It provides abstracted and encapsulated interfaces and functions that generate generic user interface elements such as lists, boxes and menus. It can also retrieve data from the Model to be displayed. Examples within the e-commerce example include the display of products for sale in a catalogue, the display of items that have been placed in the shopping cart, as well as the display of forms to enter ordering information such as shipping addresses and credit card details.

The Controller acts as an intermediary between the Model and the View. Based on the location of the user on the site, the Controller instructs the View to produce a relevant display. For example, the user may click the catalogue link at which time the controller instructs the View to construct a catalogue-type display. Conversely, the Controller retrieves data input from the user and directs the Model to make changes or additions to the data structures in the database. For example, the user may enter his/her billing information which is received by the controller which, in turn, directs the Model to insert this information into the structure containing the orders.

The MVC pattern is used in rapid web application development, and provides a practical mechanism for applications to be extensible, maintainable and scalable [33]. Figure 4.6 depicts the MVC model and shows the flow of data between different components.

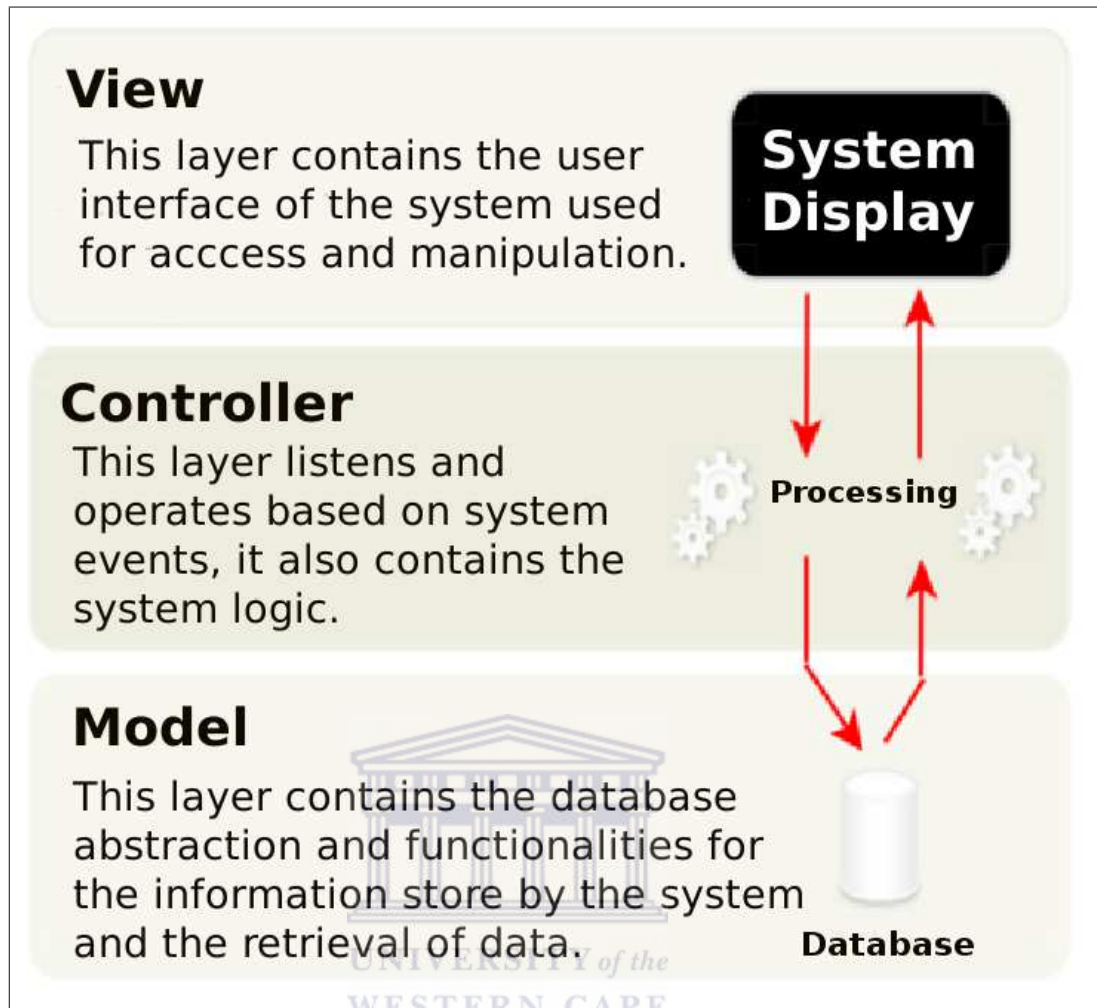


FIGURE 4.6: Encapsulation of the three domain layers and processes.

4.3.2.2 The Chisimba Component Structure

Chisimba uses the MVC paradigm. The Chisimba core consists of the Model, View and Controller that comprise a generic interface and database. Figure 4.7 depicts the Chisimba MVC core. Two additional items depicted in the figure is the object and the engine. The object is a set of classes used for the abstraction of the Chisimba MVC architecture and the engine uses the HTTP protocol for the client-server model. On top of the Chisimba core are modules each of which consist of a Model, View and Controller of their own. The Model, View and Controller of each module are configured to provide specialised functionality to that module. Each module may have database tables, display mechanisms and application logic of its own.

For example, an e-commerce module could consist of; tables such as those that store customers, products and orders; interfaces that display a catalogue and a shopping cart; and a controller which defines and manages the entire transaction service. An entirely

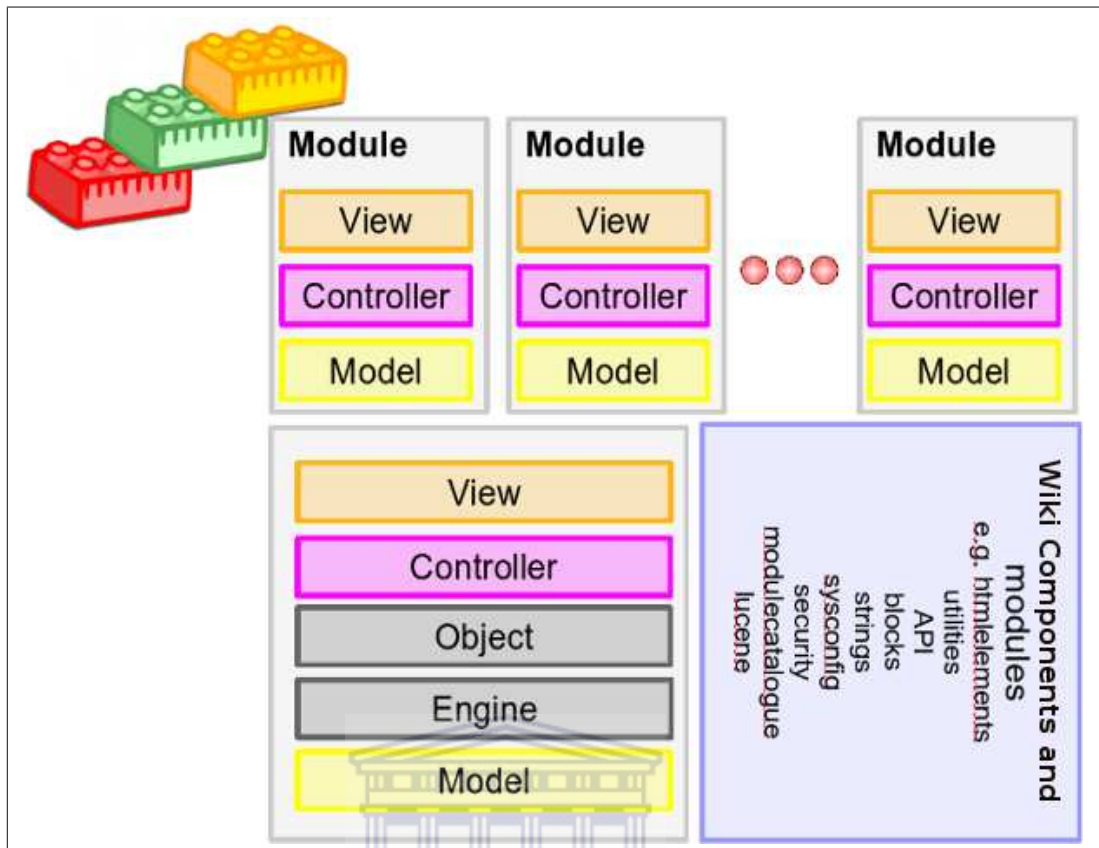


FIGURE 4.7: MVC: Using the MVC platform and helper modules

different module may be a blogging module which consists of tables that store details such as blog names, content and comments; interfaces that display the blog content and the blog comments; and the logic that allows users to add content and comments based on an authentication scheme.

An example relevant to the current research is that of the Wiki module in Chisimba, which has tables to store Wiki content; interfaces that can display the Wiki-related pages such as pages to search, view, add and edit content; and the logic that allows users to create, view and edit the Wiki content.

Figure 4.8 depicts the Chisimba MVC architecture focusing on the method by which different modules may be accessed from a web browser.

As mentioned, the Chisimba is built on a web-based Client/Server architecture. A server hosts the actual Chisimba application that is written in PHP. An Apache web-server listens for connect and page requests from clients and retrieves the requested Wiki page. The requested page is sent back to the client. The Apache web-server also provides the Wiki application with access to the MySQL database that may reside on the server itself or on a remote host. This architecture provides increased accessibility to users and is depicted in figure 4.9.

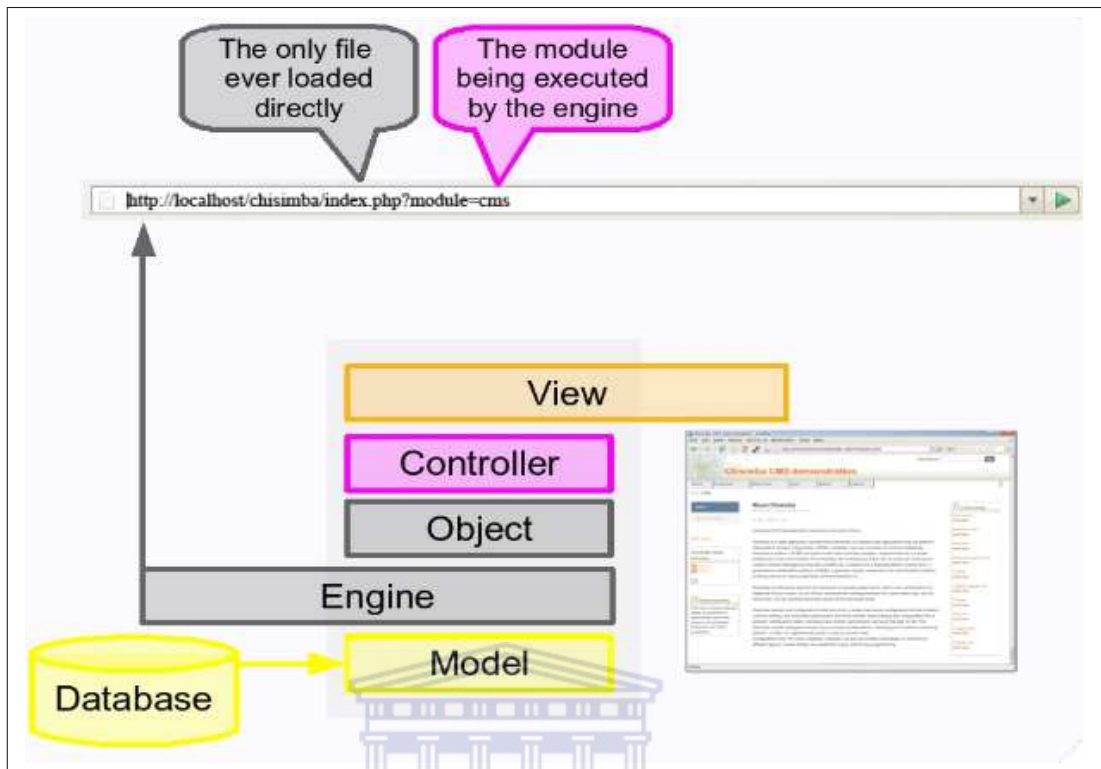


FIGURE 4.8: Software Architecture Pattern: Chisimba Model View Controller (MVC).

UNIVERSITY of the
WESTERN CAPE

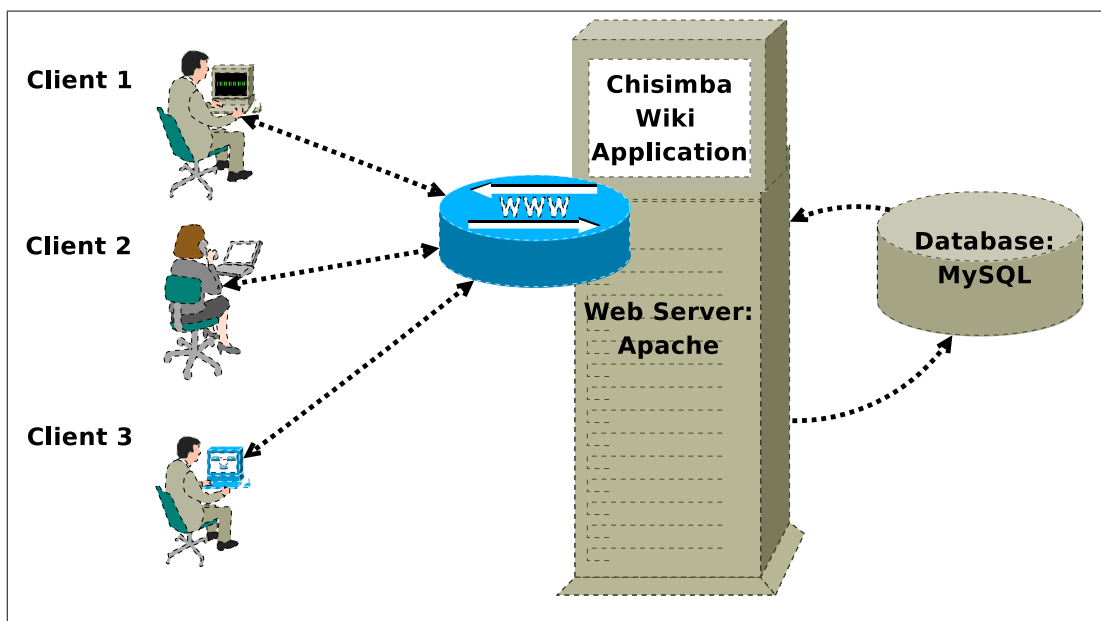


FIGURE 4.9: The architecture of the Chisimba Wiki system

4.4 Conclusion

In this chapter the most prominent KMSs were discussed in order to select the system that was most suitable for the requirements of this research. It was shown that the user policies and content generation and handling mechanisms employed by Wiki technology, which falls under Groupware Systems, were the most suitable for the requirements of this research. It was also shown that the Chisimba MVC framework, which also falls under Groupware Systems, was the most suitable framework since it was developed at the University of the Western Cape and, as such, developer resources are readily available for development in this framework. It was resolved to use the Chisimba Wiki plugin that transforms Chisimba into a Wiki application, providing the required user policies. This provided the combined advantages of both technologies.

As a basis of understanding, the Wiki subsystems that provide the required user policies were explained, as well as the Chisimba MVC structure.



Chapter 5

Design and Implementation

This chapter discusses the design and implementation of the software and technologies used to develop the system namely: the Video Annotation Wiki for South African Sign Language (VAWSASL) that satisfies the requirements mentioned in Chapter 1. It should be noted that the aim was to satisfy the requirements by providing a base framework that provides the required functionality but that will be refined and finalized in future work. Therefore, a basic set of business rules and usable interface were implemented with scope for refinement in the future.

Section 5.1 summarizes the requirements of the system required by this research, as detailed in previous chapters. Section 5.2 explains the methodology used to implement the system for both the functionality and the interface. Section 5.3 describes in detail how the system functionality was implemented.

5.1 Summary and Formulation of Implementation Specifications

In Chapter 2 the three most popular sign language notations (SNS) were discussed, namely: Stokoe, HamNoSys and SignWriting. It was concluded that support for all three SLNSs must be provided in the annotation system.

Chapter 3 discussed the conceptual structure of video annotation systems as having two elements: the content and the meta-data. It was further mentioned that a video player was required to play content and extract and display annotation data. Two types of players were mentioned: stand-alone players and embedded players. Given the resolve to adopt a Client/Server model, embedded players will most suit the requirements of this research. An embedded player is selected for use in this chapter.

Chapter 3 further described the requirements of a video annotation system from the users' perspective and found that the user must be provided with four types of functionality: the ability to carry out annotations, a usable interface, the ability to control playback and annotation processes and the ability to display the annotation data for review.

Chapter 3 also discussed several prominent sign language video annotation technologies and it was found that none of these technologies met the requirements of this research mentioned in Chapter 1 and furthered in Chapter 2. It was concluded that it was necessary to develop a sign language video annotation system that satisfies the requirements of this research. It described the Vannotea video annotation system that has a web-based Client/Server architecture and collaborative support, but did not exactly match the functionality required. It was concluded that the Client/Server model would be adopted for use.

Chapter 4 discussed various categories of Knowledge Management Systems (KMSs). It was found that only Groupware Systems (GS) provided the collaborative online support needed for the requirements of this research. Several GS systems were evaluated and it was found that the use of the Chisimba framework with the Wiki plugin satisfied many of the requirements of this research. The use of the Chisimba framework satisfies the requirements of being online and providing a collaborative knowledge management system. The Wiki plugin satisfies the requirement of being collaborative.

5.2 Methodology

In developing the system required by this research, a clear distinction was drawn between the system functionality and the user interface. The functionality of the system was implemented first. This was based on requirements derived from literature, as mentioned in the previous Chapters 2, 3, 4 and is summarized in section 5.1. Therefore, the focus was strictly on functionality rather than presentation and usability. Section 5.3 explains the implementation of the system functionality. Thereafter, the user interface was refined to cater for basic usability. In doing this, a prototyping methodology was employed. Section 5.4 describes the interface of the system and explains the methodology employed in refining the user interface.

5.3 Development of the System Functionality

This section aims to explain the implementation of the functionality of the system for this research. The requirements of being online, collaborative and providing a collaborative knowledge management system were satisfied by making use of the Chisimba Wiki. It remains to provide video annotation support in all three SLNSs. This section explains how video annotation support in all three SLNSs was implemented, thus satisfying those requirements.

This took place in three steps. Section 5.3.1 discusses the configuration of the Chisimba Wiki to, first, support video so as to create and view Wiki content with embedded video. Section 5.3.2 explains the implementation of the video annotation support on the newly implemented video Wiki content, required by the system. Section 5.3.3 discusses the implementation of the support for video annotation using the three Sign Language Notations Systems (SLNSs).

5.3.1 Configuring the Chisimba Wiki for Video Support

The type of video player chosen for use is an embedded video player. There are many embedded video players that can be adopted for use but the most popular of these embedded video players is JW FLV player [69]. JW FLV player is free for non-commercial use and is open source, flexible and easily customizable [69, 72, 76]. It supports numerous common video formats as well as many annotation, subtitle and play list meta-data formats such as ASX, ATOM, RSS, SMIL XSPF and TimedText. It is freely available for download at [71, 72]. Figure 5.1 depicts the standard web accessible JW FLV player user interface.



FIGURE 5.1: JWplayer: The user interface of the JW FLV player [72].

The default Chisimba Wiki plugin does not cater for video support, in order to cater for this functionality the JW FLV player had to be inserted into the Chisimba Wiki plugin and interact transparently in the background with the related Chisimba framework. This allowed each Wiki page to be able to display video which the user attached onto the specific Wiki content page. Embedding the JW FLV player was implemented by inserting the relevant HTML code for embedding the JW FLV player into the Wiki plugin. Figure 5.2 depicts an embedded Chisimba Wiki page with the JW FLV player.

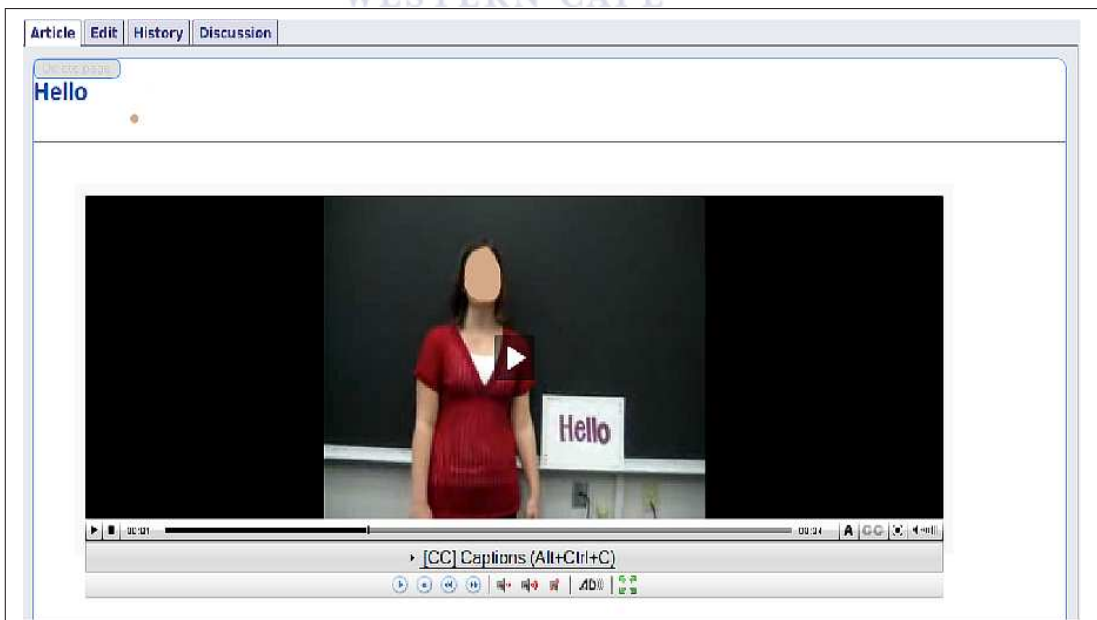


FIGURE 5.2: JWplayer: Chisimba Wiki page embedded with the JW FLV player.

The JW FLV player provides an extensive set of controls that allows the user to control playback including a seek bar and play/pause/stop buttons. As such the requirement of

being able to control playback was satisfied.

Once the JW FLV player was embedded, it was then necessary to provide the functionality for the user to upload a video file and attach the file to a Chisimba Wiki content page. This functionality requires the system to store the video file onto the system. In order to achieve file storage two methods were considered. The first was to build extra functionality into the Chisimba Wiki so that the system can store the files. The second method was to use existing functionality provided by the Chisimba framework that dealt with file management. The Chisimba framework has a file management utility called the file management module that can be used to store files into the underlying Chisimba data system. Figure 5.3 depicts the file manager.

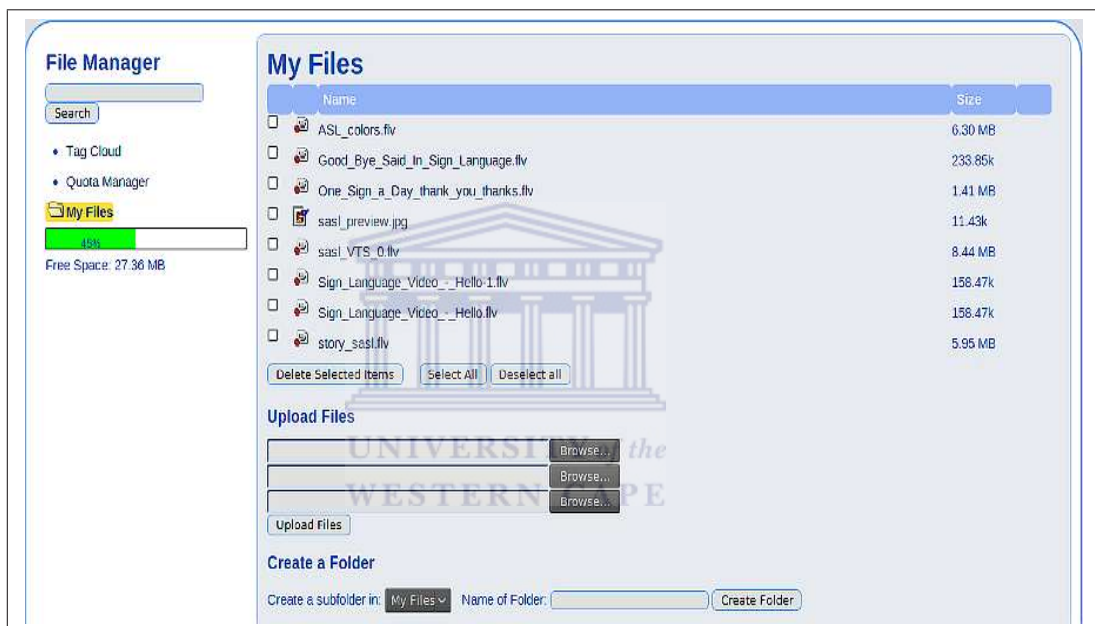


FIGURE 5.3: Chisimba framework: The user interface of the file manager module.

The second method uses the file management module, which was used and integrated into the Chisimba Wiki plugin to manage the storage and retrieval of video files. The existing page used to create Wiki content pages in the Chisimba Wiki is not used. Rather the main page of the Chisimba Wiki was modified with a file upload field that links to the file manager depicted in figure 5.4.

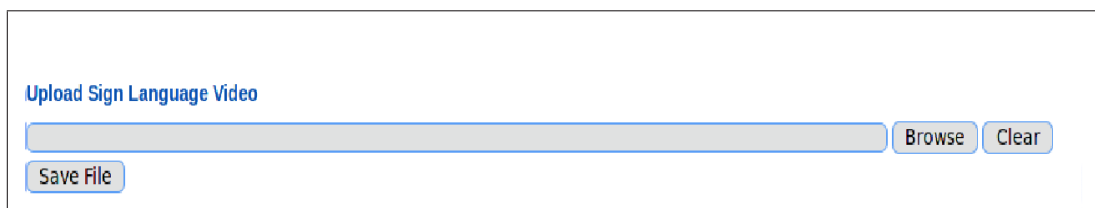


FIGURE 5.4: Modified Wiki main page with file manager link.

The file manager provides an interface that allows users to upload video files from their operating system using one of the three depicted file upload fields and stores these files onto the server. This is depicted in figure 5.3. Uploaded files are then displayed in the file manager list. Clicking the file name in the list allows a preview of the file as well as allowing it to be selected, as depicted in figure 5.5.

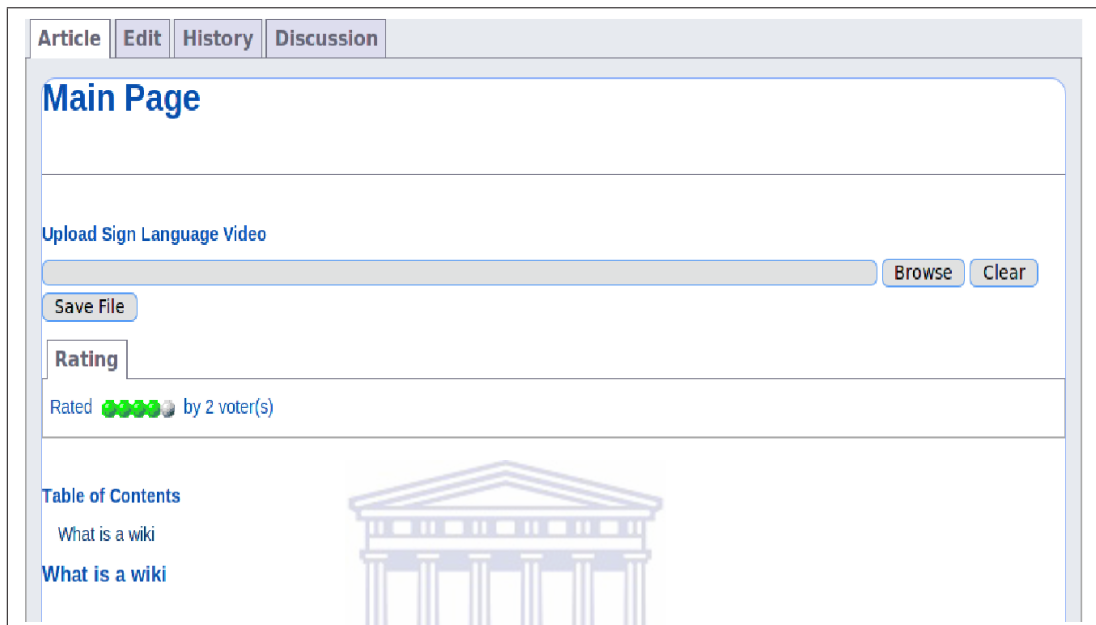


FIGURE 5.5: Button: The ‘Browse button’ is used to select a file using the file manager.

Clicking the select link depicted in figure 5.6 places the file name back into the file upload field in the Wiki main page as well as posting a hidden field containing a unique ID assigned to the video by the file manager. Clicking the ‘Save file’ button on the Wiki main page depicted in figure 5.4 then creates a Wiki content page with the selected video. The unique ID assigned to the selected video is stored along with the Wiki content page for retrieval purposes.

5.3.2 Implementation of Video Annotation Support

The implementation of video annotation support required two distinct but related activities to be catered for: adding subtitles; and retrieving and playing back subtitles. The adding of subtitles is explained first.

The JW FLV Player configuration does not cater for video annotation support by default. This meant that the functionality for annotating video had to be implemented. The JW FLV player supports a subtitling format known as TimedText. This format is encoded using XML and is stored as a separate file that can be passed into the JW FLV player along with a video file. The TimedText format has two fields: the subtitle and the time



FIGURE 5.6: Chisimba framework: The Select file link used in the user interface of the file management module.

in the video sequence at which the subtitle should appear. The time field contains the start time and end time for a specified subtitle. Once the TimedText file is passed into the JW FLV player, a parser in the player parses the TimedText document and renders the subtitles that have been annotated on the video.



FIGURE 5.7: Controls for annotating a subtitle.

A text box was added to the content page display that allows the user to enter subtitles. Two sets of drop-down menus were also added that allow the user to set the begin and end times at which the entered subtitles should appear with respect to the video’s time alignment. A button with a caption “continue” was added that adds the subtitle of the specified text and time to the database. Figure 5.7 depicts these controls.

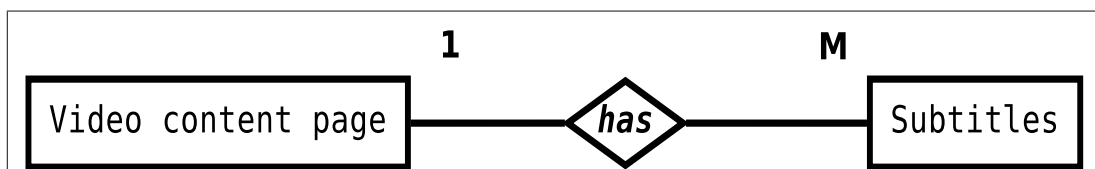


FIGURE 5.8: Table relationship between video content page and the subtitles.

A new database table was added to the Chisimba database called “table_wiki_subtitles” which will be referred to as the “Subtitles table” for the remainder of this discussion.

Field	Description
ID	Unique identifier for each row in the database (Primary key).
PageID	Unique identifier that references the Wiki video content page that the subtitle belongs to (Foreign key).
SubtitleNumber	Counter for the subtitles in a video.
Subtitle	Text containing the subtitle information.
BeginHrs	The subtitle start time in hours
BeginMins	The subtitle start time in minutes
BeginSecs	The subtitle start time in seconds
EndHrs	The subtitle end time in hours
EndMins	The subtitle end time in minutes
EndSecs	The subtitle end time in seconds
Stokoe	The subtitle in Stokoe font format
HamNoSys	The subtitle in HamNoSys font format

TABLE 5.1: Fields in the Subtitles table and their descriptions.

This table stores the annotations added to each video content page. Each video content page has a one-to-many relationship with the rows in the Subtitles table such that each video content page may have multiple rows in this table representing each of the subtitles added to the video. Figure 5.8 depicts the relationship between the video content page and the subtitles. The unique content page is identified by means of a unique foreign key. Table 5.1 summarizes the fields in the Subtitles table.

As subtitles are added during the editing process, they are added to the Subtitles table by clicking the Continue button, one row per subtitle. The subtitles provides functionality to edit and delete subtitles. Figure 5.10 depicts a subtitle entry in the row and edit and delete icons. A check box entitled “Generate” was also provided. Checking this check box and clicking the Continue button will then generate a TimedText document for that video file and, thus, commit changes. The TimedText document is named after the Wiki content page it belongs to and can be referenced and accessed as such.

As such, the requirement from the users’ perspective of providing the ability to annotate video was satisfied.

Once the TimedText document is generated, the page reloads and the video player retrieves the document and automatically renders the subtitles accordingly. This satisfied the requirement from the users’ perspective of being able to playback annotations for review.

5.3.3 Implementation of Support for Annotation in the Sign Language Notation Systems

The implementation of SLNS video annotation support, again, required two distinct but related activities to be catered for: adding SLNS subtitles; and retrieving and playing back SLNS subtitles. The following two subsections explain the implementation of each of these processes.

5.3.3.1 Adding SLNS Subtitles

It was found that the JW FLV player subtitle pane, by default, could not render the HamNoSys or Stokoe font and it could not display SignWriting graphics. Therefore, it was necessary to cater for the SLNS's using a separate viewing pane outside of the JW FLV player.

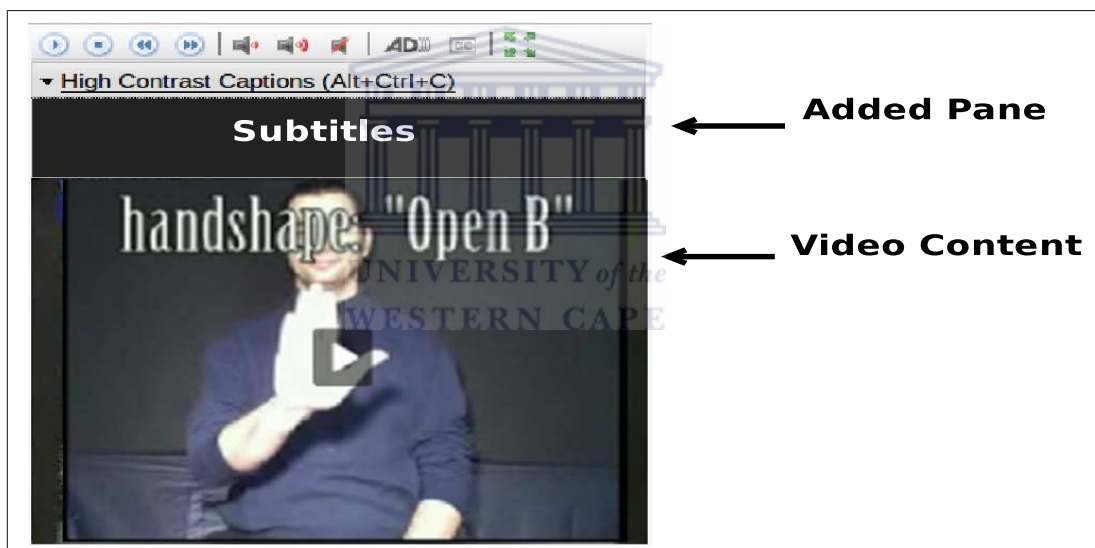


FIGURE 5.9: JW FLV player: The user interface of the video player with the additional subtitle pane area.

As mentioned in chapter 2 the Stokoe and HamNoSys fonts can be rendered using Cascaded Style Sheets (CSS) version 3, which is available in most web browsers subsequent to 2009. SignWriting is a pictographic notation and it was not possible to use a font to represent it. Therefore, the implementation of the SignWriting annotation was handled differently to that of Stokoe and HamNoSys.

A pane was added to the bottom of the content view page, which lists the annotations created by the user. This pane is, henceforth, referred to as the subtitles list. The subtitles list displays the annotation times, subtitles as well as links to edit and delete the annotation information listed in the table pane. Most importantly, it also contains

used. The subtitles list provides a button that allows the user to attach SignWriting notation to a subtitle. Once this button is clicked, a pop-up window displays the SWIS user interface depicted in figure 5.12. This user interface allows the user to drag and drop SignWriting glyphs onto the canvas provided and create any SignWriting notation pictogram.

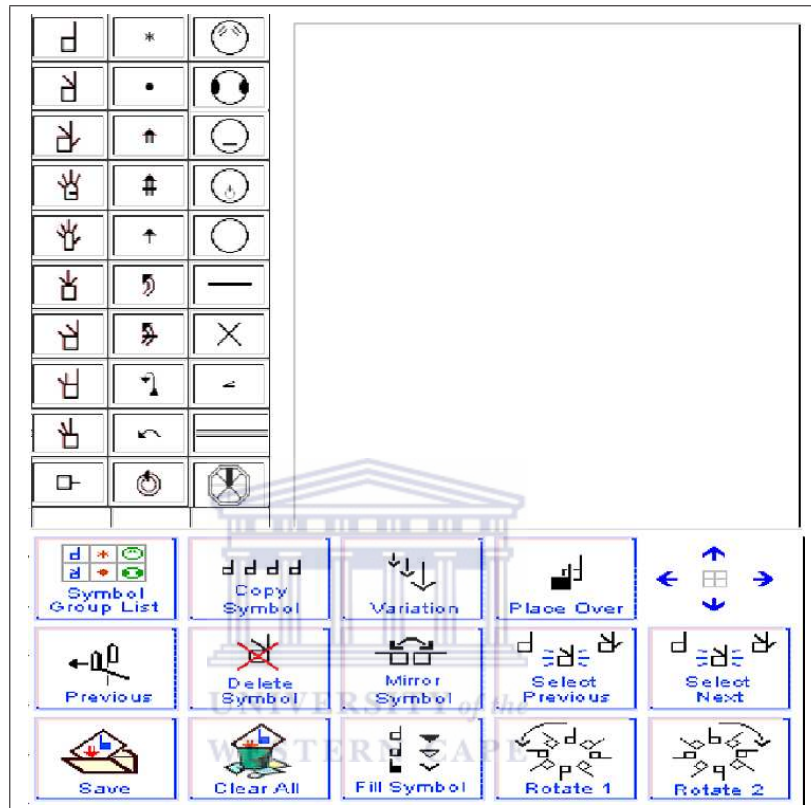


FIGURE 5.12: SWIS: The user interface with canvas and glyphs.

Each SignWriting symbol is stored as an image on the file system and is labelled with a unique number identifier. SWIS identifies a complete SignWriting pictogram configuration using a sequence of these number identifiers to identify the glyphs present in the configuration as well as information about the locations of each glyph. This identifier is henceforth referred to as the SignWriting identifier. A “submit” button is provided which can be clicked which saves the SignWriting identifier to the database. The subtitles list is refreshed and the newly created SignWriting pictogram is displayed in the “SignWriting” column as a preview.

As with the text subtitles, checking the “Generate” check box and clicking the Continue button then generates a TimedText document for that video content page and, thus, commits changes.

5.3.3.2 Retrieving and Playing Back SLNS Subtitles

Once the content page is viewed the Chisimba Wiki plugin loads the video file attached to the video content page as well as the TimedText document, if it exists. The display of Stokoe and HamNoSys was, again, handled differently to SignWriting.

The subtitle display area inside the JW FLV player was modified to cater for Stokoe and HamNoSys fonts. This modification uses HTML Iframes, in which the CSS font property is set to include both custom SLNS fonts. The display of Stokoe and HamNoSys subtitles then takes place automatically.

In order to cater for SignWriting notation images, the subtitle display area in the JW FLV player was modified to cater for HTML image tags. As mentioned previously, the information stored in the database for SignWriting subtitles is the SignWriting identifier from SWIS. In order to retrieve the SignWriting images from this identifier, it is necessary to pre-process the SignWriting identifier as it is parsed in from the TimedText document and replace it with the actual image.

The JW FLV player has a built-in parser to handle the parsing of annotation documents. The Dojo JavaScript framework is used to access the events and methods of the parser in order to retrieve subtitles as they are parsed in at the required time in the video. It is possible to use native JavaScript but the Dojo JavaScript framework provides ready made functionality to access the parser, thus, justifying its use. Once a subtitle is retrieved it is checked for a SignWriting identifier. If it is found to contain a SignWriting identifier, it is passed into SWIS to retrieve the SignWriting image. The image is then displayed in the HTML image tag in the modified subtitle display area. Figure 5.13 is a flow chart diagram illustrating the interaction process between the JW FLV player's parser, SWIS and Dojo JavaScript.

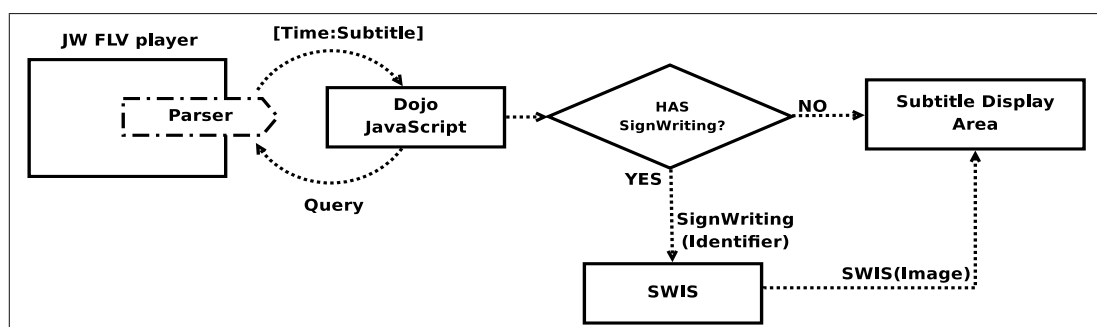


FIGURE 5.13: Flow diagram: Interaction process between JW FLV player, Dojo and SWIS.

At this point, the requirement to be able to annotate video with all three SLNSs has been met.

5.4 The User Interface

This section describes the user interface of the system required by this research. Section 5.4.1 describes the initial user interface before any refinement was carried out. A series of figures are provided to indicate the work flow of the system. Section 5.4.2 discusses the methodology used to refine the system and the changes made to the user interface to enhance its usability.

5.4.1 The Initial User Interface

The process of logging on to the system, uploading a video into a Wiki content page and annotating it involves traversing a series of screens. Following are the steps taken in this process and screen shots of the screens that are displayed to the user before any refinement was carried out.

1. The user logs on to the system online. The home page is displayed and is depicted in figure 5.14. The user is given one of two options: entering the login details into the login box provided on the home page and clicking the “Login” button; or clicking the provided link – Option 2 – which provides direct access to the video annotation functionality of the system. Therefore, the system satisfies the requirement of providing semi-authenticated access. The user is logged in.
2. The main page is displayed and is depicted in figure 5.15. This page displays information about various functionality provided by the system. It also provides a quick way for the user to upload a video into the system using the file upload field at the top of the page.
3. The “browse” button is clicked and displays the file manager which was discussed in a previous section and is depicted in figure 5.3. This provides the user with an interface to upload video files. The user clicks the “Select” button in the file manager which selects it for use.
4. The file upload field now contains a reference to the selected file as shown in figure 5.17. The side bar to the left of the page has been omitted from this and subsequent screen shot figures due to space constraints. The user then clicks on the “Save file” button which proceeds to the Wiki content description page that allows the user to specify additional details about the Wiki content page such as a description. This page is depicted in figure 5.16. Clicking the “Create page” button finalizes the creation of the Wiki video content page.

5. The Wiki video content page is loaded and is depicted in figure 5.18. This page contains the video player described in previous sections containing the video file that was attached as well as the page content which the user specified. Most importantly, it also contains controls to annotate the video. The user can use the controls to play back the video and the annotation controls to create annotations.

The screenshot shows the home page of the Video Annotation Wiki for South African Sign Language (VAWSASL). The page layout includes a search bar at the top right with a 'Go' button. The main title is 'Video Annotation Wiki for South African Sign Language'. On the left side, there are three boxes: 'Login' with fields for 'Username:' and 'Password:', a 'Remember me' checkbox, and a 'Login' button; 'Registration' with a 'Register' button; and 'Languages' with a 'Select language:' dropdown set to 'English'. The right side has three boxes: 'Skin' with a 'Select skin:' dropdown set to 'Refractions'; 'Other Sites' with the text 'insert other site links here'; and 'Join this project' with the text 'Insert information about joining this project here'. Below the 'Join this project' box is a 'W3C XHTML 1.0' logo. The central content area is titled 'Welcome to VAWSASL' and contains the following text: 'On this site, you may upload sign language videos and annotate them in SignWriting, Stokoe and/or HamNoSys. If you want to start uploading/annotating, you may follow one of the two following options:'. Below this is a box titled 'INSTRUCTIONS: Start Uploading / Annotating Videos' containing two options: 'Option 1. Register a username by using the Registration box to the right of this page and use that username to login to the main Wiki content page and upload/annotate videos with your username.' and 'Option 2. Click [HERE](#) to move directly to the main Wiki content page and upload/annotate videos anonymously.'. At the bottom of the central area, there is a watermark for the 'UNIVERSITY of the WESTERN CAPE' and a paragraph of text: 'The underlying CMS is Chisimba, developed at the University of the Western Cape. Chisimba is an advanced rapid application development platform and next generation framework with features similar to common proprietary systems. It is free software (open source) released under the GNU GPL, and available for download via cvs checkout from <http://cvs2.uwc.ac.za/>. Chisimba was developed based on several years of experience in software design and architecture at the University of the Western Cape and partner institutions using its predecessor KEWL.NextGen, and is under active development by a team of developers in 16 African higher education institutions.'

FIGURE 5.14: The home page of the system and instructions depicted to enter the website.

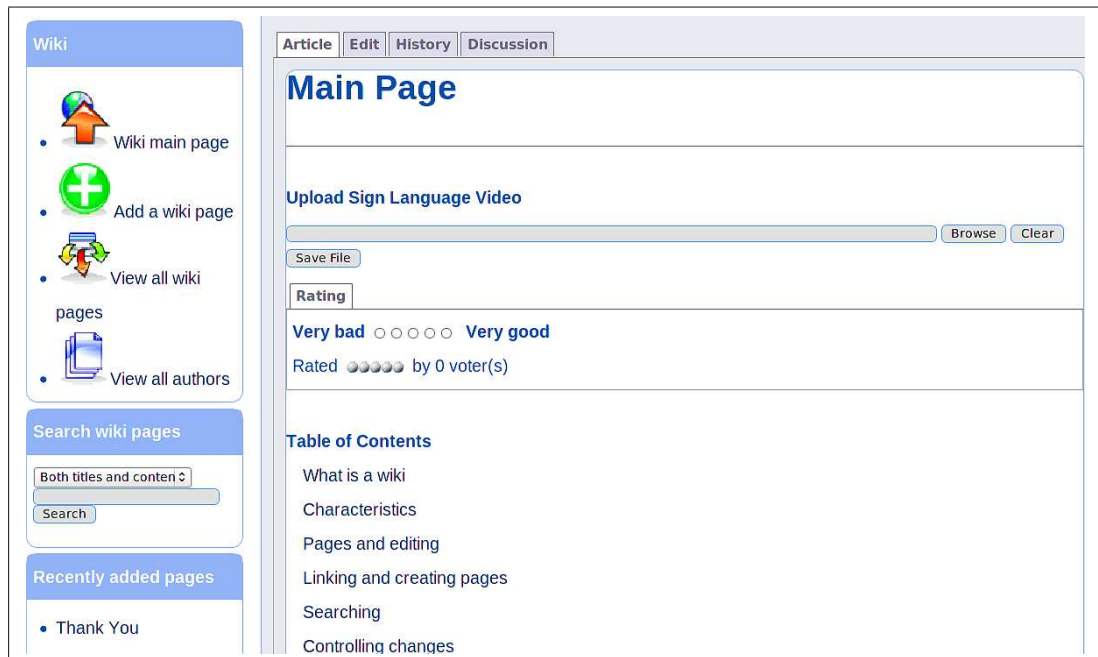


FIGURE 5.15: The main page containing the "browse" button used to upload the video files.

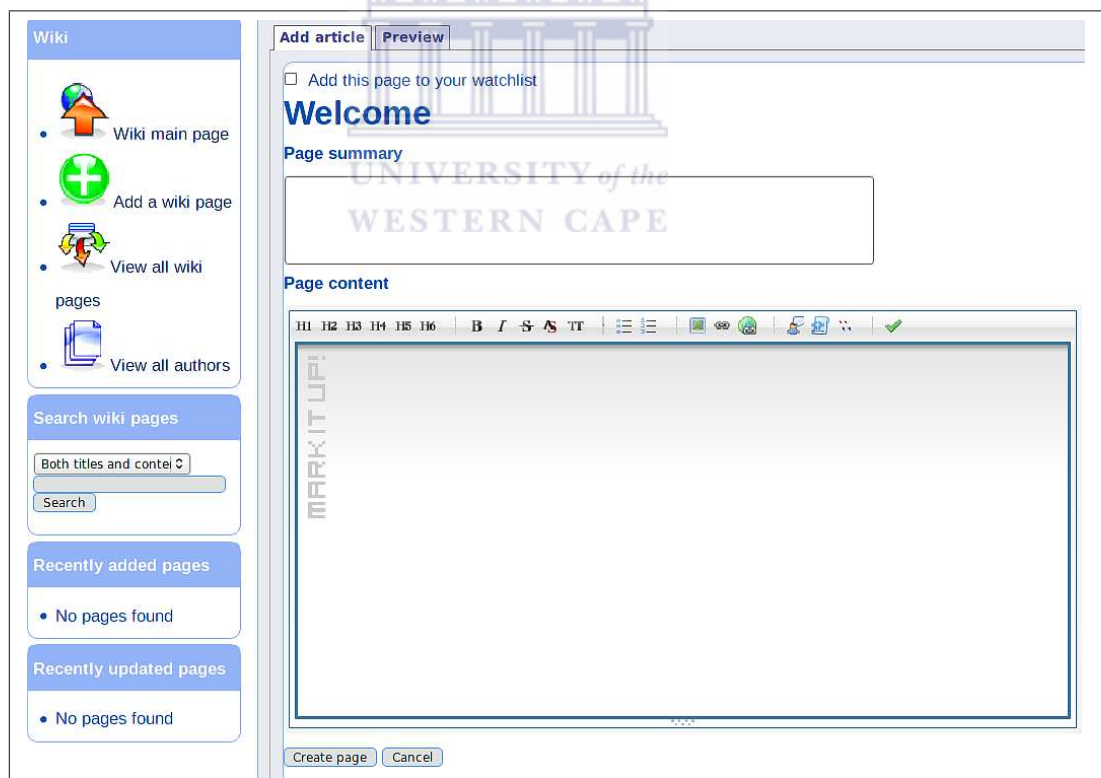


FIGURE 5.16: The Wiki content page used to insert page content and descriptions.



FIGURE 5.17: An example of a selected uploaded video file before the user saves the information.



FIGURE 5.18: Prototyping Model: The initial user interface of the system.

5.4.2 Refinement of the User Interface

After the functionality of the system was implemented, a prototyping model was used to further refine the user interface. The prototyping model is an effective method for the refinement of web-based applications [11]. Figure 5.19 depicts the prototyping model used in the refinement of the user interface of the system.

Two iterations of the model were carried out on the initial user interface. Each refinement iteration began by consulting with a focus group to highlight key usability issues on the user interface. These issues were then addressed by refining and re-designing the interface as required. Three pages – the Wiki video content page, Wiki content description page and the main page – received changes.

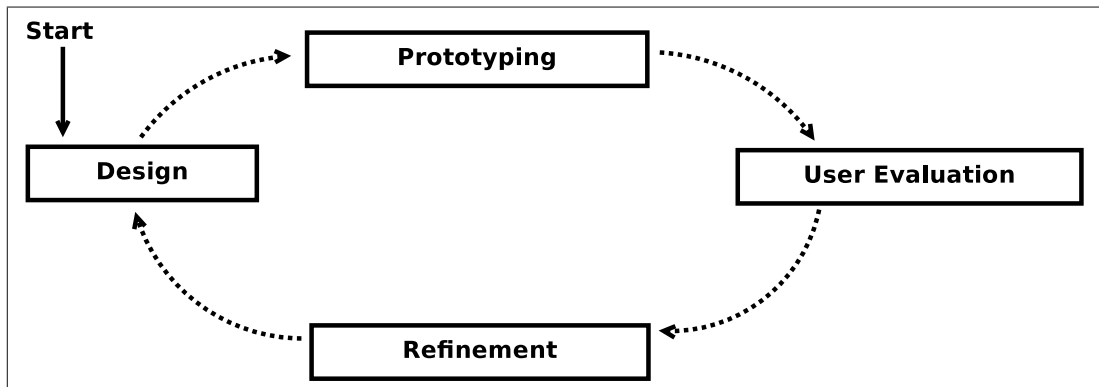


FIGURE 5.19: Prototyping Model: This model is used to refine the user interface of the system.

Tables 5.2, 5.3 and 5.4 summarize the key issues highlighted in the user interface of the Wiki video content page, the main page and the Wiki description page, respectively, in the first iteration and how they were addressed. Table 5.5 summarizes the same information for the second iteration and only applies to the Wiki content page, since only this page raised concerns during the focus group in the second iteration. Figures 5.21, 5.22, 5.23 and 5.20 depict the refined final user interfaces of the Wiki video content page, the main page, the Wiki description page and quick upload area, respectively.

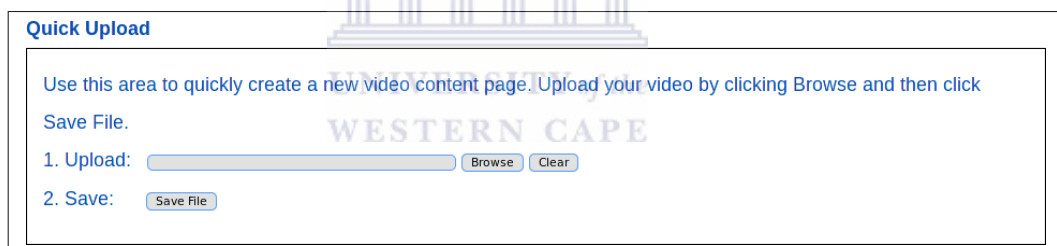


FIGURE 5.20: Prototyping Model: The second iteration of the quick upload area from the main page of the user interface.

Issue	Resolution
Create subtitle controls not marked	Clearly marked frame added around subtitle creation controls labelled “Create Subtitle”
Caption text box not marked	Label “Caption” added to name the text box
Constrained space between the caption text box, begin time and end time controls	Controls were spaced out
Function of the “Continue” button not clear	”Continue” button renamed to ”Add Subtitle”
Function and logic of the “Generate” check box not clear/straight forward	“Generate” check box removed and a button “Generate” was added instead with a description tag
Function of the SLNS add icons not clear	Icons removed and replaced with clear and descriptive links

TABLE 5.2: Key issues highlighted in the user interface of the video content page in the first iteration and resolution of these issues.

Issue	Resolution
Function of the video upload feature not clear	Clearly marked frame added around video upload feature area labelled “Quick Upload” and description of its function provided
File upload field too wide – not aesthetically pleasing	Width reduced to size
“Save file” button not visible	Space given between the button and the file upload field

TABLE 5.3: Key issues highlighted in the user interface of the main page in the first iteration and resolution of these issues.

Issue	Resolution
Function of the page not clear	Description provided at the top of the page
Difference between page summary and page content not clear	Description provided at the top of the page
Margins between text boxes and left side bar not aesthetically pleasing	Provided a margin

TABLE 5.4: Key issues highlighted in the user interface of the Wiki content description page in the first iteration and resolution of these issues.

Issue	Resolution
Position of the controls not intuitive (on top)	Controls were moved below the player
Seek bar not visible	Seek bar made visible
Position of the subtitles display area intrusive and not intuitive (on top)	Subtitles display area moved below player

TABLE 5.5: Key issues highlighted in the second iteration and resolution of these issues.

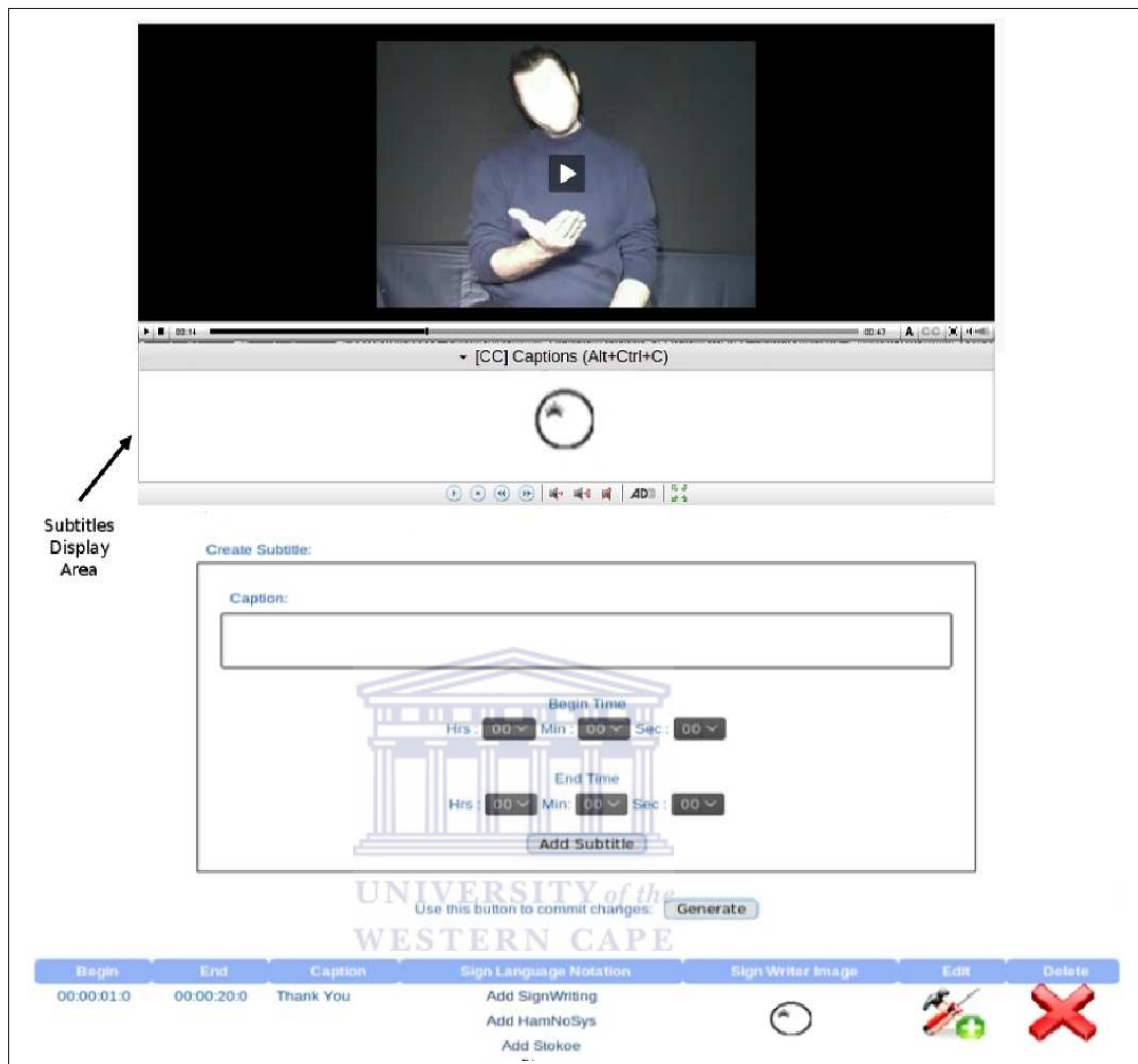


FIGURE 5.21: The final refined Wiki video content page user interface.

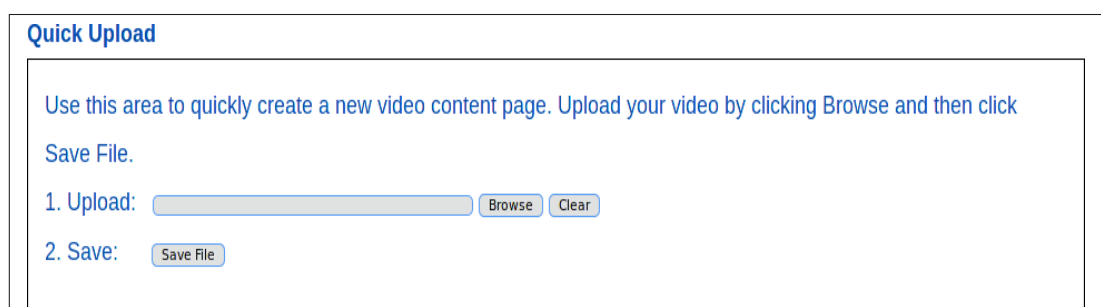


FIGURE 5.22: The final refined Wiki main page user interface showing only the refined quick upload area.

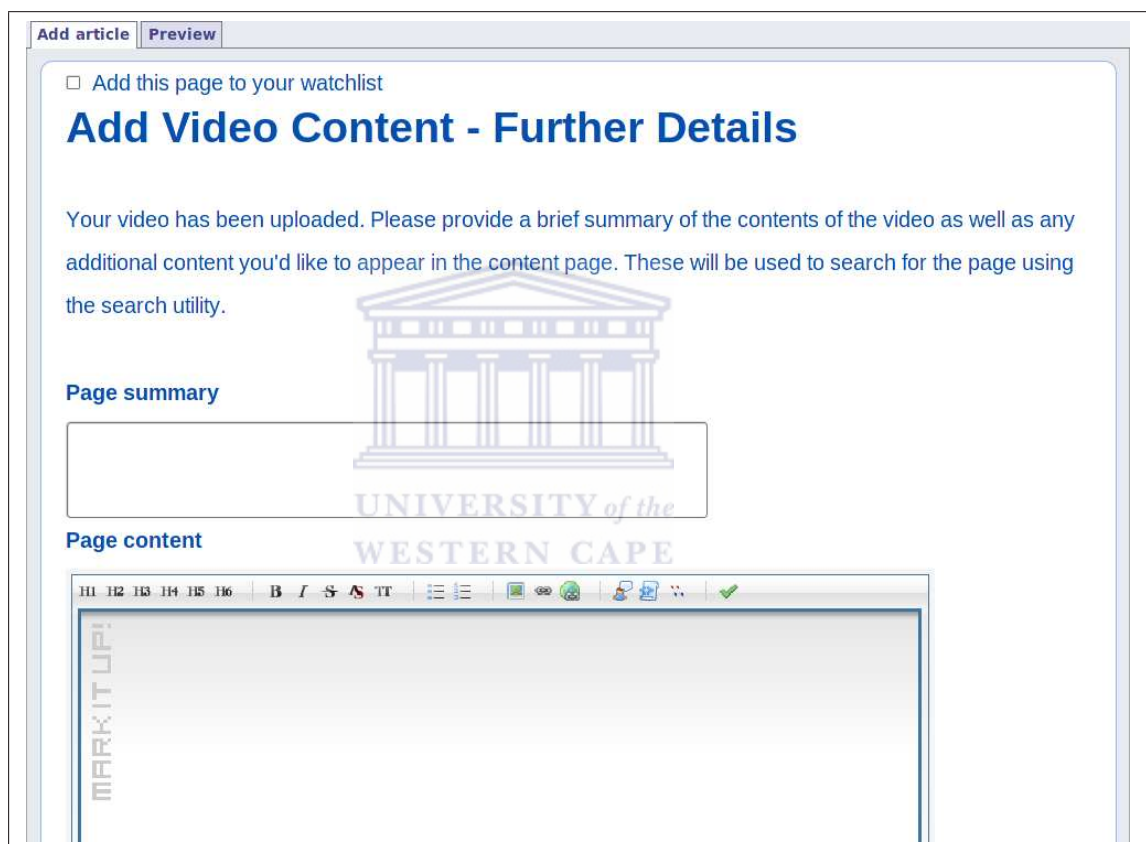


FIGURE 5.23: The final refined Wiki content description page user interface.

5.5 Conclusion

This chapter described the implementation of the SLNS video annotation system required by this research. The implementation of the basic required functionality was done separately to the refinement of the interface. The implementation of SLNS video annotation functionality took place in three steps: incorporating video into the Chisimba Wiki plugin content pages; implementation of video annotation support on to the video content; and incorporating SLNS support into the annotation process. The refinement of the interface made use of the prototyping model. Two iterations were carried out in which several usability issues were identified and addressed. The required system – a collaborative online sign language notations system video annotation knowledge management system – has been implemented. The next chapter addresses the final remaining objective by evaluating the performance and usability of the system implemented.



Chapter 6

System Testing and Analysis

In Chapter 1, it was mentioned as part of the requirements that the system for this research should have an acceptable usability and performance from the users' perspective. The usability testing evaluates the users' perspective of the user interface while the performance testing deals with the functionality and the response time of the system in providing such functionality. Therefore, similar to the previous chapter – Chapter 5 – a distinction is made in the testing strategy between the user interface and the underlying functionality.

This chapter discusses the testing carried out and performs an analysis of the derived results in order to evaluate the usability and performance of the system implemented by this research.

Section 6.1 deals with evaluation of the usability of the system. Section 6.2 discusses the evaluation of the performance of the system.

6.1 Usability Testing

The usability testing focussed on evaluating the usability of the main function of the system i.e. sign language video annotation. At this stage of the project, the focus is on functionality and the ease of use of this functionality from the users' perspective rather than on aesthetics and the “wow” factor [35]. In future, the focus may be set on providing a much more aesthetically pleasing interface.

The evaluation was conducted using ten subjects who were instructed to use the system to carry out the broad task of sign language video annotation. The details of the task are explained shortly. The subjects consisted of students and staff from the University

Subject	Gender	Age
1	Male	23
2	Male	23
3	Male	25
4	Male	27
5	Female	36
6	Male	26
7	Male	26
8	Male	27
9	Female	22
10	Female	20

TABLE 6.1: The profile of the test subjects of the usability evaluation.

of the Western Cape. Unfortunately, the researcher did not have access to people with a working knowledge of the three sign language notation systems. Therefore, it was decided to test the usability of the system in a general context with an intent to conduct testing with sign language notation system experts in the future. Nevertheless, students and staff at the University of the Western Cape, while not of the average academic capability, are of diverse backgrounds and are representative of the broader public in South Africa. Table 6.1 provides further details about the test subjects.

While not experts, all test subjects had some previous experience with computers. The evaluation was carried out inside a research computer laboratory. Each user evaluation was conducted individually whereby the user was requested to enter the laboratory and was seated in front of the computer. The broad task was broken down into a set of sub-tasks which the user was required to carry out. Each user was instructed to carry out each sub-task without any help or information from the researchers. This was meant to gauge the intuitiveness of the interface. After each task the user was asked to give the sub-task a score from 1 to 5 with 1 marked as extremely difficult and 5 marked as extremely easy. A score of 3 represents a score that is not easy but not difficult either and is therefore deemed the acceptable score level. Table 6.2 summarizes the labels given to each score. After all the sub-tasks were completed the user was asked to provide an overall rating of the system on the scoring scale mentioned previously. The sub-tasks are described in table 6.3.

Table 6.4 summarizes the score given to each of the sub-tasks by each of the users in the evaluation.

Score	Label
1	Extremely Difficult
2	Difficult
3	Acceptable
4	Easy
5	Extremely Easy

TABLE 6.2: The score scale used to measure the ease of use of the system’s interface.

Task Number	Task	Description
1	Login	The home page was loaded and displayed to the user. The user was provided with login details – a user name and password – and was instructed to login.
2	Content creation	The user was instructed to use the quick upload feature on the main page to upload a video and subsequently create a new Wiki page.
3	Annotate video	Once the user created the Wiki page with the attached video, the user was instructed to use the subtitle controls to insert three annotations into the video, one for each SLNS catered for. As previously mentioned, the test subjects had no knowledge of any of the three SLNSs. Emphasis was not on inserting subtitles that related to the video. It was sufficient to insert any SLNS. For HamNoSys and Stokoe, it sufficed to type in some text in the corresponding fonts. For SignWriting, it sufficed to create a SignWriting pictogram of any complexity.
4	Video player manipulation	The user was instructed to use the video player controls to playback the annotated video and view the annotations inserted previously.
5	View Wiki page	The user was instructed to navigate and find an existing content page that had been inserted by the researcher previously on the system. The user could either use the search feature or to use the “View All” link to display a list of all videos and navigate to the correct one.

TABLE 6.3: The sub-tasks used for evaluating the system.

Figure 6.1 is a graph of the average score for each sub-task of the usability evaluation over all ten users.

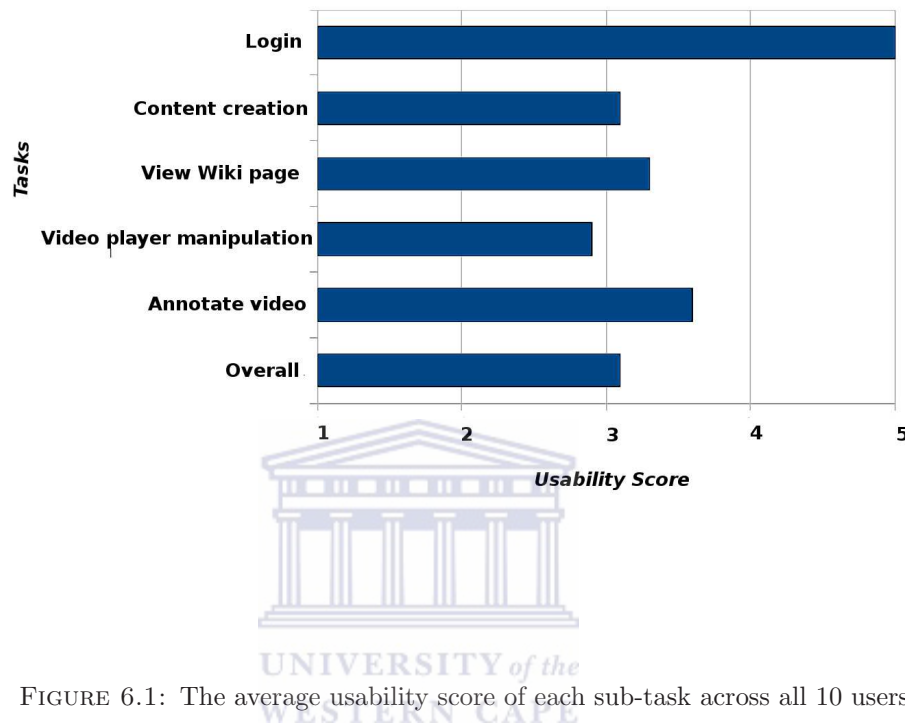


FIGURE 6.1: The average usability score of each sub-task across all 10 users.

Figure 6.1 shows that all sub-tasks except one – video player manipulation – scored above the acceptable usability score of 3 on the usability scale. However, the video player manipulation sub-task score was very close to the acceptable usability level, with an average score of 2.9.

Most sub-tasks scored between 3 and 4 on the usability scale, with only one sub-task – login – scoring in excess of 4 on the usability scale with a score of 5, which indicates that the users found this task extremely easy. The crucial sub-task of annotating video scored 3.6 which is above average.

As the experiment progressed, it was found that issues not highlighted in the two iterations of the implementation were highlighted by users. In carrying out the sub-task of content creation, some users indicated that the file manager was too complicated and that the select button was not clearly labelled. However most of the users used the file manager without any complaints.

Most users had no complaints about the *View Wiki Page* sub-task, but two users commented that the icons used to link the page were not appealing. With regards to the

Task	User 1	User 2	User 3	User 4	User 5	User 6	User 7	User 8	User 9	User 10	Average
Login	5	5	5	5	5	5	5	5	5	5	5.0
Content creation	4	4	3	3	3	3	5	1	4	1	3.1
View Wiki page	4	3	3	2	2	4	4	3	4	4	3.3
Video player manipulation	3	3	2	2	2	3	4	3	4	3	2.9
Annotate video	4	4	4	4	4	3	4	3	3	3	3.6
Overall	3	4	2	4	1	4	4	3	3	3	3.1

TABLE 6.4: Results of the usability evaluation of the system for all ten users.

sub-task of the video player manipulation, most users found the extra set of controls that were added to the video player to be confusing and cumbersome. This made it difficult to carry out this sub-task. Hence, the sub-task performed worse than other sub-tasks. Also, several comments were received from users about the sub-task of annotating video relating to the latency between creating annotations and being able to play them back. Users found this latency to be irritating.

Nevertheless, most sub-tasks scored above the required usability acceptance level of 3. The overall system score is depicted in figure 6.1 which indicates the average usability score over all users and all sub-tasks, a score of 3.1, which is above the acceptance level. Therefore, the requirement of achieving an acceptable usability with respect to ease of use from the users' perspective has been satisfied.

6.2 Performance Testing

Performance testing focuses on evaluating a quantitative metric, usually response time, to determine the speed of the the system. Performance testing involves two types of testing: load and stress testing. These two types of performance testing evaluate two different characteristics of the system. Load testing aims to determine the number of users that can be supported by the system with an acceptable response time under typical loads¹. An example of a typical load is 30 users logging on and carrying out tasks gradually. Stress testing aims to determine the number of users that can be supported by the system with an acceptable response time under extreme and unusual loads. An example of an extreme load is 30 users all logging on simultaneously and carrying out a very system-intensive task.

The two types of testing were carried out on the same set of tasks used in the usability testing and enumerated in table 6.3. Apache JMeter is a load and stress testing tool that was used for the performance testing of the system. It can simulate large numbers of virtual users logging onto a system and performing any pre-defined set of tasks. It is an open source tool that is freely available at the JMeter website [2]. It was used to measure the average response time of the system.

Sections 6.2.1 and 6.2.2 discuss the load testing and stress testing carried out and provide an analysis of the results thereof, respectively.

¹A load is a user logged into the system carrying out a task thereby consuming memory, processor and bandwidth resources.

6.2.1 Load Testing

The load testing carried out involved simulating an increasing number of users logging on to the system and carrying out the tasks mentioned in table 6.3. The users were logged on to the system sequentially with a 1 minute delay between each new user entrance. The average response time was determined by JMeter in each successive interval at which an extra user was added.

JMeter measures the response time of each request made to the web server. The response time of the web server for the request is measured by how fast the web server returns the content required. The average response time is computed over a specified number of requests.

It should be noted that the number of users logged on increased but also decreased gradually since users initiated earlier completed all sub-tasks earlier as well. Therefore, it was intended to measure the gradual and typical usage of the system in contrast with the stress testing which tested concurrent and extreme system usage.

Nielsen, Menasc and Hoxmeier [32, 48, 52] state that a response time greater than 8 seconds is unreasonable and unacceptable from the user's perspective. The number of users was increased until the average response time reached this threshold. Due to the limitations of computer hardware, reaching such a threshold at some point is unavoidable. In order to show that the software being tested is stable, the load testing was carried out on two different machines, one of much higher capability² than the other. Henceforth, we refer to the machine of higher capability as "the higher specification machine" and the other machine as "the lower specification machine" for ease of reference.

The load testing described was iterated a total of 6 times on each machine. It was hypothesized that the response times on the higher specification machine should generally be smaller – and, hence, faster – than those of the lower specification machine. Most importantly, it was expected that, if the software is stable, the number of users at which the response time exceeds the acceptable threshold on the lower specification machine should increase on the higher specification machine. This would show that the limitation is strictly on the hardware. It would also address the issue of whether or not the system can scale.

It was further expected that the test results between iterations would be very similar or, ideally, exactly the same, on each individual machine. This would indicate consistency in the experimental procedure.

²Processing power and RAM

The test was initially carried out by hosting the server on the lower specification machine which was selected to be a Sony VAIO laptop with a 1.8 GHz dual core Centrino processor and 2 GB of RAM. Thereafter, the same test was run on the higher specification machine which was selected to be a desktop computer with an Intel I3 Core processor and 4 GB of RAM.

The reader may refer to tables A.1 and A.2 in Appendix A for the complete listing of load testing results of all 6 iterations for the lower and higher specification machines. The result of this testing over all 6 iterations for the lower specification machine has been plotted in Figure A.1 in Appendix A. The result of this testing over all 6 iterations for the higher specification machine has been plotted in Figures A.2, A.3 and A.4 in Appendix A. The graph has been split into three figures due to space constraints. It should be noted that the scale on all of these graphs has been adjusted in order to make the small differences in response times between iterations observable. As was expected, the results acquired in the experiment for both the low and high specification machines are consistent and very similar across all 6 iterations. This indicates that the experimental procedure was conducted consistently.

Figure 6.2 is a graph of the average response time over all 6 iterations against the number of users simulated up to that time on the lower specification machine. Figures 6.3 and 6.4 are graphs of the average response time over all 6 iterations against the number of users simulated up to that time on the higher specification machine for users 1 to 41 and 42 to 68 respectively. Due to space constraints, the data has been provided in two graphs.

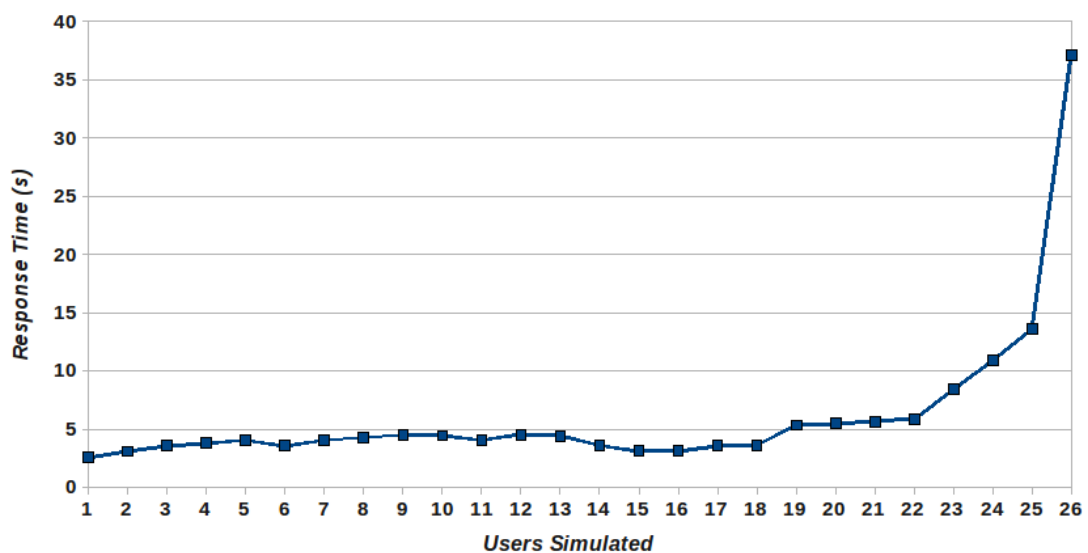


FIGURE 6.2: Results of load testing: response time for users 1 to 26 on the lower specification machine.

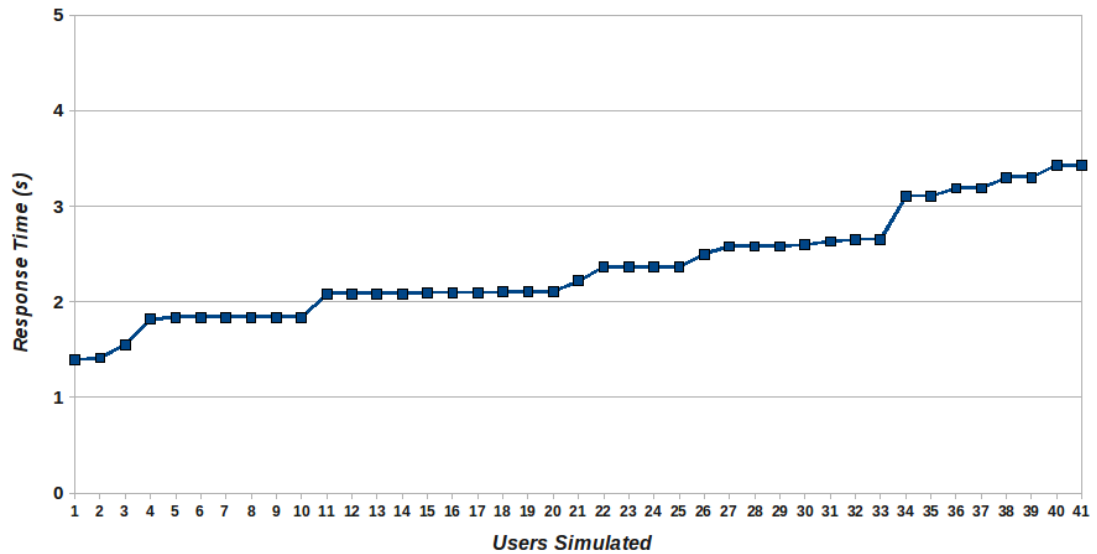


FIGURE 6.3: Results of load testing: response time for users 1 to 41 on the higher specification machine.

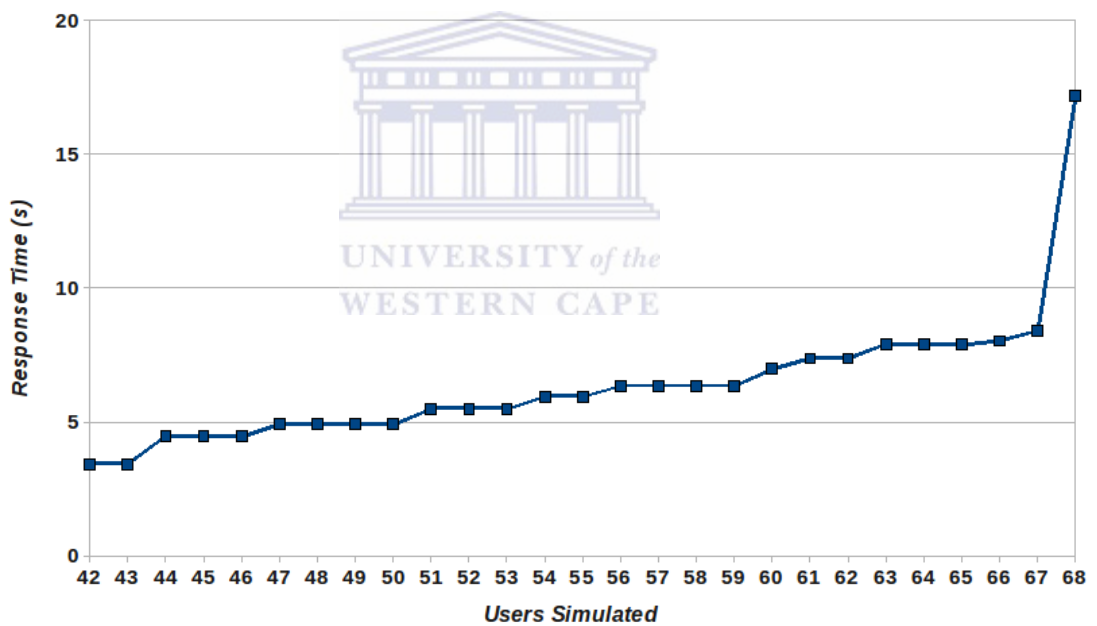


FIGURE 6.4: Results of load testing: response time for users 42 to 68 on the higher specification machine.

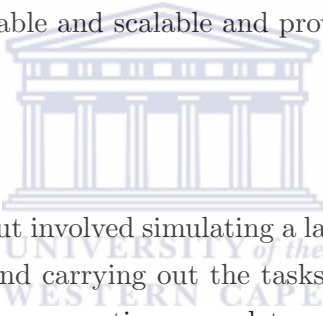
As seen in figure 6.2, the response time on the lower specification machine starts out at around 3 seconds and lightly fluctuates between 3 seconds and 5 seconds between the first and eighteenth user simulation initiated. At 19 users, the response time starts to increase until it reaches the threshold 8 second mark at 23 users. It is important to note that the server continued to respond even past this point – the 23 user mark – although with a very large response time. At 26 users, the response time reached 35 seconds.

As seen in figure 6.3, the response time for the higher specification machine starts out at around 1 second, which is far less than the starting response time of the lower specification machine in line with the hypothesis. The response time fluctuates between around 1 second and 2 seconds between the 1 and 23 user mark. At the 23 user mark – the mark at which the lower specification machine breached the acceptable threshold – the response time on the higher specification machine is around 2 seconds which is even less than the starting response time on the lower specification machine. This is in line with the hypothesis.

Generally, between the 1 and 43 user mark, the response time gradually increases from around 1 second to around 3 seconds. As seen in figure 6.4, between the 44 and 66 user mark the response time begins to increase at a greater rate from 4 seconds to 7 seconds. At the 67 user mark, the response time breaches the acceptable threshold. The load that is supported on the higher specification machine is far greater than that of the lower specification machine, in line with the initial hypothesis.

Therefore, the system is stable and scalable and provides an acceptable load capacity.

6.2.2 Stress Testing



The stress testing carried out involved simulating a large number of users simultaneously logging on to the system and carrying out the tasks mentioned in table 6.3. As in the load testing, the average response time was determined by JMeter. The aim was to determine the highest number of users that the system could support concurrently – the stress capacity – with an acceptable response time. Therefore, the number of concurrent users was varied and the average response time recorded for that number of users until the stress capacity was determined.

The stress testing was only carried out on the higher specification machine since the scalability of the system based on the underlying hardware has already been shown. A method was devised to determine the stress capacity without the need to begin at 2 concurrent users and gradually increase up to the stress capacity. Instead, following on the results of the load testing, an initial estimate was made as to the number of users that could possibly be supported. The load capacity on the higher specification machine was found to be 67 users.

Since stress testing is more strenuous and rigorous than load testing, it was decided to start out at 60 concurrent users. This value was then increased or decreased by varied increments depending on the proximity of the measured response time to the acceptable threshold. Therefore, if the response time measured was higher than the threshold, a

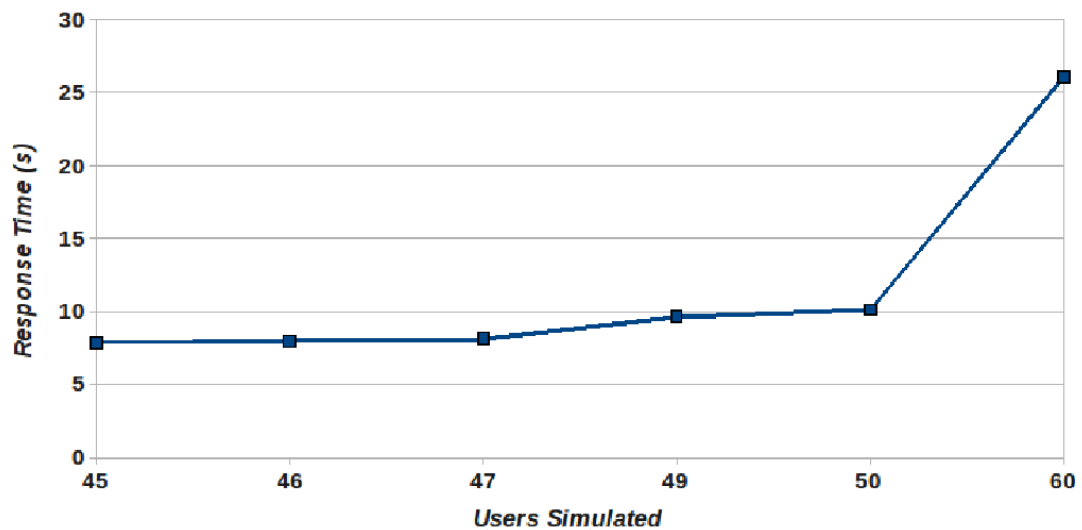


FIGURE 6.5: Results of stress testing; response time at varied number of concurrent users on the higher specification machine. The system is below the stress capacity at 46 users but exceeds it at 47 users.

decrement was carried out whereas an increment was carried out if the response time was lower than the threshold. The closer the response time was to the threshold, the smaller the increment or decrement.

Table 6.5 summarizes the number of users simulated and the average response time for that number of users and this information is plotted in figure 6.5. After the initial measurement at 60 users, the response time was found to be quite high and was reduced by 10 to a total of 50 users. The subsequent response time was found to be higher than but closer to the threshold and the number of users was reduced by 1 to 49 users. The response time was still found to be considerably higher than the 8 second threshold and the number of users was reduced to 45. The resulting response time was lower than but very close to the threshold. Therefore, the number of users was increased by 1 for two runs – 46 and 47 users. The response time for 46 users was just below the threshold

Users	Average Response Time (s)
60	26.04
50	10.11
49	9.64
47	8.13
46	7.97
45	7.85

TABLE 6.5: Results of the stress testing of the system.

while for 47 users it exceeded it by a small amount. The experiment was concluded at this point.

The system supports up to 46 concurrent users on the higher specification machine. It has been shown that better hardware can provide better and increased performance. Currently in South Africa, there are few sign language experts that have the knowledge of the sign language notation systems. Therefore, the system is not expected to cater for a large number of users at the time of its deployment. However, the hardware can be improved to cater for more users if it is required as time goes by. The system has an acceptable performance.

6.3 Conclusion

This chapter evaluated the system implemented by this research. The system was required to have an acceptable usability and performance. It was shown that the ease of use of the system exceeds the acceptable usability level. The performance of the system was evaluated by determining its load and stress capacity. It was shown that the system itself is stable and scalable and the restriction is only on the underlying hardware hosting the system. Therefore, the performance of the system is acceptable. On a pre-defined machine, it was shown to have a load capacity of 67 users and a stress capacity of 46 users for a pre-defined set of tasks. It was argued that, given the system is scalable, the hardware can be improved to cater for more users if it is required as time goes by. The final requirement set out in Chapter 1 has been satisfied.

Chapter 7

Conclusion

This thesis has made an important contribution to the South African sign language research (SASL) group at the University of the Western Cape. It was stated in Chapter 1 that very little documentation exists about South African sign language. Such documentation is crucial to the development of the machine translation system currently being developed by the SASL group. It is required in the training of classifiers to recognize SASL from a video sequence, to serve as templates for the rendering of correct SASL and as an informative source to be able carry out linguistic translations between SASL and English. Preceding this thesis, it was shown that no online collaborative video annotation knowledge management systems that, additionally, catered for all three sign language notation systems existed.

This thesis has produced a system that provides the ability to collect and pool information from the community and build a much-required SASL knowledge base. A thorough evaluation of various technologies was carried out in a bid to select the most advantageous, usable and accessible technology on which to build this system. The system provides the ability to annotate videos with sign language notation systems. The most important part of this is that all three major sign language notations systems – HamNoSys, SignWriting and Stokoe – are catered for. This provides the capability of pooling information from many different expert sources and provides flexibility. All the requirements set out in Chapter 1 have been satisfied.

Two sets of directions are provided for future research.

7.1 User Interface Refinement

It was shown in Chapter 5 that the usability of the interface meets an acceptable level from the users' perspective. At this stage, the focus was on providing usable functionality rather than on the aesthetics of the system. However, areas of potential improvement were highlighted in the usability testing. A few users suggested that the file manager and the video player controls be simplified. Others suggested that the icons on some pages be re-designed for greater aesthetic appeal. It is recommended that the user interface, in general, be further refined using the methodology provided in Chapter 5 with several more iterations. It is also recommended that usability testing be done using sign language experts and refinement of the interface accordingly if necessary.

7.2 Mobile Support

The system developed by this research does not specifically cater for mobile phone browsers. The Internet is increasingly moving towards mobile phones. It is recommended that future refinement of the system cater for mobile phone browsers. This will enhance accessibility, mobility and ease-of-use. This is hoped to encourage greater use of the system and speed up the knowledge building process.



7.3 Concluding Remarks

The system developed in this research can serve as a crucial achievement for the SASL research group. It is hoped that it serves as a useful tool that leads up to the creation of the final complete machine translation system. All the requirements set out in Chapter 1 have been satisfied.

Appendix A

Appendix

A.1 Performance Testing

TABLE A.1: Results of the load tests for the higher specification machine, measured in seconds.

Response time (s) of iteration number							
Users	1	2	3	4	5	6	Average
1	1.415	1.350	1.390	1.395	1.398	1.411	1.389
2	1.435	1.370	1.410	1.416	1.419	1.432	1.409
3	1.571	1.500	1.550	1.550	1.553	1.568	1.544
4	1.844	1.760	1.820	1.819	1.822	1.840	1.812
5	1.864	1.780	1.840	1.840	1.843	1.861	1.833
6	1.864	1.780	1.840	1.840	1.843	1.861	1.833
7	1.864	1.780	1.840	1.840	1.843	1.861	1.833
8	1.864	1.780	1.840	1.840	1.843	1.861	1.833
9	1.864	1.780	1.840	1.840	1.843	1.861	1.833
10	1.864	1.780	1.840	1.840	1.843	1.861	1.833
11	2.116	2.020	2.080	2.088	2.092	2.112	2.078
12	2.116	2.020	2.080	2.088	2.092	2.112	2.078
13	2.116	2.020	2.080	2.088	2.092	2.112	2.078
14	2.116	2.020	2.080	2.088	2.092	2.112	2.078
15	2.126	2.030	2.090	2.098	2.102	2.122	2.088
16	2.126	2.030	2.090	2.098	2.102	2.122	2.088
17	2.126	2.030	2.090	2.098	2.102	2.122	2.088

Continued on next page

Table A.1 – continued from previous page

Users	1	2	3	4	5	6	Average
18	2.137	2.040	2.100	2.108	2.112	2.133	2.099
19	2.137	2.040	2.100	2.108	2.112	2.133	2.099
20	2.137	2.040	2.100	2.108	2.112	2.133	2.099
21	2.252	2.150	2.220	2.222	2.226	2.248	2.213
22	2.399	2.290	2.360	2.367	2.371	2.394	2.356
23	2.399	2.290	2.360	2.367	2.371	2.394	2.356
24	2.399	2.290	2.360	2.367	2.371	2.394	2.356
25	2.399	2.290	2.360	2.367	2.371	2.394	2.356
26	2.535	2.420	2.500	2.501	2.506	2.530	2.491
27	2.619	2.500	2.580	2.584	2.589	2.614	2.573
28	2.619	2.500	2.580	2.584	2.589	2.614	2.573
29	2.619	2.500	2.580	2.584	2.589	2.614	2.573
30	2.640	2.520	2.600	2.604	2.609	2.635	2.594
31	2.671	2.550	2.630	2.635	2.640	2.666	2.624
32	2.692	2.570	2.650	2.656	2.661	2.687	2.645
33	2.692	2.570	2.650	2.656	2.661	2.687	2.645
34	3.153	3.010	3.100	3.111	3.117	3.147	3.097
35	3.153	3.010	3.100	3.111	3.117	3.147	3.097
36	3.237	3.090	3.190	3.193	3.200	3.230	3.181
37	3.237	3.090	3.190	3.193	3.200	3.230	3.181
38	3.352	3.200	3.300	3.307	3.313	3.345	3.293
39	3.352	3.200	3.300	3.307	3.313	3.345	3.293
40	3.478	3.320	3.420	3.431	3.438	3.471	3.416
41	3.478	3.320	3.420	3.431	3.438	3.471	3.416
42	3.478	3.320	3.420	3.431	3.438	3.471	3.416
43	3.478	3.320	3.420	3.431	3.438	3.471	3.416
44	4.525	4.320	4.460	4.465	4.473	4.516	4.447
45	4.525	4.320	4.460	4.465	4.473	4.516	4.447
46	4.525	4.320	4.460	4.465	4.473	4.516	4.447
47	4.986	4.760	4.910	4.919	4.929	4.976	4.899
48	4.986	4.760	4.910	4.919	4.929	4.976	4.899
49	4.986	4.760	4.910	4.919	4.929	4.976	4.899
50	4.986	4.760	4.910	4.919	4.929	4.976	4.899
51	5.572	5.320	5.490	5.498	5.509	5.562	5.476
Continued on next page							

Table A.1 – continued from previous page

Users	1	2	3	4	5	6	Average
52	5.572	5.320	5.490	5.498	5.509	5.562	5.476
53	5.572	5.320	5.490	5.498	5.509	5.562	5.476
54	6.033	5.760	5.940	5.953	5.964	6.022	5.928
55	6.033	5.760	5.940	5.953	5.964	6.022	5.928
56	6.431	6.140	6.330	6.345	6.358	6.419	6.318
57	6.431	6.140	6.330	6.345	6.358	6.419	6.318
58	6.431	6.140	6.330	6.345	6.358	6.419	6.318
59	6.431	6.140	6.330	6.345	6.358	6.419	6.318
60	7.081	6.760	6.970	6.986	7.000	7.067	6.957
61	7.479	7.140	7.360	7.379	7.393	7.465	7.347
62	7.479	7.140	7.360	7.379	7.393	7.465	7.347
63	8.013	7.650	7.890	7.906	7.921	7.998	7.873
64	8.013	7.650	7.890	7.906	7.921	7.998	7.873
65	8.013	7.650	7.890	7.906	7.921	7.998	7.873
66	8.149	7.780	8.020	8.040	8.056	8.134	8.006
67	8.526	8.140	8.400	8.412	8.429	8.510	8.378
68	17.440	16.650	17.170	17.207	17.240	17.407	17.135

UNIVERSITY of the
WESTERN CAPE

TABLE A.2: Results of the load tests for the lower specification machine, measured in seconds.

Response time (s) of iteration number							
Users	1	2	3	4	5	6	Average
1	2.599	2.602	2.589	2.603	2.580	2.589	2.590
2	3.192	3.192	3.040	3.192	3.030	3.192	3.140
3	3.666	3.666	3.492	3.666	3.480	3.666	3.610
4	3.908	3.908	3.723	3.908	3.710	3.908	3.840
5	4.151	4.151	3.954	4.151	3.940	4.151	4.080
6	3.677	3.677	3.502	3.677	3.490	3.677	3.620
7	4.151	4.151	3.954	4.151	3.940	4.151	4.080
8	4.393	4.393	4.184	4.393	4.170	4.393	4.320
9	4.625	4.625	4.405	4.625	4.390	4.625	4.550
10	4.572	4.572	4.355	4.572	4.340	4.572	4.510

Continued on next page

Table A.2 – continued from previous page

Users	1	2	3	4	5	6	Average
11	4.161	4.161	3.964	4.161	3.950	4.161	4.090
12	4.656	4.656	4.435	4.656	4.420	4.656	4.580
13	4.540	4.540	4.325	4.540	4.310	4.540	4.470
14	3.698	3.698	3.522	3.698	3.510	3.698	3.640
15	3.224	3.224	3.071	3.224	3.060	3.224	3.170
16	3.224	3.224	3.071	3.224	3.060	3.224	3.170
17	3.666	3.666	3.492	3.666	3.480	3.666	3.610
18	3.677	3.677	3.502	3.677	3.490	3.677	3.620
19	5.467	5.467	5.208	5.467	5.190	5.467	5.380
20	5.604	5.604	5.338	5.604	5.320	5.604	5.510
21	5.805	5.805	5.529	5.805	5.510	5.805	5.710
22	6.026	6.026	5.740	6.026	5.720	6.026	5.930
23	8.628	8.628	8.218	8.628	8.190	8.628	8.490
24	11.135	11.135	10.606	11.135	10.570	11.135	10.950
25	13.874	13.874	13.215	13.874	13.170	13.874	13.650
26	37.693	37.693	35.904	37.693	35.780	37.693	37.080

UNIVERSITY of the
WESTERN CAPE

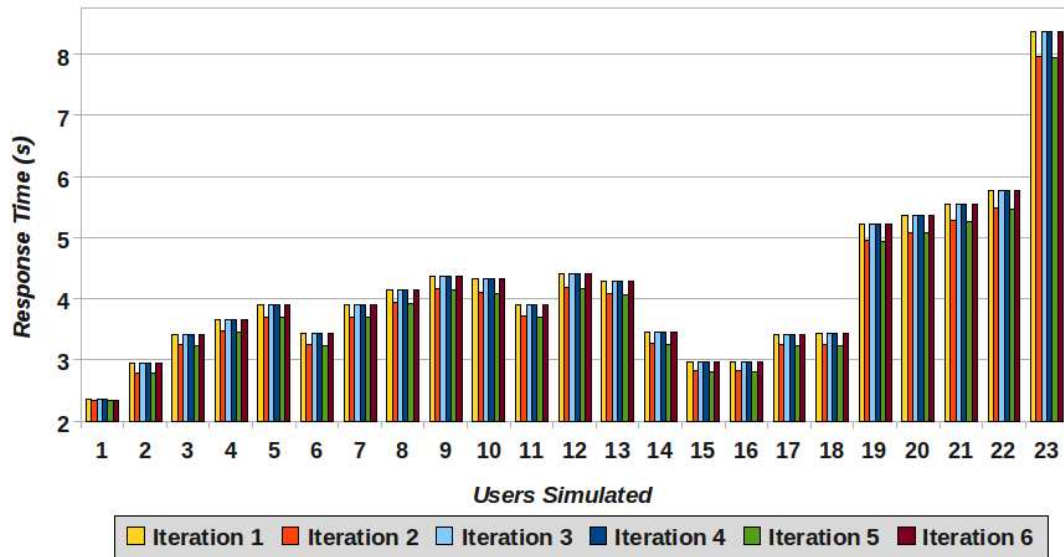


FIGURE A.1: Results of load testing: comparison in response times across the six test iterations for users 1 to 23 on the lower specification machine. The response times across the six iterations are seen to be very similar.

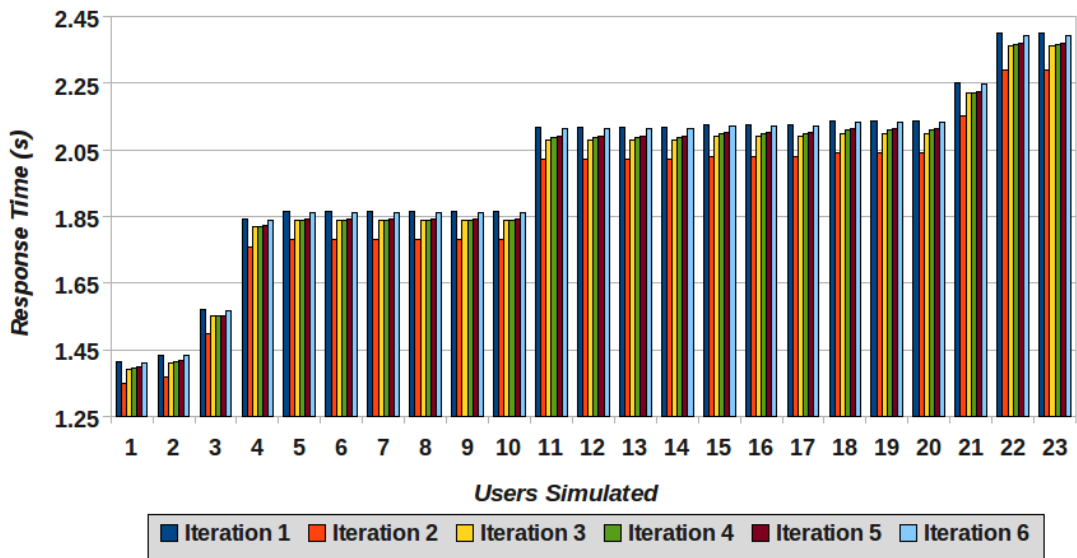


FIGURE A.2: Results of load testing: comparison in response times across the six test iterations for users 1 to 23 on the high specification machine. The response times across the six iterations are seen to be very similar.

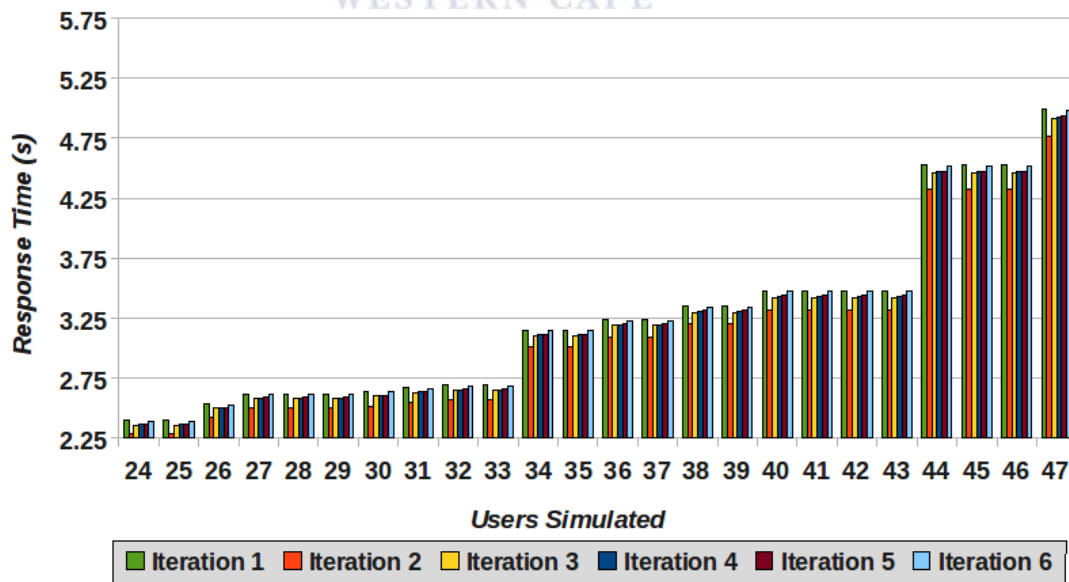
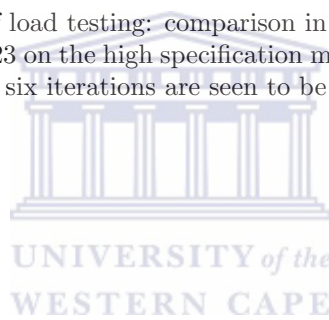


FIGURE A.3: Results of load testing: comparison in response times across the six test iterations for users 24 to 47 on the high specification machine. The response times across the six iterations are seen to be very similar.

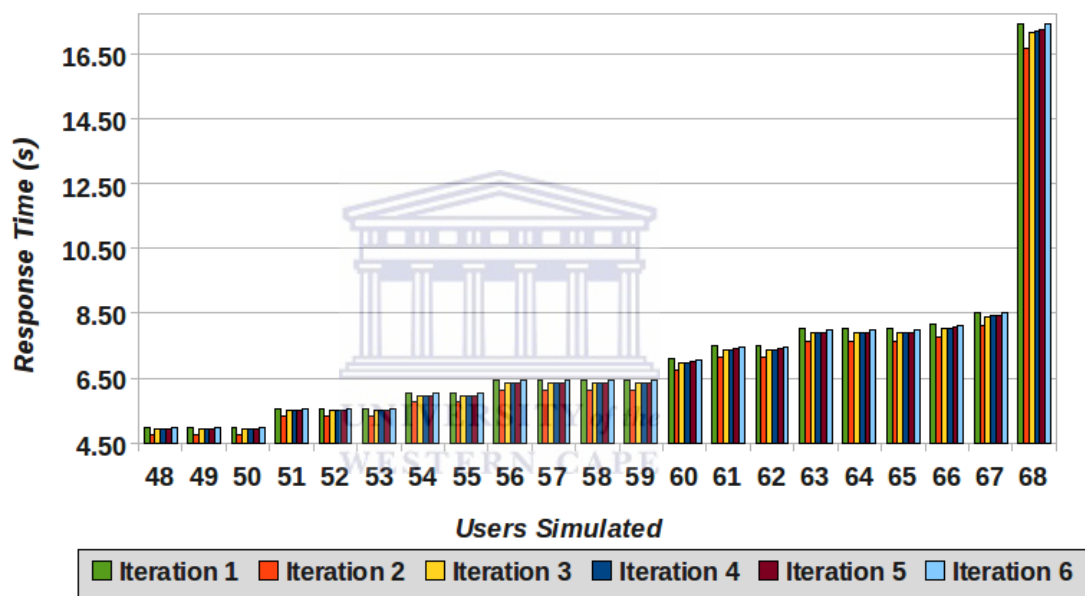


FIGURE A.4: Results of load testing: comparison in response times across the six test iterations for users 48 to 68 on the high specification machine. The response times across the six iterations are seen to be very similar.

Bibliography

- [1] C. Allen. (2010) WFD: World Federation of the Deaf. [Online]. Available: <http://www.wfdeaf.org/pdf/WFD%20Newsletter%20June%202009.pdf>
- [2] Apache Software Foundation. (2010) Apache JMeter official website. [Online]. Available: http://jakarta.apache.org/site/downloads/downloads_jmeter.cgi
- [3] AVOIR. (2010) African Virtual Open Initiative and Resources. [Online]. Available: <http://avoir.uwc.ac.za/>
- [4] D. Bargeron, A. Gupta, J. Grudin, and E. Sanocki, “Annotations for streaming video on the web: system design and usage studies,” *Computer Networking*, vol. 31, no. 11–16, pp. 1139–1153, 1999.
- [5] P. Bottoni, R. Civica, S. Levialdi, L. Orso, E. Panizzi, and R. Trinchese, “MAD-COW: a multimedia digital annotation system,” *AVI '04: Proceedings of the working conference on Advanced Visual Interfaces*, pp. 55–62, 2004.
- [6] H. Brugman, O. Crasborn, and A. Russel, “Collaborative annotation of sign language data with peer-to-peer technology,” *W3C*, pp. 1–4, Mar 2004, department of linguistics, University of Nijmegen.
- [7] M. Caldognetto, “Multimodal score: an ANVIL-based annotation scheme for multimodal audio-video analysis,” *Proceedings of LREC 2004 Workshop on Multimodal Corpora, Models of Human Behaviour for the Specification and Evaluation of Multimodal Input and Output Interfaces*, pp. 29–33, 2004. [Online]. Available: http://www.istc.cnr.it/doc/72a_20050607152633t_em-LREC2004.pdf
- [8] Y. Chen and Y. Wang, “The emerging MVC standard for 3D video services,” *EURASIP Journal on Advances in Signal Processing*, vol. 2009, no. 1, pp. 1–13, 2009. [Online]. Available: <http://dx.doi.org/10.1155/2009/786015>
- [9] Chisimba. (2010) Chisimba official website. [Online]. Available: <http://www.chisimba.com/>

- [10] T. A. Chung, "Dataset acquisition, accessibility, annotation, e-research technologies (DART) project: a case study using an X-ray diffractometer," *International Journal on Digital Libraries*, vol. 7, no. 1, pp. 53–55, 2007.
- [11] C. J. Costa and M. Aparicio, "Developing small web-based systems," *Lisbon University Institute*, pp. 1–6, 2010. [Online]. Available: www.wseas.us/e-library/conferences/austria2004/papers/482-153.doc
- [12] Cynapse. (2010) Cyn.in official website. [Online]. Available: <http://www.cynapse.com/cynin>
- [13] T. Dasgupta, "A multilingual multimedia Indian sign language dictionary tool," *Proceedings of the 6th Workshop on Asian Language Resources*, 2008. [Online]. Available: <http://www.aclweb.org/anthology/I/I08/I08-7008.pdf>
- [14] Deaf International. (2010) Deaf International official website. [Online]. Available: <http://www.deafinternational.org/DI/Home.html>
- [15] Deafnet. (2009) Deafnet official website. [Online]. Available: <http://www.deafnet.co.za/institute/1faq.html>
- [16] DeafSA. (2010) Deafsa official website. [Online]. Available: <http://www.deafsa.co.za/index-5.html>
- [17] P. Dreuw, "Appearance-based gesture recognition," Master's thesis, Rheinisch-Westfälische Technische Hochschule Aachen, Lehrstuhl für Informatik VI, 2005.
- [18] Drupal. (2010) Drupal official website. [Online]. Available: <http://drupal.org/>
- [19] DSpace Foundation. (2010) DSpace official website. [Online]. Available: <http://www.dspace.org/>
- [20] L. Dybkjær and N. O. Berman, "Data annotation schemes and tools for natural interactivity," *Natural Interactive Systems Laboratory*, pp. 1–4, 2002.
- [21] U. M. Erdem and S. Sclaroff, "Automatic detection of relevant head gestures in American sign language communication," *Proceedings of the International Conference on Pattern Recognition*, pp. 10 460–10 463, 2002.
- [22] T. Ernst, "The British sign language variant of Stokoe notation: report on a type-design project," *Sign Language Studies*, vol. 3, no. 3, pp. 341–370, 2003.
- [23] J. Fourie, "The design of a generic signing avatar animation system," Master's thesis, Department of Mathematical Sciences (Computer Science), University of Stellenbosch, Private Bag X1, 7602 Matieland, South Africa, 2006.

- [24] M. Glaser, "Telecommunications bridging between deaf and hearing users in South Africa," *Proceedings of the Conference and Workshop on Assistive Technologies for People with Vision and Hearing Impairments*, pp. 255–256, 2004.
- [25] A. Gross and I. A. Gross. (2010) Word-warp official website. [Online]. Available: <http://language.home.sprynet.com/langdex/wordwarp.htm>
- [26] T. Hanke and H. Popescu, "Interface definitions, ViSiCAST deliverable D5-1," *Institute of German Sign Language and Communication of the Deaf*, 2002.
- [27] B. L. Harrison and R. M. Baecker, "Designing video annotation and analysis systems," *In Proceedings Graphics Interfaces '92*, pp. 157–166, 1992.
- [28] H. Hasan and C. C. Pfaff, "Overcoming organisational resistance to using Wiki technology for knowledge management," *Proceedings of 10th Pacific Asia Conference on Information Systems*, pp. 926–935, 2006.
- [29] B. Haslhofer, W. Jochum, R. King, C. Sadilek, and K. Schellner, "The LEMO annotation framework: weaving multimedia annotations with the web," *International Journal on Digital Libraries*, vol. 10, no. 1, pp. 15–32, 2009.
- [30] B. Hellwig, "E-Lan – linguistic annotator," 2010. [Online]. Available: <http://www.mpi.nl/corpus/manuals/manual-elan.pdf>
- [31] B. Hellwig, "Multimedia annotation with E-Lan," 2010. [Online]. Available: <http://www.mpi.nl/corpus/a4guides/a4-guide-elan.pdf>
- [32] J. A. Hoxmeier and C. D. Manager, "System response time and user satisfaction: an experimental study of browser-based applications," *Proceedings of the Association of Information Systems Americas Conference*, pp. 10–13, 2000.
- [33] A. Hussey and D. Carrington, "Comparing two user interface architectures: MVC and PAC," *Formal Aspects of the Human Computer Interface*, pp. 3–21, 1996.
- [34] M. Jokinen, "Deaf in developing countries," 2009. [Online]. Available: http://www.wfdeaf.org/pdf/fact_deafdevelop.pdf
- [35] J. Kaasalainen, "User interface design and usability testing of a podcast interface," *Usability Testing*, 2007.
- [36] S. Kelsall, "Movement and location notation for American sign language," 2006. [Online]. Available: http://www.swarthmore.edu/SocSci/Linguistics/Papers/2006/kelsall_sarah.pdf
- [37] A. Kendon, "Sign languages of Aboriginal Australia: cultural, semiotic, and communicative perspectives," *Cambridge University Press*, 1988.

- [38] M. Kipp, "Spatiotemporal coding in Anvil," *Embodied Agents Research Group*, pp. 1–4, 2008.
- [39] M. Kipp. (2009) Anvil official website. [Online]. Available: <http://www.anvil-software.de/>
- [40] M. Kipp, "Anvil – a generic annotation tool for multimodal dialogue," *Proceedings of the 7th European Conference on Speech Communication and Technology (Eurospeech)*, pp. 1367–1370, 2001. [Online]. Available: http://www.dfki.de/~kipp/public_archive/kipp2001-eurospeech.pdf
- [41] J. G. Kyle and B. Woll, "The study of deaf people and their language," *Cambridge University Press*, pp. 328–340, 1988.
- [42] S. K. Liddell, "An investigation into the syntactic structure of American sign language," *Sign Language Studies*, 1977.
- [43] S. K. Liddell and J. Robert, "American sign language: The phonological base." *Sign Language Studies*, 1988.
- [44] J. Martin, "A linguistic comparison, two notation systems for sign language," *Western Washington University*, pp. 1–33, 2003.
- [45] J. Martin, "Writing and signed languages," Master's thesis, University of South Carolina, 2007.
- [46] MediaWiki Foundation. (2010) Mediawiki extensions page. [Online]. Available: http://www.mediawiki.org/wiki/Developer_hub#Extending_MediaWiki
- [47] MediaWiki Foundation. (2010) Mediawiki official website. [Online]. Available: <http://www.mediawiki.org/wiki/>
- [48] D. A. Menasc, "Load testing of web sites," *IEEE Internet Computing*, vol. 6, pp. 70–74, 2002.
- [49] F. Michael and B. Annelies, "Sign description: how geometry and graphing serve linguistic issues," *Theoretical Issues in Sign Language Research*, 2006.
- [50] S. Morrissey and A. Way, "Data-driven MT systems for improved sign language translation," Master's thesis, Dublin City University, School of Computing, 2008.
- [51] C. Neidle, "SignStream: a database tool for research on visual-gestural language," *Sign Language & Linguistics*, vol. 1, no. 2, pp. 203–214, 2001.
- [52] J. Nielsen, "The usability engineering life cycle," *IEEE Computer Society*, vol. 25, pp. 12–22, 2002.

- [53] T. O'Reilly, "O'Reilly network: What is Web 2.0," 2005. [Online]. Available: <http://www.oreillynet.com/lpt/a/6228>
- [54] C. Padden, "Interaction of morphology and syntax in American sign language," *Outstanding dissertations in linguistics*, 1988.
- [55] D. Rakowski. (2010) Stokoe TrueType font download page. [Online]. Available: <http://www.panix.com/~grvsmth/stokoe/>
- [56] T. Reenskaug, "The Model-View-Controller MVC," 2009. [Online]. Available: http://heim.ifi.uio.no/~trygver/2003/javazone-jaoo/MVC_pattern
- [57] K. Rohlfing, D. Loehr, S. Duncan, A. Brown, A. Franklin, I. Kimbara, J.-T. Milde, F. Parrill, T. Rose, T. Schmidt, H. Sloetjes, A. Thies, and S. Wellinghoff, "Comparison of multimodal annotation tools workshop report," *Gesprchsforschung*, vol. 7, pp. 99–123, 2006. [Online]. Available: <http://www.gespraechsforschung-ozs.de/heft2006/tb-rohlfing.pdf>
- [58] R. San-Segundo, J. M. Pardo, J. Ferreiros, V. Sama, R. Barra-Chicote, J. M. Lucas, D. Sánchez, and A. García, "Spoken Spanish generation from sign language," *Interactive Computing*, vol. 22, no. 2, pp. 123–139, 2010.
- [59] F. Saussure and R. Harris, "Course in general linguistics," *Open Court*, pp. 32–67, 1983, ISBN 0-8126-9023-0.
- [60] M. Schindler and D. Vrandeicc, "Introducing new features to wikipedia," *Wikimedia Deutschland e.V., Deutsche Nationalbibliothek, Germany*, pp. 1–4, 2009.
- [61] J. H. Schroeter R. and D. Kosovic, "Vannotea – a collaborative video indexing, annotation and discussion system for broadband networks," *In Knowledge Markup and Semantic Annotation Workshop, K-CAP*, 2003.
- [62] SignStream. (2009) SignStream official website. [Online]. Available: <http://web.bu.edu/asllrp/SignStream>
- [63] SignStream. (2010) SignStream download page. [Online]. Available: <http://www.bu.edu/asllrp/signstream/downloads.html>
- [64] W. C. Stokoe, "Sign language structure: an outline of the visual communication systems of the American deaf," *Department of Anthropology and Linguistics*, pp. 7–75, 1960.
- [65] R. Takkinen. (2009) Hamnosys official website. [Online]. Available: <http://www.sign-lang.uni.hamburg.de/projects/hamnosys>

- [66] D. E. van Wyk, "Virtual human modelling and animation for sign language visualisation," Master's thesis, Department of Computer Science, University of the Western Cape, Bellville, South Africa, 2008.
- [67] Vannotea. (2010) Vannotea official website. [Online]. Available: <http://www.itee.uq.edu.au/~eresearch/projects/vannotea/>
- [68] R. Vembu, "The 'Sign-in' (Cyn.in) system, developed by Cynapse," 2010. [Online]. Available: <http://www.expresscomputeronline.com/20090223/management04.shtml>
- [69] A. Voitkans, "Multimedia streaming in science education – new technologies and possibilities," *Journal of Young Scientists*, 2008. [Online]. Available: http://www.su.lt/bylos/moksls_leidiniai/jmd/08.03.19/voitkans.pdf
- [70] C. Wagner, "Wiki: a technology for conversational knowledge management and group collaboration," *Communications of the Association for Information Systems*, vol. 13, pp. 256–289, 2004. [Online]. Available: <http://cais.aisnet.org/articles/default.asp?vol=13&art=19>
- [71] J. Wijering. (2010) JW FLV player official website. [Online]. Available: <http://www.longtailvideo.com/players/jw-flv-player/>
- [72] J. Wijering. (2010) Modified JW FLV player official website. [Online]. Available: <http://wac.osu.edu/examples/jwplayercontrols/>
- [73] Wikipedia. (2010) Wikipedia official website. [Online]. Available: <http://en.wikipedia.org/>
- [74] Wikipedia. (2010) Wikipedia statistics page. [Online]. Available: <http://en.wikipedia.org/wiki/Wikipedia:Statistics>
- [75] I. H. Witten, D. Bainbridge, R. Tansley, C.-Y. Huang, and K. J. Don, "StoneD: a bridge between Greenstone and DSpace," *D-Lib Magazine*, no. 9, 2005. [Online]. Available: <http://www.dlib.org/dlib/september05/witten/09witten.html>
- [76] J. Zizka, M. C. Faps, and I. Szke, "Web-based lecture browser with speech search," *Proceedings of the Znalosti conference*, pp. 287–290, 2010. [Online]. Available: http://www.fit.vutbr.cz/research/view_pub.php?id=9229