

A DECISION SUPPORT SYSTEM
FOR THE READING OF ANCIENT DOCUMENTS

Henriette Roued-Cunliffe

Brasenose College
University of Oxford

Supervisor: **Professor Alan Bowman**

Centre for the Study of Ancient Documents
Faculty of Classics
University of Oxford

A thesis submitted for the degree
Doctor of Philosophy
Michaelmas Term 2011

ABSTRACT

The research presented in this thesis is based in the Humanities discipline of Ancient History and begins by attempting to understand the interpretation process involved in reading ancient documents and how this process can be aided by computer systems such as Decision Support Systems (DSS). The thesis balances between the use of IT tools to aid Humanities research and the understanding that Humanities research must involve human beings. It does not attempt to develop a system that can automate the reading of ancient documents. Instead it seeks to demonstrate and develop tools that can support this process in the five areas: remembering complex reasoning, searching huge datasets, international collaboration, publishing editions, and image enhancement.

This research contains a large practical element involving the development of a DSS prototype. The prototype is used to illustrate how a DSS, by remembering complex reasoning, can aid the process of interpretation that is reading ancient documents. It is based on the idea that the interpretation process goes through a network of interpretation. The network of interpretation illustrates a recursive process where scholars move between reading levels such as ‘these strokes look like the letter *c*’ or ‘these five letters must be the word *primo*’.

Furthermore, the thesis demonstrates how technology such as Web Services and XML can be used to make a DSS even more powerful through the development of the APPELLO word search Web Service.

Finally, the conclusion includes a suggestion for a future development of a working DSS that incorporates the idea of a layer-based system and focuses strongly on user interaction.

ACKNOWLEDGEMENTS

A sincere thank you for the time, advice and inspiration provided by my supervisor Professor Alan Bowman and the eSAD Project Investigators, Professor Melissa Terras and Professor Sir Michael Brady.

Thank you for invaluable support and motivating discussions to colleagues at the CSAD, Dr. Charles Crowther and Maggy Sasanow; member of the eSAD team, Dr. Ségolène Tarte; members of the BVREH team, John Pybus and Ruth Kirkham.

Thank you to Dr. Gabriel Bodard for much valued guidance with EpiDoc. Thank you to Dr. Mark Austin for insight into MDTSuite and ideas on how to develop a DSS.

Thank you to my family and friends for keeping my spirits up with large amounts of coffee and knitting, and a special thank you to those of you who have read through the thesis and given me your honest critique and helpful comments.

I wish to dedicate this thesis to my husband John. Without his emotional support and help with typing, reading through the thesis, feeding me and keeping me going it would never have happened.

TABLE OF CONTENTS

1. INTRODUCTION	1
1.1 THE RESEARCH PROJECT	3
1.2 VINDOLANDA TABLETS AS AN EXAMPLE	5
1.3 CONTEXT OF THE RESEARCH PROJECT	7
1.4 SUMMARY	8
 2. THE CHALLENGES OF READING ANCIENT DOCUMENTS	9
2.1 THE HISTORY OF READING ANCIENT DOCUMENTS	10
2.1.1 <i>The Private Sphere</i>	16
2.1.2 <i>The Public Sphere</i>	21
2.2 INTERPRETATION PROCESS	23
2.3 IT AND ANCIENT DOCUMENTS	25
2.3.1 <i>Image Annotation Tools</i>	30
2.3.2 <i>Textual Encoding Tools and Methods</i>	32
2.3.3 <i>Corpora and Online Resources</i>	36
2.3.4 <i>Image Manipulation and Analysis</i>	38
2.3.5 <i>Character Recognition</i>	41
2.4 COMPUTERS READING ANCIENT DOCUMENTS	46
2.5 COMPUTER BASED TASKS	49
2.5.1 <i>Remembering Complex Reasoning</i>	50
2.5.2 <i>Searching Huge Datasets</i>	52
2.5.3 <i>International Collaboration</i>	54
2.5.4 <i>Publishing Editions</i>	54
2.5.5 <i>Image Enhancement</i>	55
2.6 REASONING UNDER UNCERTAINTY	56
2.7 SUMMARY	59

3. DEVELOPING A DECISION SUPPORT SYSTEM PROTOTYPE	61
3.1 INTRODUCING DECISION SUPPORT SYSTEMS	64
3.1.1 <i>Evolution of Decision Support Systems</i>	67
3.1.2 <i>MDTSuite as a Decision Support System</i>	69
3.2 INPUT	71
3.3 STRUCTURE	73
3.4 MEANING	77
3.5 ARGUMENTS	82
3.6 SUMMARY	88
 4. THE DSS IN CONTEXT.....	90
4.1 LEXICAL RESOURCES	91
4.1.1 <i>Perseus Digital Library</i>	92
4.2 CONCORDANCES	93
4.2.1 <i>EpiDoc</i>	94
4.2.2 <i>Leiden Convention Encoding</i>	94
4.2.3 <i>Contextual Encoding</i>	98
4.2.4 <i>Transforming Encoding to XML</i>	100
4.3 APPELLO WEB SERVICE WORD SEARCH	102
4.3.1 <i>Methods</i>	104
4.3.2 <i>Patterns</i>	107
4.4 SUGGESTION ENGINE FOR DSS	107
4.5 OUTPUT FROM DSS	115
4.5.1 <i>Re-use Output with APPELLO</i>	117
4.5.2 <i>Re-use Output with Interactive Websites</i>	118
4.6 SUMMARY	127
 5. CONCLUSION	128
5.1 OUTCOME OF THE THESIS	131
5.2 RECOMMENDATIONS FOR A FUTURE DSS IMPLEMENTATION	138
5.3 UNDERSTANDING THE INTERPRETATION PROCESS WITH DSS	146

BIBLIOGRAPHY	147
---------------------------	------------

APPENDICES	155
-------------------------	------------

<i>Appendix A: Vindolanda Tablets 670 and 862</i>	<i>155</i>
---	------------

<i>Appendix B: Extract of Notes from Vindolanda Tablet 862 Interpretation</i>	<i>166</i>
--	------------

<i>Appendix C: Example videos on CD</i>	<i>168</i>
---	------------

<i>Appendix D: DSS Prototype Code</i>	<i>171</i>
---	------------

<i>Appendix E: Diagrams of the DSS Prototype Development</i>	<i>203</i>
--	------------

<i>Appendix F: APPELLO code</i>	<i>213</i>
---------------------------------------	------------

<i>Appendix G: List of DSS functions</i>	<i>244</i>
--	------------

<i>Appendix H: List of files on CD</i>	<i>247</i>
--	------------

LIST OF FIGURES

Fig. 2.1: Process of tablet reading plotted by reading level.....	24
Fig. 2.2: Network of interpretation	25
Fig. 2.3: Annotation of letter S	44
Fig. 2.4: XML from Inscriptions of Aphrodisias.....	53
Fig. 2.5: Model of interlinked arguments in the DSS	58
Fig. 3.1: Screenshot of DSS prototype main view.....	62
Fig. 3.2: Section A.i from Vindolanda tablet 670.....	63
Fig. 3.3: Vindolanda tablet 862	63
Fig. 3.4: Model of flow of information in the DSS	65
Fig. 3.5: Model of the DSS database	65
Fig. 3.6: Screenshot of letter editing pop-up box	72
Fig. 3.7: Model of DSS database with structure tables highlighted	73
Fig. 3.8: Screenshot of DSS structure.....	74
Fig. 3.9: XML from DBtoTEI Web Service	77
Fig. 3.10: Model of DSS database with meaning table highlighted	78
Fig. 3.11: Screenshot of line 7 from Vindolanda tablet 670 in the DSS	79
Fig. 3.12: Screenshot of the editing of the fifth character in cogn.tis.....	80
Fig. 3.13: DBtoTEI Web Service result for character 115	80
Fig. 3.14: Legend showing the resolution of letters	81
Fig. 3.15: Model of DSS database with arguments and links tables	83
Fig. 3.16: Screenshot of the editing of the letter ‘p’	84
Fig. 3.17: Screenshot of letter and arguments editing pop-up boxes.....	85
Fig. 4.1: URL sent to Perseus to retrieve lexicon results for the word <i>frater</i>	92
Fig. 4.2: XML from Perseus for lexicon results of the word <i>frater</i>	93
Fig. 4.3: Screenshot of results from IAph2007 for words beginning with <i>M</i>	98
Fig. 4.4: Screenshot of text 11.415.i.1 from IAph2007	99
Fig. 4.5: Model of the interconnectivity between the methods	105
Fig. 4.6: XML for APPELLO method get_tablets.....	105
Fig. 4.7: XML for APPELLO method get_tablet and tabletID = 581	106
Fig. 4.8: XML for APPELLO method get_words and pattern = pullus	106
Fig. 4.9: XML for APPELLO method get_words with pattern = *[bd]em	107
Fig. 4.10: Screenshot of the search engine selector	108

Fig. 4.11: Screenshot of line 5 from Vindolanda tablet 670 in the DSS	110
Fig. 4.12: XML from the Vindolanda Tablets APPELLO	110
Fig. 4.13: Screenshot of result from the Vindolanda Tablets APPELLO.....	111
Fig. 4.14: Screenshot of Vindolanda tablet 580 on the VTOII website.....	112
Fig. 4.15: Screenshot of the word editing pop-up box.....	113
Fig. 4.16: Screenshot from Perseus Latin Dictionaries for word <i>frater</i>	113
Fig. 4.17: Screenshot from Perseus Latin Dictionaries for word <i>productilis</i>	114
Fig. 4.18: XML from DBtoTEI for line 1 in Vindolanda tablet 670	116
Fig. 4.19: XML for line 1 from original encoding of Vindolanda tablet 670.....	116
Fig. 4.20: Screenshot of tablet 291 on the VTO website.....	119
Fig. 4.21: Screenshot of a page in the online exhibition on the VTO website	119
Fig. 4.22: XML for Vindolanda tablet 291 from original VTO website	119
Fig. 4.23: Screenshot of the search facilities of the VTO website.....	120
Fig. 4.24: Screenshot of the browsing facility of the VTO website	121
Fig. 4.25: Screenshot of the basic tablet search on the VTO2 website	121
Fig. 4.26: Screenshot of the browsing facility on the VTO2 website.....	122
Fig. 4.27: Screenshot of filtering for archaeological period 3 on VTO2 website	123
Fig. 4.28: Screenshot of index searching facility on the VTO2 website	123
Fig. 4.29: Screenshot of the result from a search on the VTO2 website	124
Fig. 4.30: Close-up screenshot of tablet 291 on the VTO2 website	125
Fig. 4.31: Screenshot of tablet 291 on the VTO2 website.....	125
Fig. 4.32: Screenshot of the pop-up for name Aelius on the VTO2 website.....	126
Fig. 4.33: Screenshot of the pop-up for note 1 on the VTO2 website	126
Fig. 5.1: Drawing of the layer-based DSS concept.....	139
Fig. 5.2: Drawing of ‘button clicking’ input method.....	142
Fig. 5.3: Drawing of ‘drag and drop’ input method.....	143
Fig. 5.4: Drawing of ‘image annotation’ input method	144

LIST OF TABLES

Table 2.1: Conventions from Oxyrhynchus and Leiden Conventions	14
Table 2.2: Inscription and papyrus symbols, van Groningen's compromise	15
Table 2.3: Terras' reading levels	18
Table 2.4: Glossary of terms for encoding texts.....	21
Table 3.1: DSS_sections table from DSS database	71
Table 3.2: DSS_lines table from DSS database	75
Table 3.3: Part of DSS_words table from DSS database	76
Table 3.4: Part of DSS_characters table from DSS database.....	76
Table 3.5: Part of DSS_letters table from DSS database	79
Table 3.6: Part of DSS_arguments table from DSS database	83
Table 3.7: Part of DSS_linkToDoc table from DSS database.....	84
Table 3.8: Part of DSS_linkToBibl table from DSS database.....	86
Table 4.1: The editing conventions used in Tab.Vindol.III	95
Table 4.2: The EpiDoc tags used in the Vindolanda tablets.....	97
Table 4.3: DSS_engines table from DSS database.....	109

LIST OF ABBREVIATIONS

ala2004 – (Roueché 2004)

AJAX – Asynchronous Javascript and XML

BVREH - Building a Virtual Research Environment for the Humanities

CIL - Corpus Inscriptionum Latinarum

CMS – Content Management System

CSAD – Centre for the Study of Ancient Documents

ES – Expert Systems

eSAD - E-Science and Ancient Documents

GRAVA - Ground Reflective Adaptive Vision Architecture

InsAph - Inscriptions of Aphrodisias

Iaph2007 – (Reynolds *et al.* 2007)

IRT – Inscriptions of Roman Tripolitania

MAMA XI - Monumenta Asiae Minoris Antiqua XI

MIS - Manage Information Systems

OeRC - Oxford e-Research Centre

PHP - Hypertext Preprocessor

RDF - Resource Description Framework

REST - Representational State Transfer

SQL – Structured Query Language

Tab.Vindol.I – (Bowman and Thomas 1983)

Tab.Vindol.II – (Bowman and Thomas 1994)

Tab.Vindol.III (Bowman and Thomas 2003)

Tab.Vindol.IV (Bowman *et al.* 2010a)

TILE - Text-Image Linking Environment

TEI - Text Encoding Initiative

TPS - Transaction Processing Systems

URI/URL – Uniform Resource Identifier / Uniform Resource Locator

VRE-SDM - Virtual Research Environment for the Study of Documents and Manuscripts

VTO – Vindolanda Tablets Online

VTO2 – Vindolanda Tablets Online II

XML - Extensible Markup Language

XSLT – Extensible Stylesheet Language Transformations

1. INTRODUCTION

The field of reading ancient documents has a long history, during which the methods and resources used have continuously evolved. Throughout this history, however, the cognitive process used by specialist documentary scholars¹ in their attempts to interpret these documents has remained the same. It is this cognitive process that forms the main focus of this research, particularly how interpretation can be remembered and stored and how IT can be used as a tool to aid in this process.

The research question presented in this thesis is whether the IT technology known as Decision Support Systems (DSS) can be used as a tool to aid documentary scholars with the interpretation of ancient documents. The overall hypothesis is that it can, and this is tested through a sequence of separate aims.

The first aim is to gain an understanding of the cognitive process of reading ancient documents and to examine how an interpretation is formed. This is accomplished in Chapter 2 through a review of literature from previous research into the challenges of reading of ancient documents.

The second aim is to establish what specific requirements are needed of IT tools for the task of aiding interpretation in Humanities disciplines. This will be achieved, in Chapter 2, by a discussion of human and computer interaction and the identification of tasks for which computers can provide beneficial tools versus tasks that require human involvement.

¹ Specialist documentary scholars included specialists in the reading of all types of texts that require deciphering. These texts can be found in many ancient cultures (e.g. Nordic runes, Chinese scripts or Cuneiform) and more modern settings (e.g. Medieval treaties or 18th Century love letters). This thesis focuses especially on Greek and Latin texts and will therefore use the term scholar as shorthand for papyrologists, epigraphers and palaeographer. The fields of Papyrology, Epigraphy and Palaeography are introduced in 2.1.

The third aim is to test the concept of using a DSS in an interpretation scenario. For this purpose I have developed a DSS prototype based on an example ancient document. This enables me to examine different concepts such as decision input and storage. Chapter 3 documents the development of the DSS prototype.

The final aim is to understand how a DSS fits into the context of other services and systems present in Digital Humanities. As an example of this, Chapter 4 presents parallel research into Web Services and discusses how this can be used within a DSS.

The conclusion of this thesis will attempt to illustrate how this research can be relevant to the reading of documents, whether ancient or relatively modern. The main body of research, however, will focus on ancient documents from the Greek and Roman periods. The history of reading ancient documents plays a large part in the understanding of the task involved, and goes back to the very first writing systems (see 2.1).

When writing is read many years after it was written there is often a degree of interpretation involved, together with a variety of other issues that prevent the full understanding of an ancient document. These issues can be anything from simple poor handwriting, the fragmentary state of a text, text that may be partially destroyed, or a lack of knowledge of the writing system itself.

Because the reading of ancient documents is so often an interpretative process it is my firm opinion that it is also a human process. Therefore it would be very unrealistic to believe that an IT system could be created to automatically read ancient documents. Any such system would need to cover all aspects of the process and be able to take into consideration the thought processes of every scholar when determining the text outcome. This opinion plays a vital role in the decision to focus this research on a Decision Support System rather than on an automated reading system, and this will be outlined in detail in Chapter 2.

1.1 THE RESEARCH PROJECT

Research for this thesis proposes to cover the subject primarily through the development of a DSS prototype (Chapter 3), and through demonstration of how this prototype can be integrated with Web Services and XML technology in order to become more powerful and to provide a fuller support tool for scholars reading ancient documents (Chapter 4). The development of a prototype is in itself a very important tool for this purpose (Galey and Ruecker 2010). The prototype DSS will explain through demonstration how the inclusion of DSS in the reading of ancient documents can facilitate a more rounded interpretation.

The DSS prototype will follow scholars through their interpretation process, enabling them to access external resources to support their reading. It will record each decision made, using previously recorded decisions as a guide in a continual decision-making process. Throughout the development I will examine which tasks in particular an application such as the DSS can potentially aid the work of scholars. Finally, I will discuss the general value of this type of software development in Humanities.

Using a DSS will make scholars more aware of the mental and physical processes they go through while reading ancient documents. The application will highlight inconsistencies in the interpretation whilst still allowing these inconsistencies to remain, since any other course of action would be counterproductive. The idea of the DSS is not to build a new framework within which readers must now operate, but to develop a system that will behave like a good assistant, who will continually remind scholars of inconsistencies, whilst still giving them the freedom to make their own interpretative decisions.

Additional technology that I think will advance the DSS, and give scholars fuller support, is centred on Web Service and XML technology. In this context, a Web Service is

defined as a service that enables the user to send a constructed URL through a browser to a server and receive a reply in a machine-readable format such as XML (Abeysinghe 2008: 13).

Although the idea of external access to data through a URL in a Web browser has been in existence for a long time, it is only recently, with RESTful Web Services², that this has become a feasible and lightweight solution enabling with the use of XML, the dissemination of knowledge in a simpler and smoother way.

The word search Web Service built for EpiDoc encoded ancient document resources is called APPELLO (Latin for ‘I call’). The APPELLO Web Service has been developed using the Vindolanda writing tablets as its primary case study. APPELLO has been tested with other EpiDoc/TEI encoded ancient document resources such as the Inscriptions of Aphrodisias (InsAph), the Inscriptions of the Roman Tripolitania³ (IRT) and the Monumenta Asiae Minoris Antiqua XI⁴ (MAMA XI) documents. Although APPELLO has been developed for EpiDoc/TEI encoded resources, there is no reason why, with some minimal adjustments, it should not work for other XML encoded or database resources. Chapter 4 contains a fuller explanation of APPELLO and Web Services and how they have been used in this research.

One of the major decisions when choosing which technologies to use in the development of both the DSS prototype and APPELLO, was the storage system for the DSS prototype. To my knowledge the most advanced Open Source database systems at the moment are Triplestore and Postgres⁵. However, as I am developing a smaller prototype, I decided to use the database systems I felt most comfortable with, MySQL⁶ (a relational

² See 4.3 for an description of RESTful Web Services.

³ <http://irt.kcl.ac.uk>

⁴ <http://mama.csad.ox.ac.uk/>

⁵ <http://www.postgresql.org/>

⁶ MySQL is an open source relational database management system which runs as a server and is a popular choice for Web applications - <http://www.mysql.com/>

database that stores data in tables and fields) and to use the program PHPMyAdmin⁷ as an interface to access the database.

Similarly, I chose the scripting language PHP⁸, instead of the more popular choice for larger applications, Java because it is generally more lightweight for browser-based applications, making it a good choice for this prototype. Where the need arose to script parts of the prototype that exceeded the capabilities of PHP, for example the LiveSearch mechanism and the pop-up windows, I used JavaScript⁹ and AJAX¹⁰ scripts, which work well in combination with PHP.

1.2 VINDOLANDA TABLETS AS AN EXAMPLE

The DSS prototype has been demonstrated through use of the Vindolanda Tablets. However, I see no reasons why it should not also work on other types of document. Throughout the thesis I have used examples from the Vindolanda Tablets to illustrate various points, and the following is a brief introduction to this dataset.

The writing tablets found at the Fort of Vindolanda offer a unique opportunity to learn about the Roman occupation in Britain, as well as the lives of the people who occupied this fort and other military settlements along Hadrian's Wall. Furthermore, they offer an opportunity to examine the standards of literacy and the training of scribes, which in turn can be compared to other military environments in the Roman Empire. The Vindolanda Fort was occupied from the late first century AD until the Roman army abandoned its northern British frontier three centuries later (Adams 1995, Bidwell 1997, Birley 1997; Bowman 1994).

⁷ phpMyAdmin is a browser-based application from which it is possible to manage MySQL databases. <http://www.phpmyadmin.net>.

⁸ PHP: Hypertext Preprocessor is a server-side scripting language often used for open source programming - <http://www.php.net/>

⁹ Javascript is a scripting language for Internet programming.

¹⁰ AJAX (Asynchronous Javascript and XML) is a method of using exciting standards to develop interactive Web applications. See Gareth 2005 for an introduction to AJAX.

This insight into the lives of the Roman military at the frontier was the motivation for the websites Vindolanda Tablets Online (VTO) and Vindolanda Tablets Online II (VTO2).

There is a real fascination with this written record from nearly two thousand years ago, and interest in the Vindolanda Tablets is widespread: The tablets are used in primary schools, as Latin teaching aids along with the help of Minimus the mouse (Bell 2007); the BBC has a section of their ancient history site dedicated specifically to Vindolanda and the tablets (Ibeji 2001); the tablets were selected as one of the Top Ten British Treasures by the BBC2 program of the same name (British Museum Website 2011). Since going live in 2004 the Vindolanda Tablets Online website has proved to be very popular, and for this reason the Centre for the Study of Ancient Documents was keen to add Volume III of the Vindolanda Tablets (Bowman and Thomas 2003) as soon as possible to the material already online. In light of this, the work described in Chapter 4 is as important to the CSAD as it is to this thesis. The published editions of tablets 670 and 862, used as the main examples in the development of the DSS prototype, can be found in appendix A.

1.3 CONTEXT OF THE RESEARCH PROJECT

This thesis research is funded by an Arts & Humanities Research Council (AHRC) Doctoral Studentship attached to the Image, Text, Interpretation: e-Science, Technology and Documents project¹¹ (also known as eSAD: e-Science and Ancient Documents). The eSAD project aims to explore the use of computer technology as an aid in the reading of damaged documents. The four-year project, a collaboration between the Centre for the Study of Ancient Documents (CSAD), the Department of Engineering Science at the University of Oxford and the Department of Information Studies, University College London, was funded under the AHRC-EPSRC-JISC Arts and Humanities e-Science Initiative¹², and ran from 2008-2011.

The eSAD project focused on different IT tools, such as character recognition and Decision Support Systems and worked closely with the Building a Virtual Research Environment for the Humanities¹³ (BVREH) project, towards a better understanding of the cognitive process of reading ancient documents, in order to provide better tools for this group of scholars.

In this thesis, many of the conclusions concerning the nature of interpreting ancient documents were formed through discussions at group meetings held by the two projects.

¹¹ The E-Science and ancient documents project (eSAD) brings together project investigators: Prof. Alan Bowman (Centre for the Study of ancient documents, University of Oxford), Prof. Sir Michael Brady (Department of Engineering Science, University of Oxford) and Prof. Melissa Terras (Department of Information Studies, University College London).

¹² http://www.ahrcict.rdg.ac.uk/activities/e-science/awards_2007.htm

¹³ The collaboration between the eSAD project and the BVREH project was particularly focused around the BVREH sub-project A Virtual Research Environment for the Study of Documents and Manuscripts (VRE-SDM), which ran from 2007-2009 with Prof. Alan Bowman as the Project Investigator. <http://bvreh.humanities.ox.ac.uk/VRE-SDM.html>

1.4 SUMMARY

This chapter has introduced the research question and hypothesis and outlined the different aims that were achieved in order to answer this question.

The chapter then presented the research project and the tools used as a part of the DSS prototype (Chapter 3) and the APPELLO word search Web Service (Chapter 4) developments. I have used mainly Open Source tools, and the intention is that products of this thesis (i.e. the DSS prototype and APPELLO) will be Open Source developments that can be of use in future projects working with similar tasks.

2. THE CHALLENGE OF READING ANCIENT DOCUMENTS

The first aim of this research is to understand the cognitive process of reading ancient documents and to examine how an interpretation is formed. Therefore, this chapter will begin with a literature review of previous research into the challenges of reading ancient documents.

There then follows a detailed discussion of the interpretation process integral to the understanding of an ancient document.

Furthermore, this chapter will address the second aim, which is to understand the requirements that are needed of IT tools for the task of aiding interpretation in Humanities disciplines. This will be achieved through a discussion of the concept of computers "reading" ancient documents. A common misconception, when the idea of building a DSS to aid the reading of ancient documents is mentioned, is that the aim is to develop an automated system that can output a final transcription of a text. In other words, it is wrongly assumed that such a system would automatically do a significant and fundamental part of the scholar's job - without involving the scholar.

Today's technology is capable of training a system to apply a set of rules to an image of a document, and to output what bears a resemblance to an interpretation. This is still an enormous task, and in this chapter I maintain that in this context it is an irrelevant task since the reading and understanding of ancient documents is essentially an entirely interpretative function that I consider to be uniquely human. However, this is part of a larger discussion on computer intelligence and whether machines can think or not, which will be covered below.

This thesis suggests instead that there are other tasks involved in the process of working with ancient documents that scholars find difficult and time-consuming, and it is these tasks that can be more easily carried out by a computer system. These tasks are introduced in this chapter along with proposed methods for addressing them.

This is followed by a discussion of the concept of *reasoning under uncertainty*. Uncertainty plays a major role in any interpretation process, as without it, it would not be a question of interpretation, but of plain transcription. The question is: "How is uncertainty incorporated into the DSS?" This research uses the method of argumentation, inspired by the MDTsuite development, rather than a more quantitative approach. This section will explain what argumentation is and how this has been incorporated into the DSS.

2.1 THE HISTORY OF READING ANCIENT DOCUMENTS

The skills of reading and writing have, by their nature, been closely interlinked since the very earliest proto-writing in the form of symbols from the Danubian Culture (5000-3800 BCE) (Guichard 2002: 17), through Cuneiform, Hieroglyphics, Greek scripts, Latin scripts and finally on to the written word of today. There are two main types of writing systems; pictographic and alphabetic¹. While alphabetic writing has a limited amount of letters which when combined create an unlimited amount of words, pictographic writing usually has a larger amount of symbols which need deciphering. These two writing systems thus provide very different practical challenges for the development of digital aiding tools.

Writing and reading must have evolved simultaneously as without one the other hardly makes sense. For this reason an enormous amount of time and interest has been invested in efforts to understand texts written several thousand years ago. These texts were written with the aim that they would be understood in the present and in some cases the future as well. Therefore, in spite of their value as a museum object the sense that they make as archaeological artefacts is seriously limited if they cannot be read and understood.

Ancient texts can in one way be compared to the hoard of gold found in Staffordshire in late 2009 (Pitts 2009a). The value of the gold hoard as a treasure and as a means of attracting visitors to museums is immense. However, without proper excavation

¹ See Brown 1998 for an overview over different types of writing and their history.

and recording in order to reveal the context of a hoard such as this, it is of little value as an archaeological artefact. It can not tell us much, other than in a certain period in history this gold was made into jewellery and other artefacts and then at some point after this time it was all buried for some reason or another. It will provide jewellery and artefacts to study and they themselves can be dated but without context it is near impossible to learn when the hoard was buried. Luckily in this case the finder of the hoard has recorded everything about the context that there was to record (Pitts 2009b).

However, there are a variety of different issues involved with the understanding of ancient documents; some can be overcome fairly easily while others present more of a challenge. These issues can range from simple bad handwriting to a fragmentary state, or the partial destruction of the document. Difficulties can also arise through a lack of knowledge of the writing system being studied or lack of comparable material, as a large part of what was written in the past has not been preserved (Turner 1973: 2). Furthermore, there are writing systems, such as Linear A, that have still not yet been deciphered at all².

The Greek and Roman historical periods have produced a massive amount of textual material and thus the community of scholars with an interest in this material is relatively large.

The following part of this thesis will cover the existing knowledge of these scholars of Greek and Roman texts, and how they read these ancient documents. The section is divided into what Herbert Youtie (1963) calls the private and the public sphere of the field. The private sphere can be described as the actual reading process of the scholars either on their own or in a team, a process that ultimately provides these individuals with an understanding of the text they are reading. The public sphere is reached by the next step in the process, when an edition of the text is published. Once in the public sphere the published edition engages with other texts and scholars who may or may not themselves

² For an overview of Linear A and the issues faced when attempting to decipher it see Gordon 1958, Nagy 1965 and Rendsburg 1996.

read ancient documents, but who will use these editions as primary sources in their own research.

Scholars who read and understand Greek and Latin documents can be divided roughly into three groups; Palaeographers who read ancient documents in order to study the physical features of the writing and to date the text itself (Bischoff 1990: 2; Ciula 2005: §1); Epigraphers who study inscriptions or epigraphs mainly in stone or metal, and which are often of a formal or legal nature (Mahoney 2006); Papyrologists who study writing on papyri or similar materials and who focus mainly on everyday texts (Pestman 1990). Inevitably, however, there is often an overlap between all these fields of study (Bagnall 2009: xvii; Ciula 2009: 220). All three groups have in common that they use interpretive skills to read and understand handwriting. Despite their different goals and the different material they study each group faces the same complex reasoning processes and the same issues of remembering this complex reasoning.

The work of Chaeremon, an Egyptian priest and Stoic philosopher is an example of a very early attempt to decode and translate a pictographic text into an alphabetic text. In the first century CE he attempted to translate Egyptian ritual texts into Greek (Bowman 2002: 204-205). Any translation, especially one from a pictographic writing system, is also an interpretation that requires knowledge of the meaning of the text. In this particular case Chaeremon's background gave him a good understanding of the meaning of the text. In most modern cases scholars have to rely on their knowledge of the context of the text to understand the meaning of the text.

Palaeography as a subject originated in the work of monks in Benedictine monasteries. Maurist Bernard de Montfaucon first coined the term itself in 1708 with the publication of '*Palaeographia Graeca*'. The field of study is thought to have been initiated in 1681 by the monk Jean Mabillon, who began the systematic study of the handwriting in Latin manuscripts (Bischoff 1990: 1; Boyle 2001; David and van Groningen 1952).

Modern Epigraphy originated during the Renaissance when there was an increasing fascination with all things Roman or Greek. Later during the 18th and 19th centuries many young men from northern Europe were sent to the Mediterranean as an integral part of their education. They often had schooling in Greek and Latin and spent time using these skills to transcribe the inscriptions which were readily available in the landscape and towns around the Mediterranean (Cayless *et al.* 2009). The field of Epigraphy became more organised with the publication of volumes of editions such as the *Corpus Inscriptionum Latinarum*³ (CIL) published first in 1862 in Berlin (Gordon 1983: 8-9, Mahoney 2006).

Papyrology, on the other hand, did not have material available as early as Palaeography or Epigraphy. In fact some would even date the beginnings of Papyrology to the discovery in the mid 18th Century at Herculaneum⁴ of carbonised papyri in the form of cigar-shaped rolls (Keenan 2009: 59). Another line of thought however, attributes the beginnings of this field of study to the first publication of papyrological texts, namely an edition of a Greek papyrus recording a series of receipts for work in 193 CE on irrigation dikes in Fayyum, Egypt by Danish classicist Niels Iversen Schow (1754-1830) (Keenan 2009: 59). The first use of the term papyrologists to describe scholars working with papyri was in relation to the publication of the first edition of BGU⁵ (van Minnen 1993).

Whenever the beginnings of each of these fields precisely occurred, there is no doubt that by the late 19th century, editions were being published quite regularly using a variety of different editorial conventions. So much so that, by the 1930s, the scholarly community called for more stringent guidelines on the format of publishing.

³ http://cil.bbaw.de/cil_en/index_en.html.

⁴ Herculaneum is a South Italian town buried by the eruption of the volcano Vesuvius in 79 CE.
<http://www.herculaneum.ox.ac.uk/>

⁵ *Griechische Urkunden* (Berlin) (Koeniglichen Museen zu Berlin 1895-1912).

Name	Oxyrhynchus 1898	Leiden Conventions
Uncertainty	abc	abc
Illegible		 , -10- , $\mp$ 10-
Gaps	[], [...]	[], [...], [-10-], [$\mp$ 10-]
Supplements		[abc]
Omissions	<abc>	<abc> , < >
Solutions	(abc)	(abc)
Interpolations		{abc} , {...}
Erasures	[[abc]] , [[...]]	[[abc]] , [[...]]
Interlinear additions		'abc'

Table 2.1: The conventions used in the first Oxyrhynchus volume of papyri found during Grenfell and Hunt's 19th Century excavations at Oxyrhynchus in Egypt (Grenfell and Hunt 1898) and the Leiden Conventions (van Groningen 1932b).

When publishing transcriptions of ancient documents it is important to be able to give the reader of the transcription an idea of the state of the text. This is done through the use of a system of symbols and brackets. For example the fact that a line of text ends abruptly leaving a gap in the text was sometimes illustrated in the published transcription with a square bracket at the end of the line.

When papyrologists Bernard Pyne Grenfell (1869-1926) and Arthur Surridge Hunt (1871-1934) in 1898 first published editions from their excavations at Oxyrhynchus, Egypt, they used their own system of dots and brackets to indicate the state of the text (table 2.1) (Grenfell and Hunt 1898).

Name	Inscriptions	Papyrus	van Groningen's Compromise	Leiden Conventions
Uncertainty			abc	abc
Illegible		[....], []	, ..10.., ..⊥10..	, -10-, ⊥10-
Gaps		[....], []	[], [..], [..10..], [..⊥10..]	[], [..], [-10-], [⊥10-]
Supplements	[abc]	[abc]	[abc]	[abc]
Omissions	(abc)	<abc>	<abc>, < >	<abc>, < >
Solutions	(abc)	(abc)	(abc)	(abc)
Interpolations	<abc>, [[abc]]	{abc}, <<abc>>	[[abc]]	{abc}, {....}
Erasures		[[abc]]	{abc}, {..}	[[abc]], [[....]]
Interlinear additions			'abc'	'abc'

Table 2.2: *Inscriptions and papyrus (Bidez and Drachmann 1932). Van Groningen's compromise (van Groningen 1932a). Leiden Conventions (van Groningen 1932b).*

In 1932 J. Bidez and A. B. Drachman published a survey of the systems used by Papyrologists and Epigraphers (Bidez and Drachman 1932). They found some similarities but also numerous dissimilarities (table 2.2). A short survey of some of the larger corpora of the time confirms their findings. For example, there were up to five different ways of indicating an interpolation (i.e. letters wrongly added by the scribe and now deleted by the editor) - {abc}, <abc>, <<abc>>, [abc] and [[abc]]. Examples of this unregulated system can be found in the early publications of *Supplementum Epigraphicum Graecum*⁶ (SEG) where in SEG 1 (Hondius 1923), interpolation is written <abc>, in SEG 4 (Hondius 1929) it is written [[abc]], and in the later SEG 7 (Hondius 1934), it is written {abc} as suggested by the Leiden Convention.

It was in response to this lack of a universally understood system that in 1931 B. A. van Groningen presented a paper (van Groningen 1932a) at the second International

⁶ <http://www.brill.nl/publications/supplementum-epigraphicum-graecum>.

Congress of Papyrologists⁷ in Leiden, in which he outlined what he felt to be a suitable compromise for an editorial signage system based on Bidez and Drachman's survey (Bidez and Drachman 1932) (table 2.2). After some amicably agreed amendments by Hunt and Bell a system was agreed upon which in effect discarded van Groningen's compromises in favour of the system used in 1898 by Grenfell and Hunt (table 2.1) (Gallo 1986: 91). This system or set of editorial signs is what is known today as the Leiden Conventions (van Groningen 1932b) and it is 'widely adopted for the editing of epigraphical and papyrological texts' (Dow 1969: 1).

2.1.1 THE PRIVATE SPHERE

In Papyrology it is possible to identify two different schools of thought. Youtie was an advocate for "pure" Papyrology in which the main function of the papyrologist is to interpret papyri and publish editions. Jean Bingen, on the other hand, is of the opinion that papyrological activities are diverse and can include literary studies, Ptolemaic chronology, economic history and New Testament textual criticism, among other topics, along with the core focus on the decipherment of texts. Bingen thus envisages Papyrology as a kind of historical sociology, which is also known as "applied" Papyrology. Amongst the supporters of Bingen's approach is C. Bradford Welles, who writes: 'I am less interested myself in making papyrologists than showing classical students what there is in the papyri, and how it relates to antiquity in general' (Keenan 2009: 72-74).

"Pure" papyrology as proposed by Youtie is the central focus of this thesis as it looks at the papyrologist as a reader of ancient documents and this led Youtie to study the psychological aspects of reading and deciphering (Youtie 1963). This study has in turn greatly helped in the understanding of the scholar and the process of reading ancient documents. The goal for reading ancient documents is to achieve the true meaning of the text. This is often hindered by damage to the text but can also be hindered by a lack of

⁷ A special session at the 18th International Congress of Orientalists.

understanding of the writing system or the context of the document. Therefore, the reading of ancient documents becomes a process of interpretation in order to reconstruct the true meaning of the text.

In order to successfully develop an aid for the reading of ancient documents it is important to first understand this process of interpretation. In human psychology, reading as a separate subject has received its fair share of interest. In the 13th century Saint Thomas Aquinas (1225-1274) realised that sight is 'the greatest of the senses through which we acquire knowledge' (Manguel 1997: 28). Earlier in the 11th century the Egyptian scientist al-Haytham (965-1039) distinguished between the unconscious 'pure sensation' and 'perception', which requires a voluntary act of recognition (Manguel 1997: 34). Nevertheless, despite centuries of focus on the psychology of seeing and reading⁸, in his book *A History of Reading*, Alberto Manguel admits that 'we are still far from the answer, mysteriously we continue to read without a satisfactory definition of what it is we are doing' (Manguel 1997: 39). In spite of all this research into the psychology of reading there is still very little research into the reading and interpretation of damaged documents. As in many other fields, scholars of this subject are not good at conveying the mental and physical processes involved in their study. Roger Bagnall suggests that Papyrology has not been a very reflective discipline. He believes that the reason for this lies in the enormous technical demands such as the mastering of palaeography, languages and editorial technique (Bagnall 1995: 1-2).

Paul Johnson calls this behaviour the 'paradox of expertise' (Johnson 1983: 79). As scholars become more and more expert in their field they conduct their tasks more automatically.

Behavior that is automatic required less cognitive awareness. Behavioral awareness hinders an expert's ability to verbally describe the knowledge or essential steps an action or solution required. (McGraw and Harbison-Briggs 1989: 17)

In other words, the process of reading ancient documents has often become such

⁸ For an overview of the psychology of seeing and reading see Gregory 1998 and Smith 2004.

an automated task for experienced scholars that most do not have the ability or see the need to verbally explain how they do it. Furthermore scholars who are experts in an area will often describe their work process in terms of idealised theories (Boose 1986).

However, a few scholars are very interested in the actual process of reading ancient documents. Youtie is one of these scholars, who, holds the view that a text is not read letter-by-letter in a linear fashion. Another of these scholars is Giorgio Pasquali, who says:

one is not sure of having read a letter correctly unless he has first guessed the word, of having read the word correctly unless he has a general idea of the meaning of the sentence. It is this truth that is ignored by the reader who is partial to mechanical transcription (Pasquali 1936: 44-45 quote from Youtie 1963: 24).

This would indicate a more recursive style of interpretation where the scholars move back and forth between a letter, a word and a sentence, with one hypothesis confirming the next.

As a part of her D.Phil research Melissa Terras (2002) gained an understanding of the reading process undertaken by three papyrologists by analysing this same process with the help of knowledge elicitation techniques such as structured interviews, Think Aloud Protocols (TAP), sorting and laddering. In order to facilitate the analysis and organisation of the scholars' discussions as they decipher a text she defined ten reading levels (table 2.3).

Reading level	Thematic subject
8	Meaning or sense of document as a whole
7	Meaning or sense of a group or phrase or words
6	Meaning or sense of a word
5	Discussion of grammar
4	Identification of possible word or morphemic unit
3	Identification of sequence of characters
2	Identification of possible character
1	Discussion of features of character
0	Discussion of physical attributes of the document
-1	Archaeological or historical context

Table 2.3 Terras' reading levels (Terras 2005: 48). Used with permission of Melissa Terras.

Her examination of the three papyrologists at work led Terras to reassert the idea of a cyclical reading process where scholars move between the different reading levels:

An expert reads an ancient document by identifying visual features, and then incrementally building up knowledge about the document's characters, words, grammar, phrases and meaning, continually proposing hypotheses, and checking those against other information, until s/he finds that this process is exhausted. At this point representation of the text is prepared in the standard publication format. At each level, external resources may be consulted, or be unconsciously compared to the characteristics of the document (Terras 2006: 82).

From her observations Terras concludes that all three scholars had different preferences in how they approached the text even though they all used the same recursive process. One scholar preferred to use digital images of the text, another preferred to draw his transcription and the third preferred to view printed photos of the text (Terras 2006).

Sékolène Tarte (2011a) has continued this line of investigation and, by examining video recordings⁹ of the same three scholars analysis of the re-interpretation of the 'Frisian Ox Sale' tablet¹⁰, has outlined two different approaches. By examining the strategies of the three scholars she has found that (P1) 'relies upon the characters that he reads as almost-certain letters and tries out different hypotheses. In contrast with (P2) who draws the letters as he sees them, i.e. with the shapes of the original script' (Tarte 2011a: 12). In this case (P1) is using what Tarte calls the palaeographical/kinaesthetic¹¹ approach. This makes use of tracing the characters and applying artistic skill to a reconstruction of the movements of the scribe, thus gaining understanding of the text. In the meantime (P2) is using what Tarte calls the philological/cruciverbalistic¹² approach, which likens the decipherment process to solving a cryptic cross-word puzzle¹³; a scholar using this approach will often begin by identifying the more legible letters of the text and then use these as a foundation for a

⁹ This digital video record of the interpretation sessions of the 'Frisian Ox Sale' tablet was produced as a part of the BVREH project (Bowman *et al.* 2010b).

¹⁰ The Frisian Ox Sale was first published in 1917 (Volgraff, 1917). It has since been re-interpreted in 2009 (Bowman *et al.* 2009).

¹¹ The kinaesthetic approach can be loosely explained as 'learning by doing' in the sense that scholars are understanding the text by drawing or otherwise reproducing what they see.

¹² Cruciverbalism is a term borrowed from French to denote the process of solving a cryptic crossword puzzle.

¹³ Cryptic crosswords are crossword puzzles where each clue is a word puzzle in and of itself.

subsequent hypothesis (Tarte 2011a; Tarte 2011b).

Of course those two palaeographical/kinaesthetic and philological/cruciverbalistic approaches are not mutually exclusive. (P1) also resorts to tracing letters (e.g. line 68) and (P2) also uses the philological register (e.g. line 34) to build a case for the 'A'. (Tarte 2011a:12)

The choice of approach by the scholar is thought to be highly interconnected with the scholar's personal skill-set (de La Flor *et al.* 2009).

Whatever the approach of the scholar, it is important to remember that the product is highly individual and the quality of the interpretation is directly linked with the scholars experience and knowledge. Most scholars will admit that even though they have years of experience and have had many successful editions published, there will still be texts that are simply illegible to them. This does not mean, however, that these same texts will be illegible to everyone (Youtie 1963).

A recent example of this is the so-called 'Frisian Ox Sale' tablet. When first published in 1917, it was interpreted as a contract for the sale of an ox (Vollgraff 1917). Ninety years later, due to some uncertainties in the dating of the text, the tablet was brought to the Centre for the Study of Ancient Documents (CSAD) at Oxford University to see if newly developed imaging techniques¹⁴ could solve the problem of the date once and for all. With the help of these new images the text was re-read and the crucial word 'bovem' ('ox' in Latin), indicating the ox sale was now re-read as 'quem' (the relative pronoun 'whom' in Latin) (Bowman *et al.* 2009). This led to a completely different interpretation of the text, in fact not centred on the sale of an ox at all, but possibly rather a document concerning a loan of money, a slave and the wife of a Roman military officer. The reinterpretation of a text is something that happens, even without the use of modern technology. This story illustrates how modern technology can enhance the visibility of a text beyond that which the human eye can see and how this in turn can lead to a complete

¹⁴ The Shadow Stereo method was used to capture images illuminated from different positions which enable image enhancement techniques such as the removal of woodgrain and other noise (Bowman *et al.* 2009: 4) For more on the shadow stereo method see Brady *et al.* 2005.

reinterpretation of a text.

2.1.2 THE PUBLIC SPHERE

More has been written about the process of publishing ancient documents than has ever been written about the reading process itself. While the process of reading is seen as something highly individual, with several equally effective approaches, the publishing of a transcription as a part of an edition brings it into the public sphere. In light of this, the community of readers of ancient documents continually try to encourage consistency in the publishing of editions.

Leiden Conventions	Name	Description
⸁⸂⸃	Uncertainty	Letters where a reading is possible but doubtful
.... , -10- , 𐀀10-	Illegible	Letters illegible by the editor and thus not read
[], [...], [-10-], [𐀀10-]	Gaps	Letters lost due to damage to the text
[abc]	Supplements	Letters supplied by the editor, which have been lost in the original text
<abc> , < >	Omissions	Gaps left by the scribe, sometimes filled in by the editor
(abc)	Solutions	Letters added by the editor to resolve symbols or abbreviations
{abc} , {...}	Interpolations	Letters superfluously added by the scribe, deleted by the editor
[[abc]] , [[...]]	Erasure	Letters deleted by in antiquity, possibly reconstructed by the editor
'abc'	Interlinear additions	Letters written by the scribe between lines

Table 2.4: Glossary of the terms used to describe the state of a text in the transcriptions of editions. Table compiled using van Groningen 1932b and Schubert 2009: 203.

The early published editions often contained hand-drawn transcriptions of the strokes seen on the document (i.e. facsimiles) (Ciula 2005: §6), but due to the huge

numbers of editions being published during the late 19th and early 20th centuries, it soon became necessary for most editors to publish typed editions which in turn required the development of methods to convey the state of the text such as gaps or illegible areas (table 2.4). For example, if the papyrus was damaged at the edges scholars would add a square bracket at the end of the line to indicate this, whereas previously this would have been clear from the drawing. As mentioned above, these editorial signs became systematised with the Leiden Conventions in 1932.

With advances in computer technology the editions became easier to compile and publish, but it was not until the Internet became more readily accessible that major changes in the way editions were published became possible. With this new medium came many new possibilities for accessing and searching editions, but with it also came the need for a suitable new convention for publishing. This time the epigraphers led the way with a system called EpiDoc (Epigraphical Documents in TEI XML)¹⁵. At the Epigraphy and Information Technology meeting in Rome in 1999 a group of epigraphers published a 'manifesto recommending the establishment of an online, free and unrestricted database of all surviving Greek and Latin epigraphical texts produced down to the end of Antiquity' (Elliott 2008). Inspired by this manifesto, in the summer of 1999 Tom Elliott published a description of his initial work on epigraphic encoding, which through collaborative work across the Atlantic became known as EpiDoc. EpiDoc now aims to be 'a mechanism for the creating of complete digital epigraphic editions and corpora' (Elliott 2008). As more and more research projects adopt this methodology for the digital publication of their ancient documents it could well become the "new Leiden Conventions" for the 21st century.

Nevertheless, it is important that the lessons learned over a century of printed publication are not forgotten. The aim with digital publication is not to change the content of editions, but to make them easier to publish, more accessible and more interactive for

¹⁵ EpiDoc (Epigraphical Documents in TEI XML) <http://epidoc.sourceforge.net/>, see 4.2.1 for a technical description of EpiDoc and a demonstration of how EpiDoc is used to encode the Vindolanda Tablets.

users (O'Donnell 2009).

2.2 INTERPRETATION PROCESS

The process of deciphering ancient documents is complex, and as mentioned earlier, it requires complex reasoning. The DSS will aim to store this complex reasoning, but in order to do this we first need to understand the reasoning process itself. Terras (2006) attempted to do this by studying three scholars reading ancient documents. By analysing the resulting material she was able to draw conclusions on the reading process with the aid of a set of ten defined reading levels. Building on Terras' work the eSAD project has developed the idea of a network of interpretations, which consists of some of Terras' levels and transitions between them (which I have called steps to avoid unnecessary confusion).

Terras (2006) used different knowledge elicitation techniques, such as Think Aloud Protocol (TAP), to collate explicit and quantitative data on three scholars' approaches to the reading of a stylus tablet from Vindolanda (see 2.1.1). She furthermore developed the 'levels of reading'¹⁶ as a system to enable a meaningful analysis of the TAP.

Terras' analysis concluded that the scholars mainly operated on three of the reading levels (fig. 2.1). These three reading levels are features of characters (level 1), identification of characters (level 2), and identification of possible words (level 4) (Terras, 2005). The DSS prototype is aimed at these three levels and the network of interpretation is a reflection of this.

Step 0 (fig. 2.2) is towards identifying elements such as lines, characters and spaces on the image. Step 1 is towards understanding the meaning of actual letters of the character element (the latter is the equivalent of Terras' level 2). Step 2 moves to the reading of

¹⁶ -1: Archaeological or historical context; 0: Discussion of physical attributes of the document; 1: Discussion of features of character; 2: Identification of possible character; 3: Identification of sequence of characters; 4: Identification of possible word or morphemic unit; 5: Discussion of grammar; 6: Meaning or sense of a word; 7: Meaning or sense of a group or phrase or words; 8: Meaning or sense of a document as a whole (Terras 2006: 49).

words or character sets (the latter correlates with Terras' level 4). Finally step 3 represents advance towards an understanding of a phrase or a sentence.

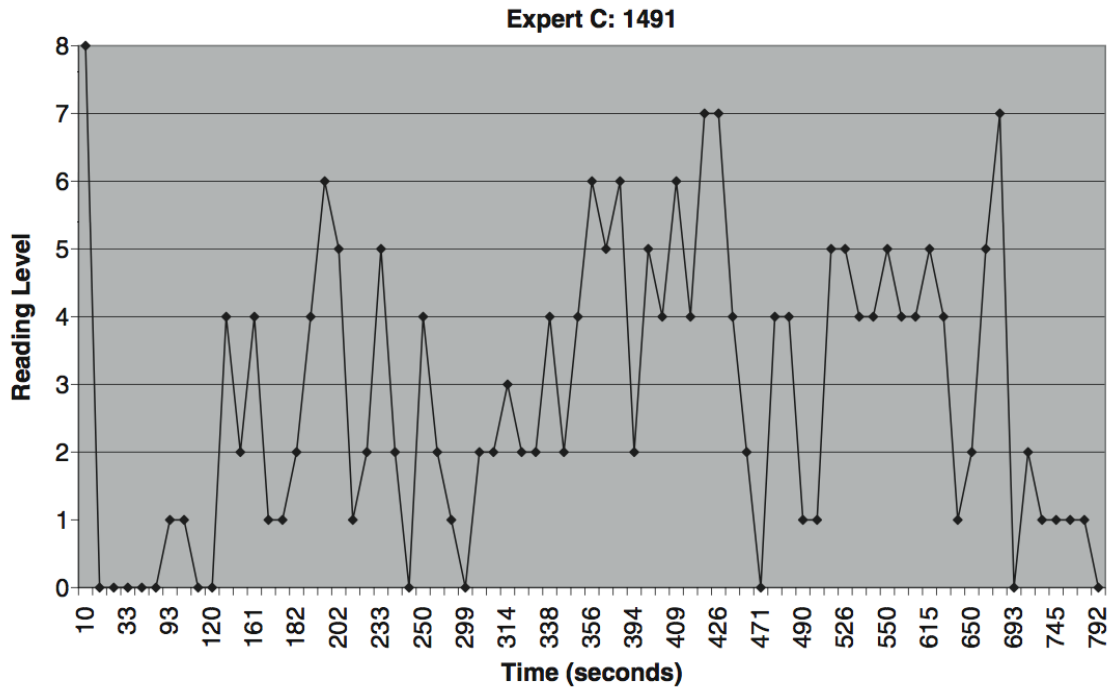


Fig. 2.1 Discussion of tablet 1491 by Expert C, plotted by reading level over the process of the reading (Terras 2005: 51). Used with permission of Melissa Terras.

The idea behind these steps is that a complete interpretation consists of a network of partial interpretations ranging from low-level, such as "These three stroke fragments are a character," (step 0), to higher level, such as "These five letters can make up the word *primo*," (step 2). However, this is an oversimplified way of looking at the interpretation process as there is a large amount of circularity inherent here. The scholars may begin by reading a couple of characters of which they are quite certain (step 1) and from there decide on the phrase (step 3) in which the word '*primo*' (step 2) would fit. This would then give them the other letters in the word (step 1).

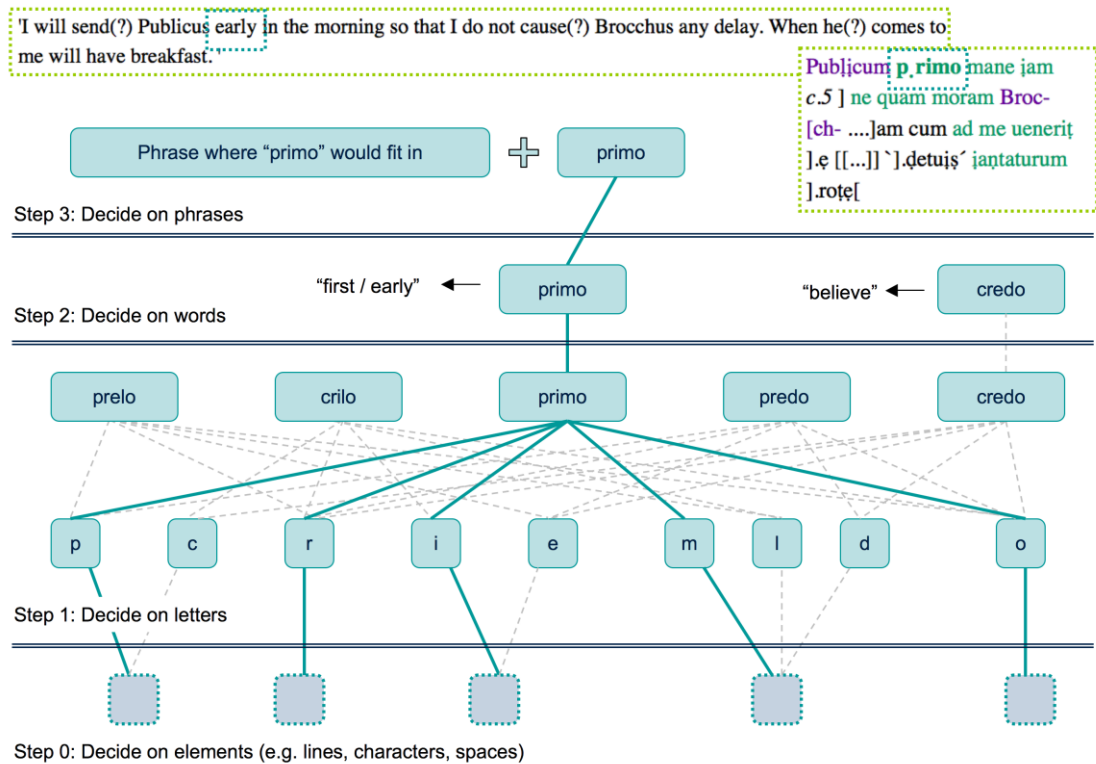


Fig. 2.2 Network of interpretations displaying steps 0-3. Step 0: Deciding on elements; Step 1: Deciding on letters; Step 2: Deciding on words; Step 3: Deciding on phrases.

The scholars will not necessarily move from the lower levels to the higher levels or vice versa in a regular manner, but will rather begin with a reading that they are confident about and then move up and down in a circular manner through the stages - a behaviour pattern identified by Terras (2005: 10) in her analysis of scholars reading ancient documents. It is therefore very important for this behaviour to be taken into account when developing the DSS prototype.

2.3 IT AND ANCIENT DOCUMENTS

Up to this point I have examined the history of reading ancient documents and the process itself. In this next section I will illustrate how IT can be used as an aid to this process. I will begin by looking very broadly at IT in society in general and then more specifically explore its use in the Humanities. I will then move on to focus on particular IT applications that can be used to aid the interpretation of ancient documents.

In the last couple of decades I believe that there has been a major change in society's attitude towards IT. These days most workplaces use computers in one way or the other to help employees with tasks that would otherwise be wastefully time consuming (Kurzweil 1990: 401-402). However, with more and more IT literate personnel in the workplace, I think that the awe that once surrounded the use of computers has gone. It is no longer acceptable to expect employees to change their working habits in order to accommodate the whim of a software programmer. Computers have lost their mystery and have become tools that must fit in with optimal work practices. This thesis is ultimately founded on this way of viewing IT tools as servants, not masters. I will not propose to suggest how the scholar should work. Rather I begin my thesis by examining how scholars are already working, and then move on to an assessment of the areas in which scholars could use a helping hand and how these needs could be met through the use of IT. Experience from the VERA (Virtual Environments for Research in Archaeology)¹⁷ project shows how 'the introduction of new technologies must be carefully managed and supported lest they alienate the very people they are supposed to be helping' (Warwick *et al.* 2009: 222). Archaeology is a very data rich field where IT plays a crucial role in collecting, storing, analysing and displaying these vast amounts of data (Lock 2003).

The idea with the VERA project was to examine how advanced IT can aid excavation and post-excavation by introducing new technology to the research excavation of the Roman town at Silchester and observing how it is used. One of the technologies introduced in this case study was the use of digital pens for the collection of data during the excavation. Despite issues with cost and the order of entry onto context sheets the conclusion was that digital pens could be an efficient tool for data recording. However, other hardware such as the Nokia N800 Internet Tablet was not popular among the archaeologists using it and the conclusion was that the tablet was not suitable for use in

¹⁷ <http://vera.rdg.ac.uk/>

excavation (Warwick *et al.* 2009).

The VERA project is a good example of just how important it is to develop and introduce new IT tools in cooperation with the user community in order for the result to be successfully integrated into the work processes of the user group.

It is also the case that since the spread of the Internet in the 90s it has become far more common for scholars in Humanities to incorporate IT into their research both as a tool and as a research subject in itself (Barrett 2005: 328). This aspect can be seen within the growing field of Humanities Computing / Digital Humanities¹⁸, which is based loosely on the idea of carrying out digital research within Humanities¹⁹. In the editorial for the first volume of Digital Humanities Quarterly the editors suggest that ‘Digital Humanities is by its nature a hybrid domain, crossing disciplinary boundaries and also traditional barriers between theory and practice, technological implementation and scholarly reflection’ (Flanders *et al.* 2007). Furthermore, the Digital Humanities Manifesto 2.0 suggests that:

Digital Humanities is not a unified field but an array of convergent practices that explore a universe in which: a) print is no longer the exclusive or the normative medium in which knowledge is produced and/or disseminated; instead, print finds itself absorbed into new, multimedia configurations; and b) digital tools, techniques, and media have altered the production and dissemination of knowledge in the arts, human and social sciences (Digital Humanities Manifesto 2.0 2009: 2).

From these two quotes it can be concluded that Digital Humanities is not a well-defined field of study. It is instead a very loose gathering of research that uses and masters digital tools within Humanities (Bodard and Mahoney 2010: 3). It also shows that IT research is no longer limited to the Computer Science Department or big software firms. Computers have been widely used in the Humanities from the early 1980s when existing technology was used to solve a variety of tasks (Waldrop 1985). There is a tendency for low budget departments like those in Humanities to use proprietary software originally developed for other higher budget departments or for commercial industries. Gregory

¹⁸ See Svensson 2009 for a discussion about the shift from using the term Humanities Computing to Digital Humanities.

¹⁹ See Borgman 2009; Hockey 2004; Schreibman *et al.* 2004; Svensson 2010 and Terras 2011 for an overview of Humanities Computing/Digital Humanities, its definition, history and future.

Crane suggests that smaller humanities fields such as classics do not have the resources to devote to creating their own IT developments and must therefore ally themselves with more resourceful informatics programs (Crane 2004). Interdisciplinary research is one of the ways in which these smaller humanities fields are able to gain access to computational infrastructure and resources (Terras 2010). However, with the introduction of open source coding and Web-based applications these same departments are now able to develop their own software without incurring enormous or prohibitive costs.

ARK²⁰ (Archaeological Recording Kit) is an example of an open source/Web-based application that can be tailored exactly to the needs of the user. Adaptability is the main principle of the tool and it has been developed in close co-operation with each of the projects it has been used for. I managed the setup of ARK for the Portus project²¹ and found that the continual development of the software on site in close co-operation with the users was an exciting process that was only achievable because of open source programming.

The most significant element of this thesis is the attention it pays to the development of IT applications to aid scholars, in close co-operation with the user group. The research is carried out in a Classics Department and not in a Computer Science Department. The DSS prototype is being developed under the close supervision of the main user group, the scholars who read ancient documents. In this way it is testing new ground, observing whether Humanities subjects can manage IT and exploit it for their own purposes, rather than change their work processes to fit the IT applications that are already available.

For readers of ancient documents, as in many other fields, IT has become an invaluable tool to aid their work. It is used to enhance images so that they are more readable (see 2.3.4); it is used to publish both online and in print (see 2.3.2); it is used to

²⁰ ARK (Archaeological Recording Kit) - <http://ark.lparcnaeology.com/>

²¹ Portus project - <http://www.portusproject.org>.

search through other documents or to look up words in online dictionaries (see 2.3.3). In other words it can aid the entire process of reading ancient documents from beginning to end. The fields of Epigraphy, Palaeography and Papyrology have now become truly intertwined with the digital media.

Scholars have been creating electronic texts for as long as there have been machines to read them, realizing early on that the calculating power of computers could be used effectively for processing textual data, for producing concordances and word lists, and for making comparisons between different versions (Deegan and Sutherland 2009: 80).

Father Roberto Busa (1913-2011) was one of the first to enter texts into a machine in order to construct the Index Thomisticus²², a concordance of all the words of Thomas Aquinas, the subject of his doctoral studies. In 1949 he visited IBM and convinced them to help him create the word index using IBM punch card machines (Bonzio 2011; Busa 1980).

Many large datasets have since then been made available on the Internet for experts and interested users alike to search through and use for research. Furthermore it has become increasingly acceptable, and even expected, that any project publishing a new dataset online will first encode its texts using an XML²³ standard like TEI²⁴ (Deegan and Sutherland 2009: 81). However, this produces new issues, as scholars cannot be expected to abandon their research and spend time sitting and encoding their editions in XML - even the newer generations of scholars are not trained for this.

Furthermore, it is important that we do not overestimate the usefulness of IT in the work of reading ancient documents. In their 2005 book on the subject, Alan Bowman and Mike Brady set out to ‘explicitly undermine the simplistic notion (perhaps now encountered less frequently than a decade ago) that the engineering scientist can offer the historian a range of off-the-shelf techniques of which one or another will provide an automated solution to problems in reading and interpreting visual data’ (Bowman and

²² Index Thomisticus is now available online - <http://www.corpusthomisticum.org>.

²³ XML (Extensible Markup Language) <http://www.w3.org/XML/>

²⁴ TEI (Text Encoding Initiative) <http://www.tei-c.org> See (Jannidis 2009) for an overview of the history of TEI.

Brady 2005: 1).

The following sections introduce some of the newer ideas and techniques for the use of IT to aid the interpretation process. In order to present similar tools in coherent groups and identify complementarities, they are divided into five subject headings: image annotation tools, textual encoding tools and methods, corpora and online resources, image manipulation and analysis and character recognition.

2.3.1 IMAGE ANNOTATION TOOLS

The development of image annotation tools is an area within Digital Humanities where extensive recent development has taken place. ‘Annotation is a crucial step in recording a reader’s encounter with a text, in developing an interpretation, and in sharing that interpretation with others’ (Juola 2008: 79).

Annotation has been used in different fields for sharing with collaborators what is detected on an image. There are currently a number of projects aiming to digitize textual documents through scanning and photography, and the need to digitally annotate these images is a natural consequence of this. Furthermore, there are great advantages in enabling the linking between transcriptions of text and images of the same text in online publication (Porter *et al.* 2009).

The BVREH²⁵ project is hosted by Oxford e-Research Centre (OeRC) and shares project investigator Alan Bowman with the eSAD project. The BVREH project began in 2006 with the development of a prototype, following on from which the project was then awarded a grant to carry out the pilot project, Virtual Research Environment for the Study of Documents and Manuscripts (VRE-SDM), completed in 2009, which aimed to provide researchers with a workspace where they could collaborate on documents through annotation and organisation. The project used documents and manuscripts as its case study,

²⁵ BVREH (Building a Virtual Research Environment for the Humanities)
<http://bvreh.humanities.ox.ac.uk/>

but built the VRE-SDM so that it can in the future be extended to other fields within Humanities (Bowman *et al.* 2010b).

The TILE project²⁶ has a similar approach, aiming at the creation of an environment that will enable annotation of images, and focussing on the development of a link between these annotations and encoded text. The project, a collaboration between MITH²⁷ (Doug Reside, Dave Lester) and Indiana University²⁸ (Dot Porter, John Walsh), has currently made available a web-based version of TILE (version 1.0), for use in creating and editing image-based electronic editions. TILE will be interoperable with other tools such as the UVic Image Markup Tool described below and will output TEI-compliant XML (Porter *et al.* 2009; Tile Website).

The UVic Image Markup Tool²⁹ is precisely what it says - an image mark-up tool. It was developed by Martin Holmes for the mark-up and publication of historic documents and maps at the University of Victoria, Canada. This desktop based system (only available for Windows) outputs TEI P5 XML.

The img2xml tool developed by Hugh Cayless outputs annotation graphics, formatted as SVG³⁰. It is being redeveloped for the DocSouth project³¹ at the University of North Carolina's Chapel Hill Library. The concept behind this tool is that the SVG formatted output can be linked to words in external documents, thus creating a link environment between annotations and text documents (Cayless 2008).

In summary, the projects concerned with developing image annotation tools, with the exception of the UVic Image Markup Tool, have mainly done so through Web based applications. The UVic Image Markup Tool and TILE have in common that they are both outputting TEI compliant XML.

²⁶ TILE (Text Image Linking Environment) <http://mith.umd.edu/tile/>

²⁷ MITH (Maryland Institute for Technology in the Humanities) <http://mith.umd.edu/>

²⁸ Indiana University Bloomington <http://www.iub.edu/>

²⁹ http://www.tapor.uvic.ca/~mholmes/image_markup/index.php.

³⁰ SVG (Scalable Vector Graphics) is a mark-up language for describing two dimensional graphics. <http://www.w3.org/Graphics/SVG/>

³¹ DocSouth (Documenting the American South) <http://docsouth.unc.edu/>

Furthermore, the img2xml tools and the TILE project are both concerned with linking annotated images and text. This works by linking the polygons, lines and points annotated on an image to a part of a text. If for example I wanted to publish an image of a document, side-by-side with the transcription of the document text I could use this technology to link each word in the image to the corresponding word in the transcription. This way, future users could click on the word in the transcription and in turn the polygon on the image where the word was read would be highlighted. Similarly, they could click on the lines I have drawn on the image where I have read a letter and the system would highlight the same letter in the text. This has great implications for publishing more intuitive corpora with greater user interaction.

As a possible method of inputting structure such as lines, words and characters, image annotation tools would greatly enhance any Decision Support System (DSS).

2.3.2 TEXTUAL ENCODING TOOLS AND METHODS

As mentioned above, the study of ancient documents has a long history of textual encoding beginning at the end of the 19th century with symbols and brackets added to typed publications to indicate the non-verbal nature of the text.

In her chapter on epigraphy for the book *Electronic Textual Editing*, Anne Mahoney suggests that encoding with something like XML is a natural development for epigraphers:

In this [Leiden] system brackets of various shapes, underdots and occasional other marks indicate letters added or corrected by the editor, letters on the stone that the editor judges to be wrong (for example, misspelled words), expansions of abbreviations, blank spaces on the stone, erasures, ancient corrections, and so on. Epigraphers are in this way trained to read and write semantically encoded texts, and therefore often find XML markup a natural extension of what they are already doing. (Mahoney 2006).

When editions were first published, the editors would sometimes produce facsimiles or drawings of each text. As the quantity of published texts grew larger, this procedure became less common, and a system of brackets and symbols known as the Leiden Conventions was developed to replace it. However, with the birth of electronic

publishing so also came the opportunity to add more meaning to the texts. Charlotte Roueché and Julia Flanders explain the emergence of electronic encoding in an introduction to epigraphers on the EpiDoc website:

At first the emphasis was on inserting formatting instructions, but there soon emerged methods of including more complex semantic information concerning document structure and even content. A simple example is marking up a book title as a title, rather than simply marking it as being italicised. The use of this more abstract markup permits a separation of structure and presentation, where structure is comparatively fundamental to the document's genre, while presentation may be varied depending on the form of publication. ... For example: missing material can be marked up as such, and then presented within square brackets; at the same time, a search can be instructed only to interrogate text which has not been marked up as missing (so only definitely attested terms) (Roueché and Flanders 2006).

SGML is a very well known type of electronic encoding, which in 1986 became an ISO standard³². As a derivative of SGML, XML too is 'used as a metalanguage to define most of the actual markup languages in use today' (Schmidt 2010: 339). In other words SGML and XML are not in themselves mark-up languages but are meta-languages used to formulate mark-up languages such as HTML³³. The concept behind XML is to have tags surrounding the encoded content; the beginning tag uses the format `<tagname>` and the end tag uses the format of `</tagname>`. Another principle of XML is that tags can have attributes in the beginning tag such as `<tagname attributename="attributevalue">tag content</tagname>`. Furthermore tags can be nested as in the following example:

```
<tagname>
    <nestedtagname>nested tag content</nestedtagname>
</tagname>
```

The naming and structure of tags depend on the schema³⁴ used to define it. One of the most known encoding languages is HTML, which is used for tagging browser content. The most relevant schema for the study of ancient documents is TEI, which is currently the most

³² SMGL (Standard Generalized Markup Language) ISO 8879: 1986, http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=16387.

³³ HTML (Hypertext Markup Language) <http://www.w3.org/MarkUp/>

³⁴ An XML schema is a document that defines the content, structure and vocabulary that can be used in a certain type of XML document.

popular method of encoding electronic texts within the Humanities academic community (Deegan and Sutherland 2009: 81).

There are two essential goals motivating the development of the TEI. The first is to enable scholars to represent their research materials in digital form using a descriptive language that mirrors the kinds of analytical terms and concepts that are familiar and essential to humanistic scholarship. The second goal is to enable scholars to share the resulting materials intelligibly, by using a common descriptive language. (Roueché and Flanders 2006).

EpiDoc is a customisation of TEI that ‘removes irrelevant elements from the main body of the TEI, and it adds provisions for the specific kinds of transcription, analysis, and metadata that are essential for epigraphic work’ (Roueché and Flanders 2006).

There are already several projects aimed at digitising corpora that have used EpiDoc as a method of encoding. Some of the earliest projects to include this encoding method are the Inscriptions of Aphrodisias³⁵ (Bodard 2008) and the Vindolanda Tablets³⁶. Other major corpora, such as the Duke Databank of Documentary Papyri (DDBDP)³⁷, are already published online, but much work is being invested in at least providing them with an EpiDoc XML output.

The Leiden Conventions are now widely used for inscriptions and papyri alike. These conventions were primarily developed as a simple way of illustrating different features of a text in a printed edition. Square brackets and under-dots do not take up much room on the printed page. However, these symbols are not meaningful in themselves and without a convention glossary (table 4.1) they are impossible to decipher.

The under-dot has always been very confusing to apply as it represents one of the most difficult issues in the reading of ancient documents – uncertainty (Youtie 1966). When is a reading uncertain? In some editions the under-dot is used to signify broken or uncertain letters while in others it only indicates pure uncertainty. In the 1930s some

³⁵ The Inscriptions of Aphrodisias (InsAph) are available into two online publications: ala2004 (Roueché 2004) and IAph2007 (Reynolds et al. 2007) - <http://insaph.kcl.ac.uk>.

³⁶ The Vindolanda Tablets are currently published as: Tab.Vindol.I (Bowman and Thomas 1983), Tab.Vindol.II (Bowman and Thomas 1994), Tab.Vindol.III (Bowman and Thomas 2003) and Tab.Vindol.IV (Bowman *et al.* 2010a).

³⁷ <http://idp.atlantides.org/trac/idp/wiki/DDBDP>

scholars experimented with the idea of underlining letters that were certain, even though they were damaged (Hunt 1932). The Leiden Conventions uses the following formula for the meaning of the under-dot: ‘Letters about the reading of which there is a genuine doubt or that are so mutilated that without the context they might be read in more than one way’ (Schubert 2009). This can be interpreted as letters that are uncertain regardless of their damage, but not letters that are damaged and can still be clearly interpreted even without their context. In other words letters that have been damaged, but can clearly be read independently of the reading of the letters surrounding it are not uncertain and would therefore not be under-dotted.

Should characters that are damaged or faint not be marked up? Hunt (1932) thinks that this sort of detail is insignificant. However, broken and faint letters had at one time been indicated, when producing epigraphic facsimiles, by drawing the part of the letter that could be seen, or drawing the shape of a faint letter with dots (Pritchett 1955).

XML offers a solution for this, in that it can be used to semantically encode transcriptions with meaningful tags instead of meaningless symbols and brackets. EpiDoc was developed to reflect common practice such as the Leiden Conventions and uses the `<unclear>` tag to signify letters that are doubtful (Elliott *et al.* 2007). In EpiDoc, as in the Leiden Conventions, there is no tag to represent damaged letters that are confidently read. However, due to the nature of XML the editors could add a `<damage>` tag to their XML editions. The edition would thus not conform to either EpiDoc or the Leiden Conventions.

Nevertheless, even though most modern editions conform to the Leiden Conventions they may still have a different interpretation of what the under-dot signifies. Through XML these differences could be made explicit instead of being hidden, thus undermining the unity that van Groningen sought.

In the same way that the system of symbols and brackets was developed for printed editions, encoding editions with XML enables a new way of publishing digitally, allowing

information about specific words to be added to that particular word tag. This can then be used to create an index or a knowledge base of words from a corpus (Roued-Cunliffe 2010). XML encoding makes it possible to analyse ancient documents in new ways and use technology such as Web Services³⁸ to create interoperable knowledge resources (Roued 2009).

However, despite the wide range of applications in which XML has been adopted in the last decade there is still the issue of interoperability that has not been solved yet. Libraries are a good example of a field in which XML has had widespread use but where, due to both technical and social issues, there is still a lack of interoperability between resources (McDonough 2009).

XML encoding will play a big part in this thesis both as a part of the Decision Support System (DSS) and as a part of the tools that can be used to extend the DSS. In the DSS prototype TEI Encoded XML will be used to output the structure of the document (see 3.3). XML is furthermore a vital part of the Web Service, APPELLO (see 4.3).

2.3.3 CORPORA AND ONLINE RESOURCES

Corpora of editions of Greek and Latin texts were first published in the mid 19th Century. Among the early corpora is the aforementioned Corpus Inscriptionum Latinarum (CIL). Since that time, these corpora have been used to provide search material for scholars to find comparable examples for the texts they are reading. Internet technology has enabled the online publishing of these corpora in such a way that they can now be searched through browser-based applications. This has had great implication in terms of access to ancient documents both for non-scholars and scholars from institutions where access to traditionally published corpora is limited (Bodard 2008).

Digital corpora furthermore offers the opportunity to use computational and quantitative tools for corpus-based analysis, which in turn provides new insights into how

³⁸ See 4.3 for an introduction to and demonstration of Web Services.

language is structured and used in the text material (Biber *et al.* 1994).

Some of the larger corpora which are currently available online are the Duke Databank of Documentary Papyri (DDBDP), the Advanced Papyrological Information System (APIS)³⁹, the Heidelberger Gesamtverzeichnis der griechischen Papyrusurkunden Ägyptens (HGV)⁴⁰, and Trismegistos⁴¹. These can all be accessed through Papyri.info⁴² and the Perseus Digital Library⁴³.

The DDBDP project began in 1982 and is an electronic corpus of published Greek and Latin documents. It was first available on a CD and more recently as a part of the Perseus Digital Library. It has now been relocated to Papyri.info and is moving towards open-source standards based developments for improving the corpus and access to the corpus (IPD Wiki).

APIS is more of a collections-based repository where it is possible to search and retrieve information about and images of papyrological editions. This dataset connects to collections all over the world (APIS Website 2011).

HGV is a corpus of Greek documentary papyri from Egypt. This corpus is based in Heidelberg and the metadata is therefore available in German.

Trismegistos is a portal to papyrological and epigraphical material from Egypt. This dataset includes non-Greek/Latin text such as Hieroglyphs and Coptic. The collection is built on the Leuven papyrus collections and includes a topographical database (Trismegistos Website).

Planning for the Perseus Digital library began in 1985. It contains a variety of texts (e.g. Greek/Latin, Arabic, Germanic, and 19th Century American) and focuses its research on general questions such as, ‘What happens when libraries move online?’ (Perseus Website).

³⁹ <http://www.columbia.edu/cu/lweb/projects/digital/apis/>

⁴⁰ <http://www.rzuser.uni-heidelberg.de/%7Egv0/>

⁴¹ <http://www.trismegistos.org/>

⁴² <http://www.papyri.info/>

⁴³ <http://www.perseus.tufts.edu/hopper/>

Many individual projects also use an online medium to publish editions; the Vindolanda tablets project is a good example. Chapter 4 describes in detail both the old (i.e. VTO) and the new (i.e. VTO2) online publications of the Vindolanda Tablets.

Online corpora such as these play a big role in the DSS as a means of providing scholars with suggested words for their interpretation. Chapter 4 demonstrates how the Perseus Digital Library can be searched through the DSS prototype to provide suggested words for a particular pattern of characters.

2.3.4 IMAGE MANIPULATION AND ANALYSIS.

Photography has long played an integral part of the study of ancient documents (Bearman and Spiro 1996). The ability to photograph inscriptions and other documents has provided a way for texts to be seen by scholars who are not able to inspect the original. Photography has furthermore provided less destructive methods of enhancing writing that is otherwise illegible⁴⁴.

An example of this is infra-red photography. This technique is used in difficult cases where the contrast between ink and the medium it is written on is low (Bagnall 1997; Smith 1938). The Dead Sea Scrolls are an example of soft media texts⁴⁵ where in some cases infra-red photography was the only method of extracting information from the documents. Infra-red photography has also proven very effective on texts written with carbon based inks such as the Elephantine papyri from the 5th century BCE. Unlike newer inks, carbon based inks do not penetrate and dye the writing surface and is therefore prone to flaking off. However, the carbon does adhere to the micro-structure of the material and it is this that can be detected with infra-red photography (Bearman and Spiro 1996).

Digital imagery has brought many advantages to the study of ancient documents;

⁴⁴ There have been early attempt at reading largely illegible texts with the aid of chemical methods. Unfortunately, such methods can have a very destructive effect on both the text and the media (Benton et al. 1979).

⁴⁵ Soft media texts are texts written on media such as untreated animal skin, leather, vellum and papyrus (Bearman and Spiro 1996).

higher quality images can now be shared online or via email in a matter of minutes. Furthermore, it is now possible to produce as many images as needed of an item without the expense that previously came with traditional photography.

Multi-spectral imaging is an advanced form of digital imagery, which like infra-red photography can be used to enhance images by using infra-red and other spectral bands and conducting pixel analysis on these images.

For example, on a digital image of an ancient document each pixel has a different spectral signature depending on whether the pixel is a part of the writing or the medium. These differences are enhanced by multi-spectral imaging to make the document more legible. It is a technique most often used by geography and geology where the images are recorded by remote sensing aircraft (Bearman and Spiro 1996; Chabries *et al.* 2003; Gonzales and Woods 2008).

In 2003, using the concept of holding a tablet up against light sources from different angles to better see the incisions in stylus tablets, a new technique involving casting angled low level light on a object, the Shadow Stereo method, was developed to record similar information through photography (Brady *et al.* 2005). Shadow Stereo involves taking multiple photographs of the tablet in the same position, but with the light source in at least two different positions. This results in the shadows of the incisions moving, dependent on the direction from which the light shines on the image. Stains and other marks on the images that are not incised will not have a shadow and therefore will not move if the direction of the light source is altered. The movement of the shadow between the two images allows the detection of edges and the identification of actual incisions on the tablet (Bowman *et al.* 1997; Molton *et al.* 2003; Pan *et al.* 2004; Schenk 2001; Schenk and Brady 2003).

In 2004 Xiao-Bo Pan completed a project at the Department of Engineering Science, Oxford University, which used various image enhancement techniques to increase

the legibility of otherwise partially illegible stylus tablets from Vindolanda (Pan *et al.* 2004). Using the images taken with the Shadow Stereo technique the project applied a homomorphic filter⁴⁶ to remove shading and correct uneven illumination. Using existing knowledge that wood grain usually occurs in predictable regular intervals across a tablet, the next stage was then to remove the wood grain from the image. Through this process of correcting the illumination and removing the wood grain the actual writing/incisions on a tablet become much more visible.

In 2010, the RTISAD project was established to explore how Reflective Transformation Imaging (RTI) or Polynomial Texture Mapping (PTM)⁴⁷ could refine the implementation of Shadow Stereo imaging. By making a PTM of an object or a tablet it becomes possible to visualise the same object's 3D attributes to an extent that may not even be possible with the human eye.

‘A PTM can be very easy to create. In its simplest form, you take a sequence of photographs of an object from above, in a dark room. The camera is fixed but in each photograph you light the object from a different direction’ (Goskar and Earl 2010: 28-29).

Advanced images created with PTM have since been subject to analysis similar to those from the Shadow Stereo technique. As a part of the eSAD project these new images have undergone illumination correction and wood grain removal (Tarte *et al.* 2011). Tarte experimented with a sample of 18 images from a PTM session⁴⁸. As a part of the ENGAGE project these techniques have since been incorporated as a Web Service into the VRE imaging browser (Tarte *et al.* 2009).

The next step in this process is to identify features on an image. Pan explored feature detection using the concepts of phase congruency⁴⁹ and local energy to detect otherwise non-visible features in an image (Pan *et al.* 2004).

⁴⁶ A homomorphic filter is an image enhancement technique that normalises brightness and enhances contrast on an image.

⁴⁷ See Earl *et al.* 2011 for an overview of RTISAD, PTM and RTI.

⁴⁸ There are typically between 28 and 56 images in each PTM session taken from different light angles (Tarte *et al.* 2011).

⁴⁹ Phase congruency is an edge detection method that uses the frequency of the image.

Examination of high quality digital images plays a crucial part in the actual reading of ancient documents. While re-interpreting the Frisian tablet, mentioned above, the readers, Alan Bowman, Roger Tomlin and Charles Crowther referred frequently to a set of images captured and enhanced with the technique of Shadow Stereo or phase congruency. The ability to view the incised text with shading from different angled light sources helped the scholars to see more clearly incisions that would otherwise have been impossible to see, even on the original tablet. The scholars also used images that had undergone illumination correction and wood grain removal, and these further assisted in making visible, writing that would otherwise be illegible.

Both the high quality images from Shadow Stereo and PTM, along with the analysed images, can be a great addition to a DSS in the input phase, especially if combined with image annotation tools.

2.3.5 CHARACTER RECOGNITION

Following on from the previous section on image manipulation and analysis I will proceed to the subject of character recognition. The field of machine character recognition is well known as OCR – Optical Character Recognition - which is the electronic translation of an image into typed text or audio. OCR has a history that goes back to the first half of the 20th Century. In 1949 the Radio Corporation of America (RCA) began experimenting with a reading machine that was able to read typed text, both letter-by-letter and a small number of entire words. Even though the machine was laborious to run and very expensive to build it did demonstrate that it was possible to automate the task of reading (Mann 1949).

Work on OCR is ongoing, but the interesting aspect of character recognition for this thesis is the more challenging task of handwriting recognition. Handwriting can be difficult for even humans to recognise as the visual patterns used to recognise words in printed text are missing (Oakhill and Garnham 1988). The context of the character (i.e. the

characters surrounding it) plays an important role as the perception of a character is often facilitated by the word in which it can be found (McClelland and Rumelhart 1981). If it is difficult to recognise the word it becomes even more difficult to recognise the individual characters.

Today it is possible to buy digital handwriting recognition pens at a very reasonable price, and the VERA project has experimented with the use of these pens for archaeological onsite recording (Fulford *et al.* 2010; Warwick *et al.* 2009). In ancient documents, however, the character recognition technique that might be used must inevitably differ, as the characters are not recorded while being written and the handwriting can not always be sampled in advance.

Forensic document analysis is probably the most known field in which handwriting identification is applied. It is a field in which the methods and results must necessarily be very transparent in order for them to be accepted as evidence in court. The field of forensic handwriting recognition has now developed computational methods that can compare handwriting and output statistically meaningful results, even though it is not advisable to use these results on their own without human interpretation (Schomaker 2007; Srihari *et al.* 2008). Although some of the same techniques used in this field have been tested, on for example medieval documents, they cannot be used uncritically without further improvements (Stokes 2009). Furthermore, Tom Davis (2007) believes that there are fundamental differences between the ancient document analysis and forensic document analysis that make it difficult to use the same tools for both fields. While ancient document analysis operates with a lack of information about the subject being studied, forensic document analysis operates instead with almost too much information where some of it will be deliberately untrue. In other words forgery plays a much larger role in forensics than it does in the field of palaeography (Davies 2007: 268).

The postal service is another area in which automated handwriting recognition is

applied frequently. For a postal sorting office there is a huge advantage in automated reading, especially of the post code/ZIP code part of an address. There are two steps in this process. The first is to locate the address on the posted item and the second is to read the address (Gaceb *et al.* 2008; Srihari 1993).

Many character recognition systems require a dataset of letter-forms against which to compare recorded features. These datasets of letter-forms would need to be relevant to the context and time period in which the text is presumed to have been written. Additionally, letter-forms found in any text are influenced by the material on which the text was written, and even the material of the tool used to write the text.

During the course of her D.Phil work, Terras (2002) developed a knowledge base of letter-forms from the Vindolanda Tablets to use with the Ground Reflective Adaptive Vision Architecture System (GRAVA) described below. This consisted of a dataset of characters where each one was recorded as a graphic representation in SVG (fig. 2.3) and labelled with a set of predetermined characterisations. The SVG record could then be used as the letter-form dataset to which other unknown characters could be compared. The labels were used to sort the character sets. As an example, *S* might be encoded as [**s*, SHl, SWl], where **s* identifies the letter as a small *S*, SHl represents SizeHeightLarge, and SWl represents SizeWidthLarge, describing the height and width of this particular version of *S* as large. Examining the strokes of the first *S*, the first stroke is encoded [DCSdlcr, Ll, Wa, PDI] and the second is encoded [DSur, La, Wa, Par]. The first stroke has a Simple Curve -> Down to Left -> Curve Right (i.e. DCSdlcr) and the second stroke is Straight -> Up Right (i.e. DSur). The first stroke is long and average in width and is placed with the descender below left on the line (i.e. Ll, Wa, PDI), while the second stroke is average length and average width and is placed with the ascender above right (i.e. La, Wa, Par) (Terras 2006: 109, 200-204).

The GRAVA system, developed as a part of Paul Robertson's D.Phil (Robertson

2001) was used by Terras to produce plausible interpretations of characters in an ancient document based on a description of their features (Terras and Robertson 2005).

... the GRAVA system is unique in that it uses description length as a unifying mean of comparing and contrasting information as it is passed from one level of the system to another. 'Description length' is calculated from the probability of an input such as the data describing the shape of an unknown character, matching a known model. By adding the description lengths of the outputs of different agents, it is possible to generate a global description length for the output of the whole system. [...] Therefore, the smallest global description length generated from many runs of a system is most likely to be the correct solution to the problem: the minimum description length or MDL (Terras 2006: 125).

Terras concludes that the GRAVA system 'provides the means to implement a system that works in a similar manner as the expert, generating possible, reasonable, solutions to interpretation problems.' (Terras. 2006: 151). Note however that this is again not an expert system providing an automated reading of a text. It is instead a tool that aims to provide a plausible interpretation of a character that then requires human interpretation (Terras and Robertson 2005).

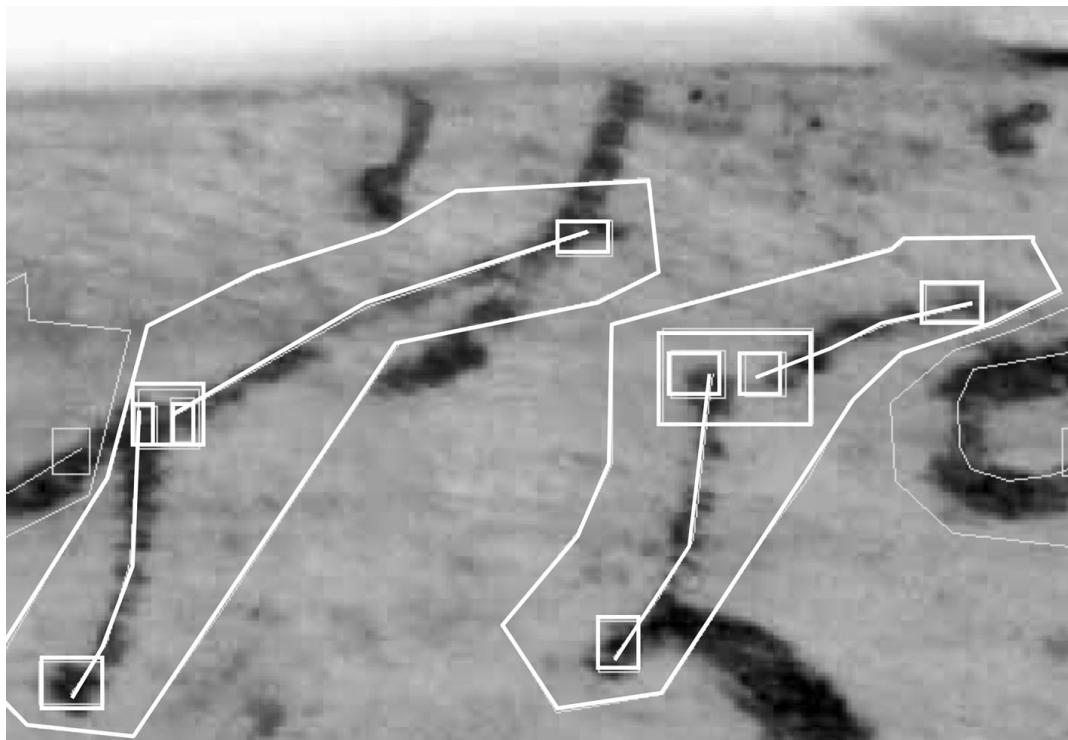


Fig. 2.3 Annotation of the letter S from Vindolanda tablet 225 in Terras image annotation viewer (Terras 2002: Multimedia CD). Used with permission of Melissa Terras.

Therefore even though a system such as this could tell us that a Latin cursive character is a t, p, or c (which may all appear very similar in individual cursive hands),

only a scholar with palaeographical and linguistic experience would be able to judge which one of the characters is most likely to be correct, based on the context.

In 2009, Martin Moran, as a part of his Engineering Science undergraduate project (Moran, 2009), developed an ontology for ancient documents. His aim was to complete the broken strokes that form a character and to develop a recognition system that, based on a combination of strokes, could make a suggestion for an interpretation of a letter (Moran 2009: 5).

Peter Stokes is moreover developing a prototype that provides a framework for the analysis of medieval scribal hands. The aim of this prototype is to answer the question: “were these different texts written by the same person?” using computer technology as an aiding tool. As a part of this project Stokes has experimented with the use of automatic handwriting recognition using feature extraction and automatic clustering (Stokes 2007). Stokes furthermore takes great care to make it clear that any result from any such automated tool must always be subject to subsequent human interpretation (Stokes 2007; Stokes 2009).

The use of digital technology to automate the study of hands on ancient documents has also been attempted on Greek inscriptions (Tracy and Papaodysseus 2009) and other palaeographical material (Ciula 2009) with some success.

Character recognition would be a very useful addition to a DSS in the same way as a word search tool would be. It would enable the DSS to suggest characters based on high quality imagery. The scholar could then use these suggested characters as an argument to support the reading of a particular letter. However, the DSS prototype will only include a word suggestion tool and will not deal with character recognition as such. Nevertheless, the idea of character recognition is included amongst the suggestions for a future DSS development in the conclusion (Chapter 5).

2.4 COMPUTERS READING ANCIENT DOCUMENTS.

When discussing the use of IT to support the reading of ancient documents, some would understand this as an automated system into which an image of a document could be input resulting in a fully formed decipherment and interpretation, perhaps even without human interaction. Such an application would fall within the field of Artificial Intelligence (AI)⁵⁰.

For two principal reasons this is not the type of application I am seeking to research and develop here. Firstly I do not believe that this type of research into automated reading can benefit the study of documents generated in Graeco-Roman antiquity, in which this thesis is based. It is my impression that scholars need not or do not have any interest in developing robots to do their work for them⁵¹.

Good reading is a reflective as well as a sensational process which engages our abstract mental faculties, and we give it up at our peril (Deegan and Sutherland 2009: viii).

In other words, we need to think carefully before attempting to remove humans from the process of interpretation (Bell 1985).

Secondly, I do not think that it is possible to develop an automated interpretation system that can truly interpret an ancient document, as this act requires human intelligence. Interpretation is a part of the ‘higher levels of analytic and creative functions which are the prerogative of the human mind’ (Busa 1980: 89).

The failure to distinguish between the symbol-processing powers of a computer and the human intelligence of its operator is at the heart of many misconceptions as to the role of information technology in society (Jonscher 2000: 149).

Charles Jonscher (2000: 61, 121) operates with a data-information-knowledge hierarchy where data consists of raw signals and symbols, information consists of facts

⁵⁰ AI (Artificial Intelligence) See Feigenbaum and McCorduck 1983; Kurzweil 1990; Russell and Norvig 2010; Turing 1963 for a discussion about the difference between artificial intelligence and human intelligence and whether or not machines can think.

⁵¹ This is instead an interesting line of research in a computer science or engineering setting, mainly from an Artificial Intelligence perspective, to investigate how close computer software could come to replicating or modelling the intrinsic thought patterns of a human being. See Russell and Norvig 2010 for an overview of the use of AI in different lines of research.

distilled from data, and knowledge consists of ideas, thoughts and beliefs distilled from information. Computers are more than capable of operating with data; in fact they are faster and more efficient than humans at this task. Computers can also simulate the mental activity of information processing for the solving of computable problems, but they are not able to operate with knowledge. They cannot distil ideas, thoughts and beliefs from information. Simply put, they cannot think.

I am a proponent of the weak AI hypothesis, which states that machines can only act *as if* they were intelligent. This is in opposition to the strong AI hypothesis that machines do not merely simulate thinking but are in fact thinking (Russell and Norvik 2010: 1020). If one believes in the weak AI hypothesis, the idea of whether or not machines can conduct interpretations instead becomes a question of whether machines can act as if they can conduct interpretations. I am not qualified to judge whether or not this is possible now or will be in the future, but I do think it is an irrelevant, albeit interesting, question in the context of this thesis.

When scholars deliver a result in the form of an interpretation this reading can be evaluated based on what is known about their prior experience and skill (Siemens *et al.* 2009). An interpretation made by a professor of 30 years standing is more likely to be believed than that of an undergraduate student, but it does not necessarily follow that the professor is correct as interpretation is by its very definition not an exact science and all such ‘acts’ in Humanities subjects are, as the name implies, just interpretations. It can merely be understood that ‘professorial experts’ are more likely to give a more soundly based interpretation, taking all factors into consideration on the basis of their wider knowledge, than an undergraduate student is.

However, when faced with a result produced by an automated system how could this reading be evaluated? We might put the results through the Turing Test⁵² to see if the

⁵² The Turing Test is based on a paper by Alan Turing called *Computing machinery and*

system produces a result that looks and behaves as if it was produced by a human being. However, this does not tell us whether or not we can trust the result. Margaret A. Boden argues that:

We take it for granted that human beings, including experts (perhaps *especially* experts), can be mistaken or ill advised about any of these three aspects of thinking. We must equally take it for granted that computer programs – which in any event are far less subtle and commonsensical than their programmers and even their users – can be questioned too. (Boden 1992: 453)

Some might take this advice as far as not trusting computers at all, and will discard altogether any result from an automated system. Even worse, as Boden fears, some might be inclined to accept any result from an automated interpretation because they assume that a computer will be able to deliver the ‘correct’ answer⁵³, since a computer system may be considered to be less error-prone than a human being. Results from an automated system might be more **consistent** than results from most human beings, but they could equally be consistently **wrong**. Because of the consistency of its results, a computer system could all too easily come across as objective and thus more reliable than a human being, but its interpretations still carry the human bias that was input at the beginning - they are the results of whatever information the computer initially received. In contrast to the results of human beings it is difficult to evaluate these interpretations.

Stokes (2009: 325) makes an interesting point here as to the advantage of open source software that, as opposed to proprietary software, can be more easily deconstructed in order to understand what it does. This allows for much more clarity about the results output by a computer.

I have outlined above the reasons why I do not see the ultimate goal of this thesis to develop an automated interpretation system. The DSS that is being developed has a very

intelligence (Turing 1950). Turing suggested that instead of focusing on whether a machine can think or not it would be better to subject it to a test to see if it can behave intelligently. If the machine can act intelligently and a judge is not able to see the difference between the machine’s results and those of a human then the machine must be intelligent. However, ‘programs have yet to fool a sophisticated judge’ (Russell and Norvik 2010: 1021).

⁵³ See Bell 1985 for issues involving user acceptance of expert systems such as ‘lack of trust’ or ‘too much trust’.

different agenda, in that it will instead focus on supporting the scholar who is interpreting ancient documents by doing the tasks that the human brain finds difficult, but which computers can be programmed to do much more rapidly and with greater ease (Bell 1985).

2.5 COMPUTER BASED TASKS

In the previous section it was established that this research is not focused on computer programs that can replicate the ultimately human function of interpretation. However, there are other tasks involved in the interpretation process which humans find more difficult. A machine could be programmed to do these tasks relatively easily and it would be better suited than a human being to perform these tasks.

This section will look at five such tasks and illustrate how they can be resolved with the help of the systems developed in relation to this research.

The first task is remembering and recording complex reasoning (see 2.5.1). As mentioned above, complex reasoning forms an integral part of the reading of ancient documents and an aspect which humans are much better equipped than computers at dealing with. However, when it comes to remembering their complex reasoning process and organising it for later use computer application can be very useful.

Furthermore, the human brain is not equipped to search through huge datasets with the same speed or focus as a computer-based search facility (see 2.5.2). As research increasingly crosses national borders, it is also becoming more important to support collaboration with colleagues overseas (see 2.5.3). Furthermore, new technology is setting new standards for the publication of editions of ancient documents (see 2.5.4). In addition to paper publishing in the form of a monograph there is also often expectation of publishing on the Internet. Formatting editions to fit these different publishing formats is a tedious chore and is another area where purpose-built software can be very useful.

Finally, the use of imaging technology is now at a stage where, with the help of mathematical analysis, it is possible to extract from images data that would otherwise be

impossible to see with the naked eye (see 2.5.5).

2.5.1 REMEMBERING COMPLEX REASONING

Today most scholars who work on interpreting ancient documents have other tasks to perform alongside this work. If they work at a university they may need to teach or attend meetings. They may be working on several different documents at any one time. Their interpretation of a single document might be frequently interrupted by numerous other tasks throughout the week. A likely scenario is one of a scholar sitting down to read an ancient document on a Monday. After much internal debate the scholar settles on a certain reading of the document. The scholar then proceeds to a week full of other unrelated work and is not able to return to this document until the Friday.

During the week a colleague mentions something that makes our scholar reconsider the interpretation that came out of Monday's session. However, since Monday the scholar has forgotten the reasoning that led to certain conclusions, and so the scholar decides to scrap the old reading and adopt the new one. In this scenario our scholar could have made good use of a computer system into which could be input not only the conclusions of the interpretation but also the reasoning that led to these conclusions. Then on the Friday the DSS system would prompt the scholar with a reminder of the previous reasoning, giving the scholar an opportunity to evaluate the new information in the light of the old.

This scenario highlights the precise purpose of the DSS, which is to have the ability to store, present, and remind scholars of their previous reasoning process. Scholars have the ability to create these very complex chains of reasoning, but they may not have the capacity to remember them instantly with absolute accuracy after a week of unrelated work.

Even published editions of ancient texts are constantly being refined and re-interpreted. The re-interpretation of the so-called "Frisian Ox" tablet is a very dramatic example of this (Bowman *et al.*, 2009). From the time of the first reading in 1917

(Vollgraff, 1917) the date of the tablet has continually been open to question. It was therefore suggested that with the help of the imaging techniques developed at the CSAD and Department of Engineering at University of Oxford scholars would be able to get a clearer picture of the text and would be able to solve the dating problem once and for all.

I will use the text in line 4 on the front of the tablet as a demonstration of a chain of reasoning. In Vollgraff's original transcription this reads:

Line 4: **RITE UTI L(ICET) BOVEM** (Vollgraff, 1917)

The last word (i.e. *bovem* – Latin for 'Ox') gave the name to the tablet. The re-interpreted text reads:

Line 4: **RIT AUT AD QUEM** (Bowman *et al.*, 2009)

This is a very different interpretation, completely removing the word *bovem* and giving a whole new meaning to the tablet. Tarte (2011a) has examined the chain of reasoning that has evolved from the previous interpretation to the current. She observes that the real shifting point in the interpretation process was when the scholars realised that the letter before *D QUEM* is *A*. The scholars argue that this letter looks like a different type of *A* than they would have expected in a text from this period, though they do nevertheless agree that it is indeed an *A* (Bowman *et al.* 2009).

In the example above this chain of reasoning unfolded within one interpretation session. However, if we were to imagine the same scenario unfolding over a series of sessions some of which might have been conducted remotely by the three scholars working on this reading in different locations, the result might have been very different indeed. In a situation like this it would be advantageous to have the facility to share not only the conclusions drawn by each of the scholars reading the text, but also share the arguments that lie behind the process of working through the interpretation.

2.5.2 SEARCHING HUGE DATASETS

The second issue faced by scholars is the task of searching through huge datasets for parallels that can help them decipher their text. An increasing number of these datasets are currently becoming available online, enabling scholars to search through them with much less effort than previously. Many of these online textual corpora are now searchable collectively through a handful of central portals⁵⁴. This is one of the most important areas where digital technology has made a significant impact on the work process of scholars reading ancient documents.

However there are still many areas in which these huge datasets can be optimised for better use. Chapter 4 approaches one of these areas, namely enabling the data to be searchable through a Web Service. By doing this, the dataset can be made available not only in a human readable format (e.g. HTML) but also in a computer readable format (e.g. XML). A Web Service enables the user to send off a request to the dataset through the URL.

The Web Server receives the request and returns a dataset based on the parameters sent through the URL. This dataset can be returned in different formats, but a favoured format is XML.

If XML is returned it will be both human and computer readable. In theory a human being can understand the following: `<book-title>Great Expectations</book-title>`.

In the tag `<book-title>` the text between the chevrons demonstrates the nature of the text between the tags, in this case the fact that *Great Expectations* is a book title. However, XML often contains a confusing number of such tags, so that even though in theory human beings can read and understand them, in practice it becomes a daunting task.

Programming an XSLT Stylesheet that transforms the XML and outputs it in a more legible format usually solves this problem. This could be HTML (if it is to be

⁵⁴ See page 36 for an overview of online corpora.

displayed in a browser), a CSV file (for importing into a spreadsheet), an SVG file⁵⁵ (for graphs or barcharts) or a plain text file. The XSLT Stylesheet can be encoded to find each `<book-title>` tag in an XML document and print the tag value after the text '*Book Title*'. It can also be encoded to change the font to bold, give it a different colour or place it in a certain position on the screen as seen below:

Book Title: Great Expectations

By using a Web Service to return the result of a search it is also possible to incorporate this search result in a different Web application such as the DSS, meaning that scholars will not have to go to a different website every time they wish to search for interpretation parallels, they will instead be able to do it through the in-built search mechanism in the DSS.

```
<list type="words">
  <item n="suus">
    <ident type="lemma">suus</ident>
    <list type="tablets">
      <item n="eAla008">
        <ident type="tabletID">eAla008</ident>
      </item>
      <item n="iAph010201">
        <ident type="tabletID">iAph010201</ident>
      </item>
    </list>
  </item>
</list>
```

Fig. 2.4 The XML returned from Inscriptions of Aphrodisias when searching for the Latin word 'suus' using APPELLO. The result shows that there are two documents (eAla008 and iAph010201) in which the Latin word 'suus' is present.

To illustrate how this could work, I have used the Inscriptions of Aphrodisias Project, which is available as EpiDoc encoded XML originating from the two online publications; ala2004 (Roueché 2004) and InsAph2007 (Reynolds *et al.* 2007). Inscriptions of Aphrodisias, is a dataset containing almost two thousand mostly Greek texts from a single city in Asia Minor. By using the word search Web Service APPELLO with this

⁵⁵ See Piez 2005 for a description of how TEI encoded XML files can be transformed into SVG graphics.

dataset it is possible to search through the entire corpus for examples of the word *suus*. The result is retrieved as an XML document (fig. 2.4), which through the use of an XSLT style sheet can be displayed as HTML in the DSS or any other browser based application.

2.5.3 INTERNATIONAL COLLABORATION

The third issue arises when colleagues in different parts of the world attempt to work together on an interpretation in real time. Recently, technologies such as instant messaging (e.g. MSN) and Internet phone calls (e.g. Skype) have made this more feasible. Nevertheless, this still leaves the task of explaining complex reasoning to a colleague without being able to show what you mean in the text. The development of virtual research or viewing environments such as the BVREH project (see above) or the Virtual Vellum project (Ainsworth and Meredith 2009) will help to solve this issue by enabling users to view high quality imagery through a browser.

Another solution is a browser-based DSS application as presented here, which would enable two scholars to sit in different departments and work together on the same text with only the use of an Internet connection and a computer with a browser installed.

The DSS would use AJAX technology, meaning that when one user made a change to the reading it would instantly be updated for the other user to see. Even if the scholars did not wish to collaborate on an interpretation, the DSS would still make it possible to publish certain parts of a scholar's interpretation so that colleagues could comment on it and use it to advance their own interpretations.

2.5.4 PUBLISHING EDITIONS

The fourth issue is that of preparing interpreted documents for publishing. It is the tradition of ancient historians to publish these editions in a certain format, preferably using encoding symbols and brackets, such as the Leiden Conventions. These days this is often done in text editing software (e.g. MS Word) by manually inputting the encoding symbols

and brackets. However, if the text is to be encoded and re-encoded for different editions, this method of manual encoding can very easily lead to errors due to mistyping.

Encoding documents with an XML schema (e.g. EPIDOC or TEI) makes publishing much easier and leaves less room for mistakes once the encoding has been completed. Using XSLT stylesheets the XML can be transformed into different formats. It can be transformed into HTML for viewing in a browser, or into text for printed publishing. Furthermore it can be used to extract bits of data to be added to a database or spreadsheet.

However, the main problem with this method is that it can be very time-consuming and costly to encode with XML (Tupman 2010). Luckily, there are methods of automating or semi-automating this process. As a part of the Vindolanda Tablets encoding project a script was developed using pattern recognition (see 4.2.4), which enabled the input of the text from the volumes of published editions of the Vindolanda Tablets (Bowman & Thomas 1994, 2003). In effect it is a PHP script, run on the text version of the edition. When it comes across a predefined pattern it will add the appropriate XML tags to the XML file, and this XML file is then saved onto the server⁵⁶.

A DSS would enable us to skip this process all together. The result of the interpretation process is stored in the DSS, which can then output the finished result as an XML encoded edition.

2.5.5 IMAGE ENHANCEMENT

In the early days of papyrology the main means of reading ancient documents were either possession of the original or in the case of inscriptions making a "squeeze" (paper impression) version of it. The use of photography later enabled scholars to interpret a written document without ever having seen the original. Today we take the use of photography for granted, and digital technology allows us to collect an enormous amount

⁵⁶ See page 101 for more on how this script was developed.

of imagery from all angles, using different techniques such as multi-spectral imaging. This enables the reader to extract much more information from the imagery than they might have gained even when viewing the original with the naked eye.

Another advantage of digital imagery is that the pixels that make up each image can be individually subjected to mathematical analysis. Using these techniques there have been major advances made in the enhancing and manipulating of digital images of ancient documents.

However, this potentially huge quantity of imagery, which can all in one way or another aid the reading of ancient documents, is dependent on a successful integration into the process of interpretation. Chapter 5 describes some recommendations for a future DSS implementation that involves a layer-based system that can include different images as layers.

2.6 REASONING UNDER UNCERTAINTY

Scholars reading ancient documents, from the undergraduate student to the professorial expert, can be said to be simply, to a greater or lesser degree, making an educated guess. Many and varied factors can effect just how educated this guess is, for example experience is one of the major factors involved, as is access to resources. The former factor relies rather more on years of practice than the latter due to the present day accessibility of so many online datasets, but every scholar is in some way operating under some degree of uncertainty.

The question then is: How can this uncertainty be measured? To which I would answer: it cannot! In Humanities disciplines, where uncertainty is abundant, it is also largely un-measurable. Scholars are not able to state that they are 75% certain that the letter they see is *A*. They can only say that they believe it to be *A* because they have seen an *A* like this before, or because if the letter is *A* then the word could be *Flavia*, which would fit with the assumed context of the document. It is this word *because* that I will take

an interest in, as it represents the beginning of an argument. “I have seen a letter *A* looking like this before,” is a valid argument **for** the reasoning that the letter is *A*, whereas the argument “other examples of *A* in the text look different,” is an argument **against**.

Building on the conclusion of the MDTSuite development that forms a part of Austin’s research (Austin, 2008) (see 3.1.2) I will use argumentation to construct the basis of the development of the DSS prototype.

Every decision made, whether it is “This letter is *A*” or “This word is *Flavia*”, will have attached to it a set of arguments **for** and **against**.

The next step is to show how to use these arguments in the decision-making process. Can one assume that if there are two arguments **for** a particular reasoning and one **against** then this must be the most likely interpretation? I do not believe that it is possible to draw such a conclusion. The arguments cannot be tallied up because they are themselves subjective, non-exhaustive and non-quantifiable.

Indeed scholars might consider that the one argument **against** carries more weight than the two **for**, and this may influence their decision that the letter is not *A*. With the linking of arguments this method becomes even more appropriate.

This can be demonstrated by the following example: ‘An argument for *A* as the sixth letter is that the word could be *Flavia*, which would fit with the assumed context of the document’. This argument is linked to the word *Flavia* and the argument **for** the word *Flavia* is ‘it would fit into the context of the document’. But what if the context were to change? New evidence may come to light showing that the document is likely to be from a different period, where there is no previous knowledge of a person named *Flavia*. Because of the interlinked argumentation this would filter back to the argument **for** the letter *A*.

This would not automatically change the reading of the letter from *A*, but it would alert the scholars that one of the arguments upon which they based their decision has changed. It is then up to the scholars to decide whether or not they wish to adjust their

reading of the letter A.

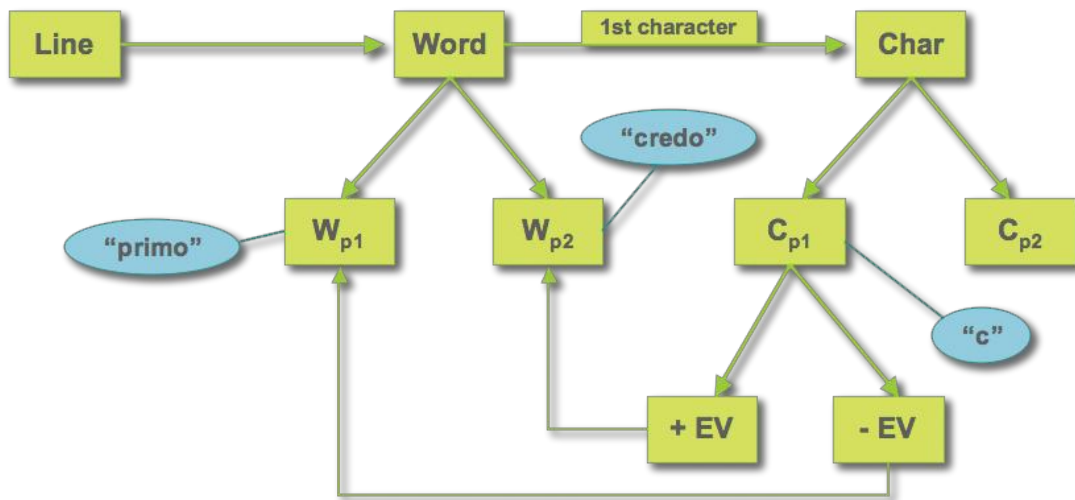


Fig. 2.5 Model explaining the use of interlinked arguments in the DSS

In the model above (fig. 2.5) the notion of inter-linked argumentation is featured using a word and the first character in this word. In the example of a five-letter word there may be different interpretations of what the first letter could be. Based on other letters in the word that are more certain (i.e. that the second letter is *R* and the fifth letter is *O*) the possible words have been narrowed down to two suggestions - the word is either *credo* or it is *primo*. If scholars believe the first character to be *C* then the interpretation of the word as *credo* would be an argument **for** this. The interpretation of the word as *primo*, on the other hand, would work as an argument **against**.

Similarly, the idea that the first character is *C* is an argument **for** *credo* and **against** *primo*. If new evidence was uncovered suggesting that the first character could in fact be *p* the DSS would notify the scholars that the arguments **for** *credo* has in fact changed and thus give them a chance to change the reading of the word to *primo*.

2.7 SUMMARY

This chapter has discussed the history of reading ancient documents and the various challenges faced by scholars. It has described the ‘network of interpretation’, derived from Terras’ ‘reading levels’, which is used to better understand what scholars actually do when reading ancient documents and which is also used to develop the database model for the DSS prototype.

It has debated the role of IT in the reading of ancient documents, which has been divided into five subjects that each represent areas in which I think IT can make a useful contribution. These five subjects are: image annotation tools, textual encoding tools and methods, corpora and online resources, image manipulation and analysis, and character recognition. However, the most important tool contributing to the interpretation process in this thesis is the DSS

Following on from this is a discussion of the basis upon which this research builds by specifying that the aim is not to develop a computer system that can read ancient documents, but rather to develop one that aids the tasks that scholars find difficult. These tasks are ‘remembering complex reasoning’, ‘searching huge datasets’, ‘international collaboration’, ‘publishing editions’ and ‘image enhancement’. The first task in particular, remembering complex reasoning, is the main objective of the development of the DSS, because while human beings are exceptionally good at complex reasoning they might not be as good at remembering this complex reasoning.

Uncertainty is another issue that must be factored in when reading ancient documents. Each and every interpretation undertaken as a part of a reading contains some degree of uncertainty. This is the inherent nature of interpretation. However, scholars cannot be expected to have the ability to express the degree of their uncertainty in a meaningful way and for this reason I have introduced the concept of argumentation to the DSS prototype, enabling scholars to list arguments **for** and **against** each interpretation,

which the prototype can then use to prompt them if their interpretation changes. Nevertheless, these arguments **for** and **against** should never be used to make a decision for the scholar, they should merely be seen as reminders.

In summary, this chapter highlighted the specifications for the design of the Decision Support System, namely that it can remember complex reasoning by storing decisions and arguments for and against decisions, during the reading of ancient documents.

3. DEVELOPING A DECISION SUPPORT SYSTEM PROTOTYPE

The process of interpreting ancient documents may be likened to solving a cryptic cross-word puzzle in a foreign and often dead language where you are given very few clues beforehand. Unlike most cryptic cross-word puzzles, these clues could very well come from the context of the document. If the document was found at a fort it could be related to military matters. If the format of the document begins with an address it is likely to be a letter. If it looks like a list it might be accounts. If some characters and words are legible then these can furthermore act as clues to other less legible parts of the document.

Chapter 2 introduced some of the tasks that form part of reading ancient documents such as remembering complex reading and searching through huge datasets. These are essentially activities that humans find difficult but which computers could be programmed to do rather effortlessly.

In the present chapter these concepts have all been included in the DSS prototype, which has been developed as a Web application that can run in any mainstream browser. It is mainly a PHP development that makes heavy use of XSLT to transform XML, the format in which most of the content is provided¹.

The integrated DBtoTEI Web Service², developed as a part of the DSS prototype, generates the basic structure and meaning of the document as a TEI compliant XML file using the content of the MySQL relational database tables. This dynamic XML file is transformed through an XSLT stylesheet for the purpose of creating the main view of the DSS (fig. 3.1).

¹ See 1.1 for the reasoning behind this choice of technologies.

² See 4.3 for an introduction to Web Services



Fig. 3.1 Screenshot of the main view of the DSS using Vindolanda tablet 670 as an example.

This main view is output as HTML and is thus viewable through a browser. Each word and character element in the main view is a link, which activates a pop-up box. The pop-up boxes are created with a combination of JavaScript and PHP using AJAX. Their purpose is to act as editor boxes where the scholar can edit the details displayed in the main view. By using AJAX the main view can be edited without the need to reload the webpage, giving the user a smoother interaction with the application. The information used and added via these pop-up editing boxes does not operate through the DBtoTEI Web Service but is instead directly selected from or inserted into the MySQL database. Furthermore, the `<Search parallels>` facility in the word edit box uses XML from different Web Services to provide a result.

In this chapter I will use two Vindolanda tablets as examples. Tablet 670 (fig. 3.3) is used to demonstrate how the DSS prototype was developed. Tablet 862 (fig. 3.4) is a newly interpreted tablet and is used to demonstrate how three scholars (S1, S2 and S3) can use the same prototype to aid their collaborative interpretation.

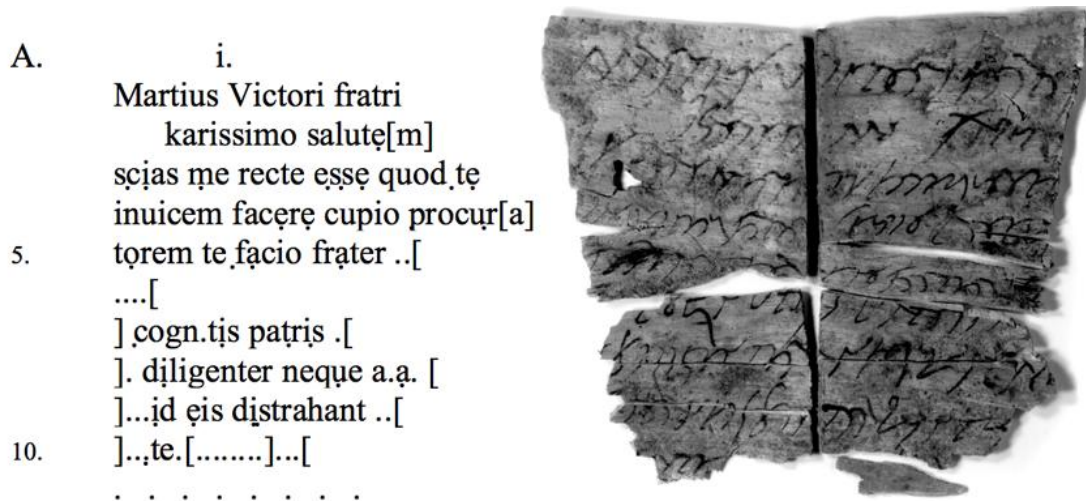


Fig. 3.2 Section A.i from Vindolanda tablet 670. Image used with permission of the Centre for the Study of Ancient Documents, University of Oxford and the British Museum.

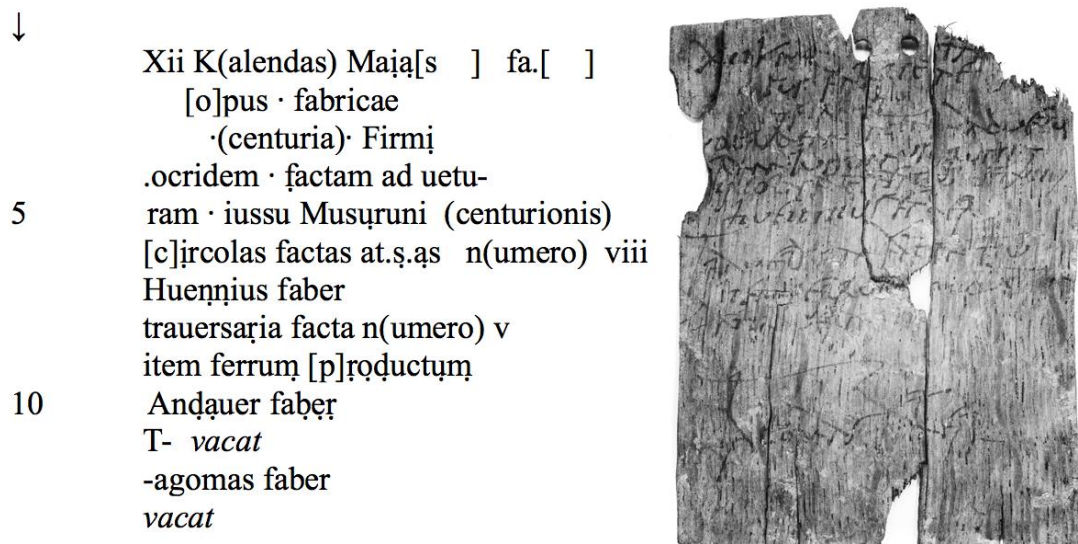


Fig. 3.3 Vindolanda tablet 862. Image used with permission of the Centre for the Study of Ancient Documents, University of Oxford and the British Museum.

The examples from tablet 670 are from the published edition of the document (Bowman and Thomas 2003), while the examples from tablet 862 are partially from the newly published edition (Bowman *et al.* 2010a)³ and partially extracted from unpublished notes made by the three scholars (Appendix B). The examples from tablet 670 can also be viewed as videos on the attached CD (Appendix C and Appendix H).

³ See Appendix A for the published versions of tablet 670 and tablet 862.

3.1 INTRODUCING DECISION SUPPORT SYSTEMS

A DSS is defined as a computer based system for the purpose of aiding decision-making (Sol *et al.* 1987). DSS have evolved since they were first introduced as managerial aids in the 1970s, and major breakthroughs in technology have allowed them to become increasingly analytical in their approach, to the extent that they are beginning to overlap with Expert Systems (Turban and Watkins 1986), which makes sense in terms of business and managerial needs.

If, based on a set of pre-determined parameters, a computer can calculate exactly the most appropriate insurance policy to sell, then this will save on manpower and give more consistent results than a human being making the same calculations. However, for someone reading ancient documents consistency is not necessarily a primary goal. Furthermore, the number of parameters used in the interpretation of ancient documents, most of which are not even possible to calculate, cannot be compared to those needed to compute an insurance policy. It is for this reason that this thesis is focusing solely on the decision support aspect and disregarding Expert Systems for this purpose.

The DSS prototype that this thesis is developing will focus on the storage and organisation of complex reasoning, by making use of the concept of argumentation. Each section, line, word and character element interpreted in the text will be stored in the relevant MySQL table together with each letter, argument and link as outlined in the database model discussed below.

The main HTML view of the DSS will be generated with a Web Service using the RESTful protocol to extract the known structure of a document from the tables (fig. 3.4). The XML output through this Web Service will be transformed with XSLT stylesheets into the HTML of the main DSS view.

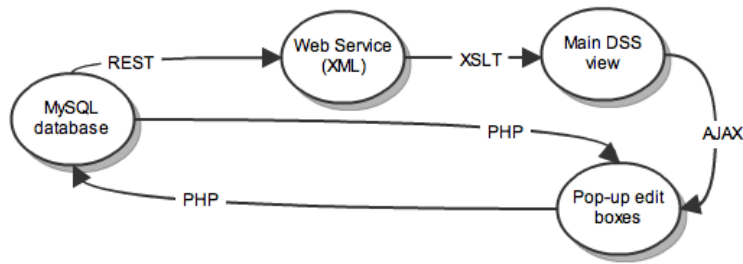


Fig. 3.4 Model of the flow of information in the DSS prototype.

Other parts of the DSS, such as the pop-up elements, are built on data extracted directly from the database tables. Similarly, any edits made through the DSS will be edited directly in the tables.

The system will use AJAX technology to ensure that any edits are available in the main view and pop-up boxes without the need to re-load the page.

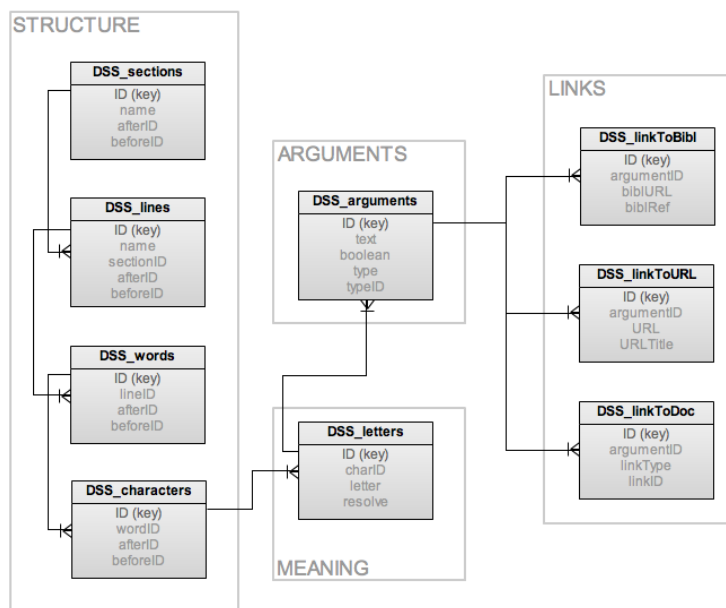


Fig. 3.5 Model of the MySQL relational database with the tables, fields and relationships between both displayed.

The model of the DSS (fig. 3.5) is in effect the relational database schema for the MySQL database that is used to run the DSS. The MySQL tables and content are presented in each of the following sections of this chapter. I will use the model here to explain the concepts that are being used in this approach to the reading of ancient documents. The

model builds upon the research described in Chapter 2 and illustrated through the model of interlinked arguments (fig. 2.5) and the model of the network of interpretations (fig. 2.2).

The model is divided into four sections. The first part deals with the structure of the DSS, and is related to step 0 in the network of interpretations. The structure of a document can consist of sections, lines, words and individual character elements, combined in a one-to-many relationship where one section can have many lines, one line can have many words and one word can have many characters. A character in this context is a structural element in the text where a character is believed to be and must not be confused with the letter which is the actual letter perceived to be in that character space (e.g. a, b or c). The letter is not part of the structural elements but is instead thought of as an element adding meaning to the document. Each character space can have many perceived letters, but only one letter in each character space can be resolved. If a letter is resolved it is an indication that the scholar has decided to adopt this particular reading of the letter. Each character space can have an infinite amount of un-resolved letters, whether or not it has any resolved letters. In the network of interpretations this step towards meaning relates to the jump from step 0 towards the other steps (i.e. step 1, 2 and 3).

The next part of the model is the arguments that again have a one-to-many relationship with letters. Each letter can be linked to an infinite number of arguments, and each argument must be either **for** or **against** the interpretation of the letter. Each argument can consist of either a text or it can be linked to either a bibliographical reference, to a URL or to another place in the same document. In the latter case the argument can be linked to another letter or a word.

All the tables in the model are linked to the previous table in a one-to-many relationship. Furthermore the relational database model reveals that the DSS prototype will only cover the network of interpretation up until step 1. It will not move on to demonstrate how the meaning of a word or the meaning of a phrase can be incorporated into a DSS.

3.1.1 EVOLUTION OF DECISION SUPPORT SYSTEMS

In order to understand the evolution of DSS it is important to first understand that it began as a managerial support tool, but has since evolved and is now used to support any field in which decisions are being made. In his 1978 book *Decision Support Systems: An Organisational Perspective*, Peter Keen credits the Carnegie Institute of Technology for their theoretical research on organisational decision-making in the late 1950s and early 1960s. The technological work on interactive computing systems was done primarily at the Massachusetts Institute of Technology during the 1960s, and in the early 1970s (Keen and Morton 1978). Michael S. Scott Morton articulated the field as 'Management Information Systems' (Gorry and Morton 1971: 55). However, DSS as an actual field of research was not acknowledged until the mid 1970s (Keen and Morton 1978).

The definition of DSS has changed over time; in the 1970s it was defined as 'a computer based system to aid decision making' (Sol *et al.* 1987: 1), and in the late 1970s Ralph Sprague and Eric Carlson defined it as 'interactive computer based systems, which help decision makers utilize data and models to solve ill-structured problems' (Sprague and Carlson 1982: 4). In the 1980s a new definition had emerged; DSS had become 'systems using suitable and available technology to improve effectiveness of managerial and professional activities' (Sol *et al.* 1987: 1).

In 1989 Sprague revised his former definition of DSS as '*interactive* computer based systems, which *help* decision makers utilize *data* and *models* to solve *unstructured* problems' (Sprague and Watson 1989: 10), adding the italicised words as a way to make keywords in the concept of DSS stand out.

Sprague's definition is a more limited definition of a DSS, against the more extensive definition of 'any system that makes some contribution to decision making' (Sprague and Watson 1989: 10). In the 1990s Vicki Sauter defined DSS as 'a computer based system that supports choice by assisting the decision maker in organising

information and modelling outcomes' (Sauter 1997: 13). In this study she focuses on models being 'created to help decision makers understand the ramifications of selecting an option' (Sauter 1997: 13).

Even though the definition of Decision Support Systems has changed and will probably continue to change in the future, there are certain general conclusions that can be drawn from these earlier definitions. Firstly, it can be generally agreed that DSS are computer-based systems. Secondly, that they are, as the name implies, aimed at supporting the process of decision-making (Klein and Methlie 1995). I also agree with Sprague that a DSS requires a certain element of interactivity. Additionally, both Sprague and Sauter focus on the use of both data/information and models to aid the decision making process. Finally, a recent evolution step in DSS development is towards Web-based/browser-based applications (Eom *et al.* 1998).

When discussing DSS, the subject of Expert Systems invariably arises too. Up until 1980 Expert Systems were mostly researched in British, American and Japanese universities. However, the 1980s saw a big expansion in commercial competition in Expert System application (Harmond and King 1985: 1). Expert Systems are systems programmed to make automated decisions on their own without human interference⁴. Perhaps in the 80s a project such as this one might have aimed to develop an Expert System that could automatically read ancient documents. But at the present time research is more geared towards the development of aiding systems, such as DSS⁵, that allow humans to do the interpretation while computers do the other tasks that humans find difficult. I believe that this represents a better and more realistic division of labour.

Sauter developed a scale to illustrate how DSS can be viewed in the context of

⁴ For an overview of Expert Systems, their definition, their issues and how they evolved see: Coats 1988; Harmons and King 1985, Hart 1986, McGraw and Harbison-Briggs 1989 and Waterman 1986.

⁵ See Turban and Watkins 1986 for a discussion about the differences between Expert Systems and Decision Support Systems.

computer systems ranging from Content Management Systems (CMS) to Expert Systems (ES). The scale went from **A** to **B**, in which **A** represents systems that are repetitive and which use linear logic to produce regular reports (these systems, such as Manage Information Systems (MIS) and Transaction Processing Systems (TPS), do not offer any kind of decision support), and **B** represents systems that are specialised heuristic and do not produce regular reports (these systems, often known as Expert Systems (ES), are programmed to make decisions themselves). Sauter places DSS in the middle of the scale between **A** and **B**, as, on one hand, it is neither repetitive nor does it produce regular reports and on the other hand it is also not programmed to make decisions (Sauter 1997). Sauter explains that, instead, a DSS ‘helps decision makers identify information useful in poorly structured underspecified problems’ (Sauter 1997: 13).

The interpretation of ancient and often damaged documents represents a poorly structured and underspecified problem; poorly structured because the scholar will most likely not be able to begin from the top of the document and work down in a regular manner, and underspecified because the scholar rarely has all possible parameters at hand. According to Sauter's stated purpose of DSS, such a system would seem to be a prime candidate for aiding the interpretation of ancient documents.

3.1.2 MDTsuite AS A DECISION SUPPORT SYSTEM

The idea of considering the possibility of DSS as a tool for aiding the reading of ancient documents initially occurred during the collaboration between the eSAD team at the Centre for the Study of Ancient Documents (CSAD) and the Department of Engineering at the University of Oxford, where Mark Austin was developing the MDTsuite (Austin 2008).

The idea behind MDTsuite was the creation of a system that would support the decision making process of a Multi-Disciplinary Team (MDT) working with colorectal

cancer patients. Austin worked on this concept, together with a group of doctors and engineers, as a part of his D.Phil, during which he attended several MDT meetings as an observer to get an idea of how the DSS could work in the context of these particular decisions.

Groups of experts (radiologists, surgeons and cancer specialists) meet once a month for rather short (1 hour) MDT meetings. Sometimes they need to discuss as many as 40 patient cases within this time frame, therefore needing a system that can also function as a Content Management System (CMS) in which they can store images such as Computed Tomography (CT) scans and patient information. This information can then be integrated into the DSS to enable them to use it as evidence for a decision, based on a knowledge base of rules (Austin *et al.* 2008).

Austin evaluated several methods for handling the decisions in MDTSuite. Traditionally a statistical approach such as Bayesian theory might have been favoured, but Austin argues against this on the grounds that it would not be realistic to expect experts to formulate their knowledge in this way and that it would stifle discussion.

Furthermore, it is not realistic to expect an expert to be able to format their expert knowledge into a numerical network of this type. In addition to this, we must remind ourselves of the requirements of MDTSuite and that we intend to support the decision making process through recommendations which encourage discussion. The numerical nature of the output of a Bayesian classifier is not well suited to this task (Austin 2008: 30).

Austin realised that real life medical decisions were too important and complex to be reduced to a numerical output. This type of ‘decision making in everyday life is typically much more complex than under laboratory conditions’ (Eysenk and Keane 2010: 525).

Fuzzy Logic was also rejected by Austin:

Also, it is the gulf between the clinicians’ perceptions of what these fuzzy values represent, and how they are actually manipulated mathematically, which might prove the major weakness of fuzzy systems for MDTSuite. We therefore also rejected fuzzy logic as a tool for decision support in MDTSuite. (Austin 2008: 34).

Austin finally turns to the concept of argumentation, which can be defined as ‘the

process of constructing arguments about propositions, and the assignment of statements of confidence to those propositions based on the nature and relative strength of their supporting arguments’ (Fox *et al.* 1996: 428). Austin used this method because he found that ‘by identifying the arguments for and against a treatment option explicitly, argumentation would also help to integrate diverse information, and ensure that all the relevant information about a patient being discussed is shared amongst the team’ (Austin 2008: 36).

Initially, Austin developed a version of MDTSuite that interacted with a relational database, but halfway through the project he redeveloped the system so that it is now built on a Resource Description Framework (RDF)⁶ with an ontology running the rule-base. The rule-base is made up of standard recommendations that can continuously be edited to suit the guidelines of an individual team. The MDTSuite was tested by the local MDT at the John Radcliffe hospital in Oxford (Austin 2008).

3.2 INPUT

The DSS is highly reliant on the input of data, whether it is the manual input of arguments or the automatic input of suggestions from a search engine.



ID	name	afterID	beforeID
1	A.i	0	2
2	B.ii	1	3
3	A. Address	2	0

Table 3.1 DSS_sections table from the DSS_670 database

The input used to demonstrate the examples in this prototype is one of manually adding the finished information directly into the MySQL database. Vindolanda Tablet 670

⁶ RDF (Resource Description Framework) is a W3C specification used for making statements about resources as subject-predicate-object triples <http://www.w3.org/TR/rdf-primer/>

line 5 in section A.i is used as an example of this input method and phpMyAdmin is used to manage the MySQL tables.

In this example the main structural elements of the document will be input in table `DSS_sections` (table 3.1). Section A.i was the first section to be added and therefore has the `ID = 1`. The relationship between the sections is not dependent on the `ID` but is instead determined by the `afterID` and `beforeID` fields. In this case section A.i is the first section (i.e. `afterID = 0`) and comes before section B.ii (i.e. `beforeID = 2`). 0 is used to indicate that the section does not come after or before any other section, in other words that it is the first or the last section.

In the prototype it is furthermore possible to edit the letter interpretations and the arguments for and against each in the front-end interface. When scholars add a letter to a character element where there was no letter to begin with then the following process outlines how this has been addressed in the DSS system. A click on the character element brings up the letter editing pop-up box (fig. 3.6). Here they can first click the `<+Add a letter>` button and then fill in the letter or letters that they want to associate with this particular character element. Through this interface they can also edit the letters, add more letters, delete letters, resolve letters and see the arguments for each letter.

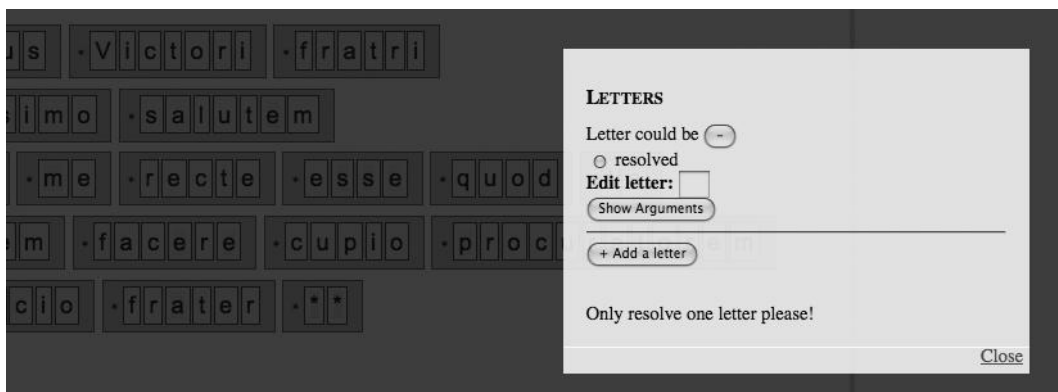


Fig. 3.6 Screenshot of letter editing pop-up box

3.3 STRUCTURE

Most text documents will have a similar structure (i.e. they will have lines, words and characters), but seldom will scholars be able to identify this structure right from the outset. To begin with, as discussed in Chapter 2, scholars interpret text by moving between the steps in the network of interpretation. This is why the input of the structure should ideally be as flexible as possible to allow scholars to move back and forth between this and subsequent levels. The prototype necessitates a much more rigid method of input that would not suit real-world scholars.

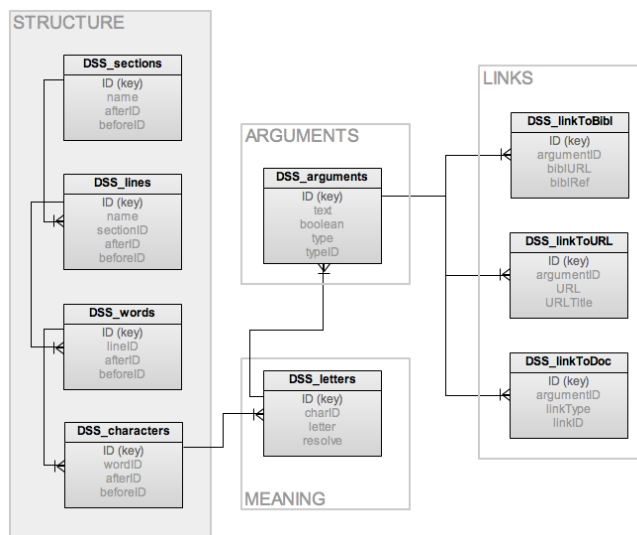


Fig. 3.7: Model of the MySQL relational database with the tables, fields and relationships between both displayed and the structural tables highlighted.

Referring back to the model (fig. 3.7), the structure is thought of as the sections, lines, words and character elements of the document, where they are and how they relate to each other. Understanding the structure of the document relates to step 0 in the network of interpretation (fig. 2.2), which is the step towards identifying the structural elements of such a document.

The DSS is being stored within a MySQL database. Four tables cover the structure: DSS_sections, DSS_lines, DSS_words and DSS_characters. The DSS_sections table includes the fields ID, name, afterID and beforeID. The ID field is the unique identifier for this table

and is an integer. The reason for the `name` field is that the section may have a different identifier from the `ID` (e.g. A, ii, one). Furthermore the `ID` has no meaning in terms of relation with other sections. Thus `ID = 1` is not necessarily the first section in the document and is not necessarily followed by `ID = 2`. The relationships between sections are instead determined by the fields `afterID` and `beforeID`. Thus the first section in the document has `afterID = 0` and the `beforeID` is the `ID` of the section after it.

The tables `DSS_lines`, `DSS_words` and `DSS_characters` are very similar except that they include the fields `sectionID`, `lineID` and `wordID` respectively. The field `sectionID` in the `DSS_lines` table contains the `ID` of the section that the line is in, `lineID` in the `DSS_words` table is the `ID` of the line that the word is in and finally `wordID` in the `DSS_characters` table is the `ID` of the word that the character is in.

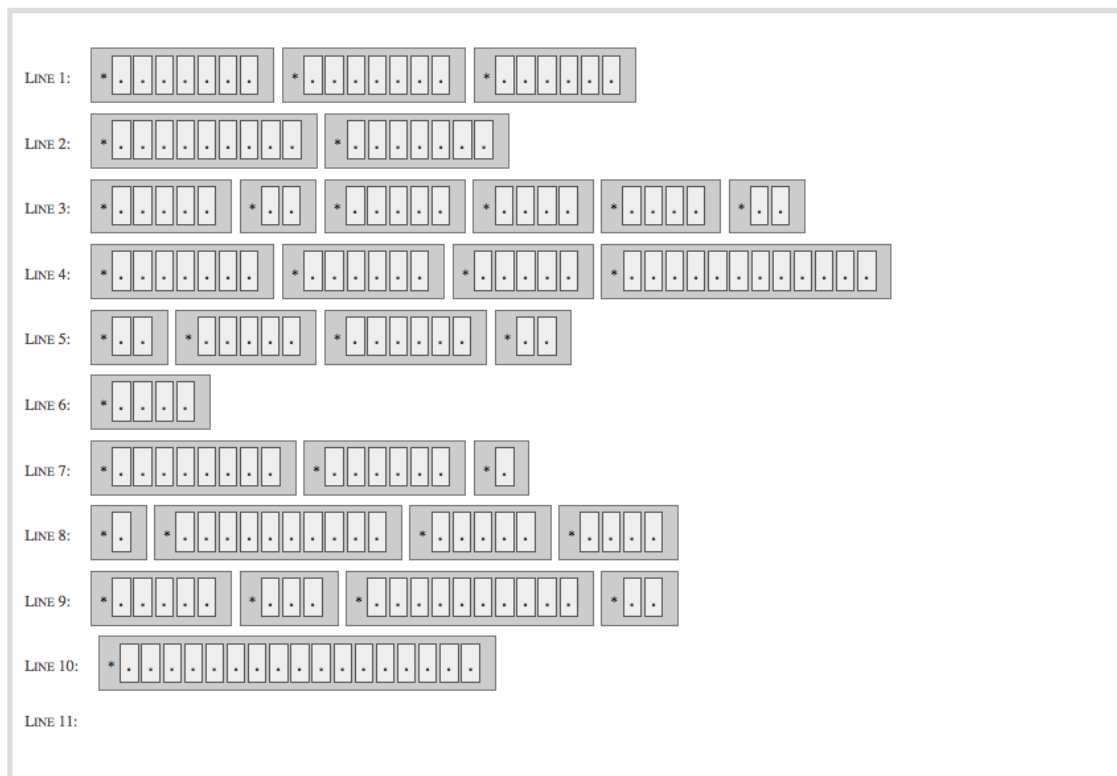


Fig. 3.8 Screenshot of section A.i of Vindolanda tablet 670 in the DSS with structure (i.e. sections, lines, words and character spaces) of the document only.

Tablet 670 (fig. 3.8) is a document where the current interpretation of the structure consists of three sections one of which is section *A.i*. The details of this section were previously input into the database in table `DSS_sections`. The interpretation suggests that section *A.i* contains 11 lines. These 11 lines are stored in table `DSS_lines`. As can be seen in table 3.2 each of these lines' ID corresponds to their name. This is because they have been input in the order they occur. Even though they do not need to be input in this order for the system to work (because of the `beforeID` and `afterID` fields) it is still probable that they will be as the amount of lines in a document is likely to be one of the first interpretations of a document that a scholar will make. Thus line 8 is after line 7 (`afterID` = 7) and before line 9 (`beforeID` = 9).

ID	name	sectionID	afterID	beforeID
1	1	1	0	2
2	2	1	1	3
3	4	1	2	4
4	4	1	3	5
5	5	1	4	6
6	6	1	5	7
7	7	1	6	8
8	8	1	7	9
9	9	1	8	10
10	10	1	9	11
11	11	1	10	0



Table 3.2 *DSS_lines* table from the *DSS_670* database

Line 8 in the table `DSS_words` (table 3.3) contains four words, among them word 25, which is the second word in the line between word 24 and word 26.



ID	lineID	afterID	beforeID
24	8	0	25
25	8	24	26
26	8	25	27
27	8	26	0

Table 3.3 Part of DSS_words table from the DSS_670 database

In the table `DSS_characters` (table 3.4) word 25 (i.e. 'diligenter') has 10 characters. This is an example of how the ID of a character does not have to be in sequence with the characters placement in the word. Character 39 is the fourth character in this word and it is placed between character 129 and character 130.



ID	wordID	afterID	beforeID
39	25	129	130
127	25	0	128
128	25	127	129
129	25	128	39
130	25	39	131
131	25	130	132
132	25	131	133
133	25	132	134
134	25	133	135
135	25	134	0

Table 3.4 Part of DSS_characters table from the DSS_670 database

```

- <w n="25">
  + <c n="127"></c>
  + <c n="128"></c>
  + <c n="129"></c>
  + <c n="39"></c>
  + <c n="130"></c>
  + <c n="131"></c>
  + <c n="132"></c>
  + <c n="133"></c>
  + <c n="134"></c>
  + <c n="135"></c>
</w>

```



Fig. 3.9 XML returned when running the DBtoTEI Web Service for word 25

Whilst the tablets each contain the data that provides the structure of the document, it is in fact through an inbuilt DBtoTEI Web Service that this structure is fed to the main view (fig. 3.1). DBtoTEI is a Web Service that connects to the database and retrieves the structure in the correct order as a TEI encoded XML document (fig. 3.9). In the example above the word 25 is output as a `<w>` tag with the attribute `n="25"` containing ten `<c>` tags, one for each character element in the word. Each of these `<c>` tags is before and after the correct character from the table. As an example of this the fourth character from above with the `ID = 39` is now fourth in the list. The DBtoTEI Web Service is called every time the main view of the prototype is reloaded and every time it in turn calls up the database and receives the latest status of the structural elements. These are then converted into XML tags and returned to the prototype that then uses an XSLT stylesheet to display these tags as the boxes in the main view (fig. 3.1).

3.4 MEANING

Understanding the meaning of a document is the ultimate goal of the interpretation process. In itself it suggests that scholars have read and understood at least part of the document. There are different levels of meaning in an interpretation. There is the meaning of a letter, a word, a phrase or the document as a whole. This thesis differentiates between

a character element and the actual character itself (i.e. a, b or c), for which I use the word ‘letter’.

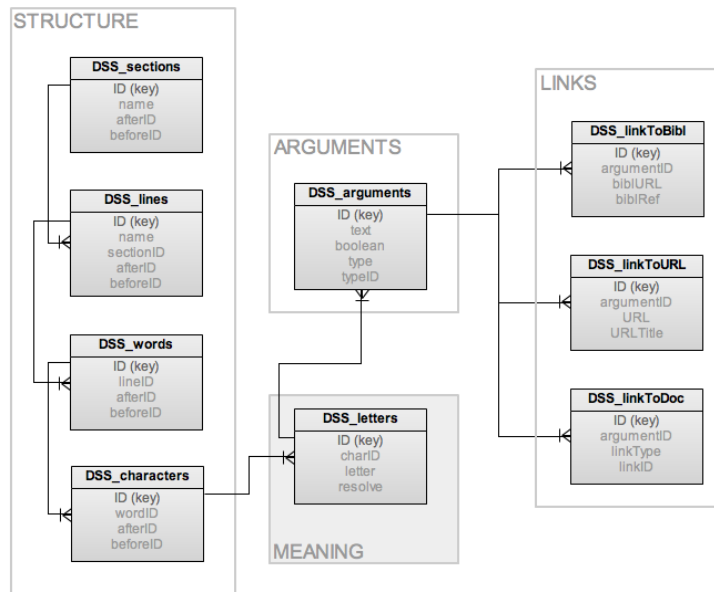


Fig. 3.10 Model of the MySQL relational database with the tables, fields and relationships between both displayed and the meaning table highlighted.

A fully developed DSS would enable scholars to enter and edit the meaning for all the different levels of a document. However, for the DSS prototype I have chosen to demonstrate the letter level of interpretation as an example of how meaning can be incorporated in the DSS (fig. 3.10). Thus the table that contains this meaning is `DSS_letters`, which includes the fields `ID`, `charID` (the `ID` of the character space to which the letter is connected), `letter` (the actual letter: a, b or c), and finally `resolve` (a boolean value that scholars can use to indicate whether they favour one letter over others, e.g. if `resolve = 0` the letter is not favoured but if `resolve = 1` the letter is favoured). Only one letter can be resolved in each character element.

As with the structure of the document the meaning is also displayed in the main view via the integrated DBtoTEI Web Service.

Tablet 670 is again used as an example (See Example 1, Appendix C). The character space with the `ID=115` it is connected to three letters and none of these three

letters have been resolved (table 3.5).

ID	charID	letter	resolve
223	115	a	0
224	115	u	0
225	115	i	0

Table 3.5 Part of DSS_letters table from the DSS_670 database

Note that character spaces may contain more than one letter thus resulting in a one-to-many relationship between `DSS_characters` and `DSS_letters`. However, only one letter per character can be resolved (i.e. `resolve = 1`). On the other hand it is perfectly acceptable that none of the letters connected to a character space are resolved (i.e. `resolve = 0`). If for example, scholars have two or more letters that could fit the character space, but do not think that there is sufficient argument available for either of the letters, all letters can be left unresolved. This is the case with the example from above. The fifth character element in line 7 (fig. 3.13) of tablet 670 (i.e. character `ID=115`) is left unresolved, but there are three potential vowels that the scholars thought could possibly fill the space (i.e. *i*, *a*, and *u*). The commentary from tablet 670, section A, line 7 states:

There is little doubt that the first four letters which survive in this line are *cogn* and that the next but one is *t*; thus *çognat-* or *çognit-* seems inevitable. The reading, however, seems to be *çognut*, for which we have no suggestion. If we are to read *çognat* or *çognit*, we have to assume that the first stroke of the apparent *u* is really a descender from a letter in the (lost) line above (Bowman and Thomas 2003: 128).

Therefore, in the letter editing pop-up none of the three characters are marked as resolved and the character element in the main view is filled with a question mark and coloured red (fig. 3.13).



Fig. 3.11: Screenshot of line 7, section A.i, Vindolanda tablet 670 in the DSS prototype.

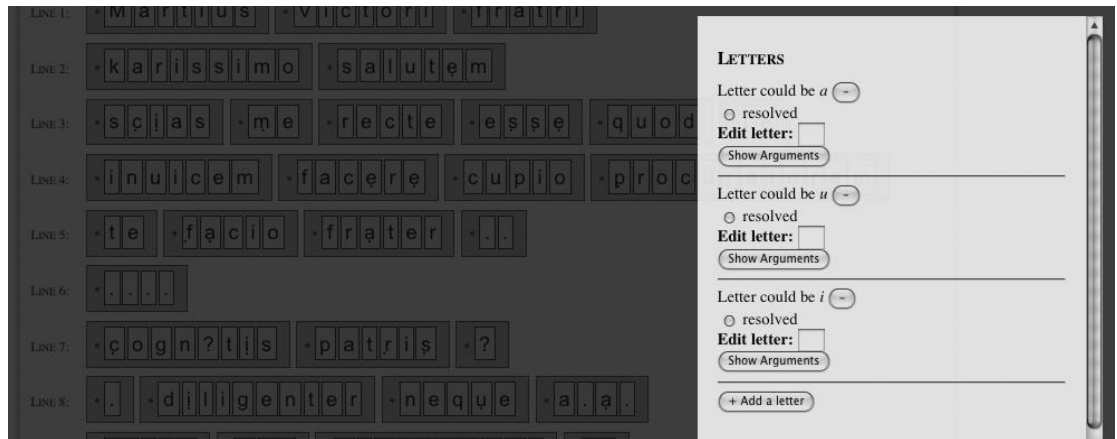


Fig. 3.12: Screenshot of the pop-up box with the letters connected to the fifth character space in the word *cognit*. Of the three letters (i.e. *i*, *a*, and *u*) neither has been resolved.

While the display of these three characters in the letter editing pop-up box (fig. 3.14) is created through a direct request to the database for all letters in the table `DSS_letters` that have `charID = 115` (table 3.5) the main view rendering of a red character box with a question mark inside (fig. 3.13) is created by running the DBtoTEI Web Service result (fig. 3.15) through the XSLT stylesheet. In the TEI encoded XML result returned by the DBtoTEI Web Service a letter is encoded with the `<g>` tag. Each `<g>` tag has an attribute `ana`, which can either be `ana = "#resolved"` if the letter in question is resolved and `ana = "#unresolved"` if it is not. In this example none of the three characters are resolved and the stylesheet will output a case such as this as a red highlighted question mark.

```
<c n="115">
  <g ana="#unresolved">a</g>
  <g ana="#unresolved">u</g>
  <g ana="#unresolved">i</g>
</c>
```

Fig. 3.13 DBtoTEI Web Service result for character 115

In the main view different colours are used to display the state of resolution of the letters (fig. 3.16). The colour blue is used if no letters have been suggested for a character

space. If there is only one letter suggested for a character space and it has been resolved, this letter is displayed in white. If only one letter is taken into consideration, but it has been left unresolved, the letter is displayed with a yellow background. In the same way the colour green is used if more than one letter is considered but one of these letters has been resolved. If on the other hand there is more than one letter and none have been resolved, as is the case in the example above, (fig. 3.13) the colour red is used and a question mark is displayed instead of a letter.

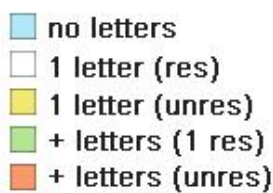


Fig. 3.14 Legend of the colours used to illustrate the state of resolution for letters in the DSS prototype

The interpretation of a character element can be added as a letter through the DSS main view by clicking on the character space in question. This will activate a letter editing pop-up box (fig. 3.14), which will list any letters already connected to a certain character space. This also gives the option of adding other letters or deleting current letters. In addition it is possible to edit, resolve, and un-resolve current letters. However, the system will only allow one resolved letter per character element and will not react if an attempt is made to resolve a second letter.

Up until now the focus has been on how this part of the DSS prototype has been built and how the tables in the database have been populated. Now tablet 862 will be used as an example to show how scholars can use the interface.

The first example uses the third word in the first line, which in the final interpretation reads:

Maiā[s

In S1's notes (Appendix B), he discusses whether the first letter is M or N. When using the DSS prototype for this document, he clicks on the first character of the word to bring up the letter editing pop-up box. Here he adds two possible letters, one in which he enters N and one in which he enters M.

He then realizes that M provides the best reading so he chooses M as the resolved letter. At this point the system enables him to choose between either deleting N as a possible letter or keeping N as an unresolved letter with the related arguments in order to remind himself or his colleagues why he is not considering N after all.

3.5 ARGUMENTS

Chapter 2 discusses how to deal with reasoning under uncertainty. Several quantifying/probability methods were discarded in favour of argumentation. The idea behind this is that each decision can have arguments **for** and **against** it. A common misunderstanding is that these arguments will be quantifiable. For example if the scholars had four arguments **for** a decision and one **against**, the misconception is that the DSS would try to persuade the scholars to take this decision based on the number of arguments alone. However, the DSS should not treat the arguments in such a quantitative manner.

Instead, an argument may be linked to another character, word or line in a document. The DSS will then warn scholars when any changes are made to this linked item and thus give them a chance to change their mind based on this new information.

In practice the arguments are stored in the table *DSS_arguments* and the related link tables: *DSS_linkToBibl*, *DSS_linkToURL*, and *DSS_linkToDoc* (fig. 3.17). The *DSS_arguments* table has a many-to-one relationship with the *DSS_letters* table, meaning that one letter can have many arguments connected to it. The *DSS_arguments* table includes the fields: *ID*, *text* (a free-text field where scholars can write the argument text), *boolean* (whether the

argument is **for** (1) or **against** (0) the letter in question), *type* and *typeID*.

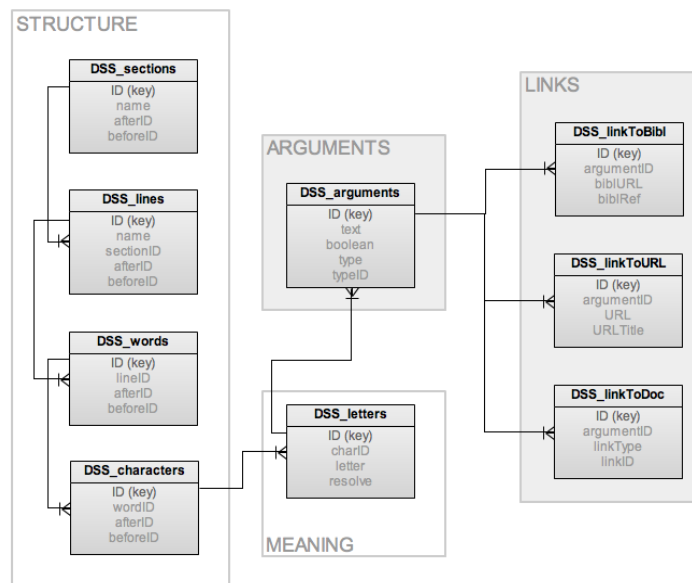


Fig. 3.15: Model of the MySQL relational database with the tables, fields and relationships between both displayed and the argument and link tables highlighted.

ID	text	boolean	type	typeID
24	The p of procur[a]torem lacks the loop which is fo...	0	letter	150

Table. 3.6 Part of *DSS_arguments* table from the *DSS_670* database for argument 24

In this example using tablet 670 (See Example 2, Appendix C), the argument (*ID* = 24) is linked to the letter 150 (i.e. *type* = *letter* and *typeID* = 150) (table 3.6). Letter 150 is *p* and is also the first and only letter currently connected to the first character of the word *procur[a]torem* in line 4-5 (fig. 3.18). This letter *p* has not been resolved and the reason for this is stated in the argument text: “*The p of procur[a]torem lacks the loop which is found elsewhere in p in this hand*”.

The link tables all have an *ID* field unique for each table and the field *argumentID* linking them to a specific argument. The *DSS_linkToBibl* table can link the argument to any bibliographic reference and includes a field to specify the reference (*biblRef*) and a field to add a URL for the reference if one is available (*biblURL*). *DSS_linkToURL* includes a

field for the URL that is being linked to (*URL*) and a field for a title for this URL (*URLTitle*). Finally, *DSS_linkToDoc* includes the field *linkType* (i.e. word or letter) and the field *linkID* (i.e. the ID of the word or letter).

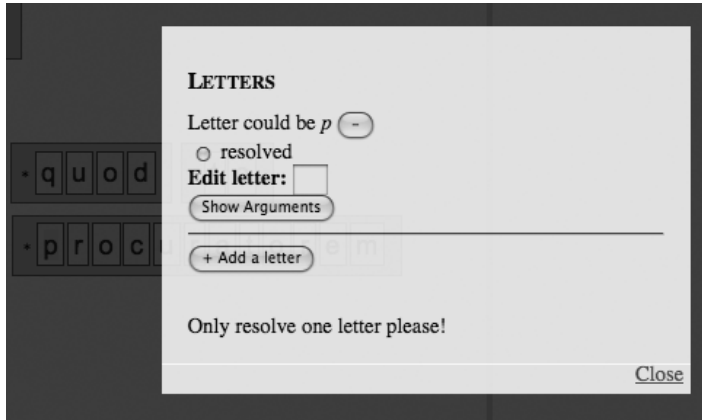


Fig. 3.16 Screenshot of the letter pop-up box for the letter 'p' in the DSS prototype

In the table *DSS_linkToDoc* it is possible to link from any argument to other parts of the document. Using the previous example, argument 24 does indeed have a link (*ID* = 13) to somewhere else in the document. It is linked to the letter 119 (i.e. *linkType* = letter and *linkID* = 119) (table 3.7).

ID	argumentID	linkType	linkID
13	24	letter	119
14	27	word	22
15	27	word	15

Table 3.7 Part of *DSS_linkToDoc* table from the *DSS_670* database

The letter that this argument is linked to is the *p* in *patris* in line 7 (fig. 3.19). This *p* is an example of another *p* in this hand that has the certain loop which the *p* in *procur[a]torem* does not have. Therefore, this other *p* is linked to this argument **against** the *p* in *procur[a]torem*. Now if the scholars were to change their minds about the *p* in *patris* the DSS could notify the scholars that this might have an influence on their lack of decided preference for the *p* in *procur[a]torem*.

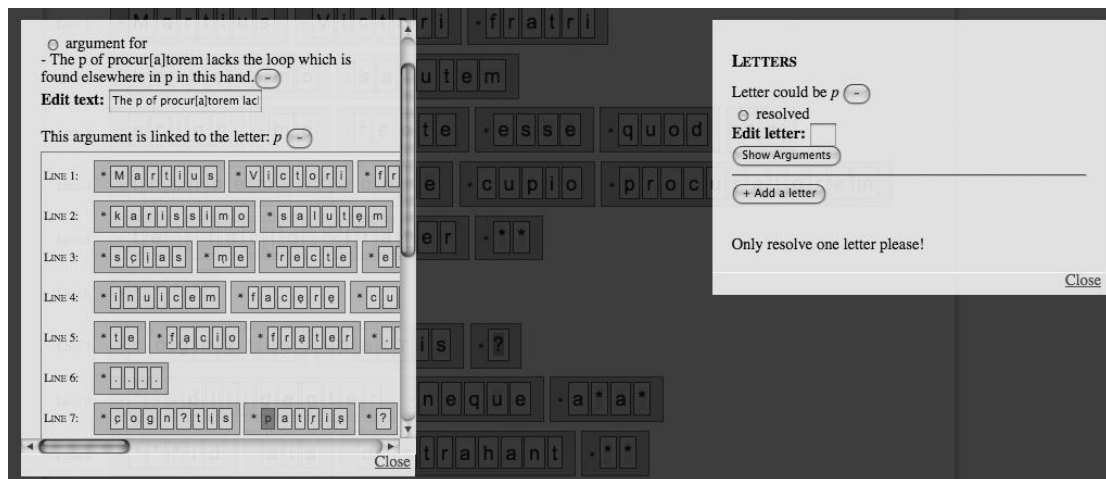


Fig. 3.17 Screenshot of the letter editing and the argument editing pop-up in the DSS prototype

Another example of how these arguments work is taken from line 7 in tablet 670 (fig. 3.13). In the letter editing pop-up box there are three possible letters (i.e. *a*, *i*, and *u*) connected to the fifth letter in the word *cognat.is*. By clicking on each of the <Show Arguments> buttons we activate an argument editing pop-up box for the particular letter. This is a list of the arguments **for** (+) and **against** (-) each of the three letters:

Arguments for and against the letter a:

(+) A reference to *cognati patris* would make good sense with *procurator* in a private legal capacity

1. Document link to *patris* in line 7
2. Document link to *procur[a]-torem* in line 4-5
3. Bibliographical link to *TLL X.2 1573ff, CPL 221 and 225*

(+) *cognat-* seems inevitable

Arguments for and against the letter u:

(+) The reading seems to be *cognut*

(-) First stroke of the apparent *u* is really a descender from a letter in the (lost) line above.

(-) We have no words that fit *cognut*

Arguments for and against the letter i:

(+) *cognit*- seems inevitable

For the untrained eye it may seem clear, that with the arguments for the letter *a* in comparison to the lack of arguments for the letters *i* and *u*, the letter must be *a* and the word must be *cognatis*. Nevertheless, the scholars interpreting this tablet did not think that there was sufficient argumentation allowing them to draw this particular conclusion and therefore they have left all three letters unresolved.

The first argument for *a* is linked to the outside resources *TLL* (Thesaurus linguae latinae) and *CPL* (Corpus papyrorum latinarum). The table `DSS_linkToBibl` (table 3.8) illustrates how these three bibliographic links are associated to argument 27.

ID	argumentID	bibiURL	bibiRef
1	27	http://www.thesaurus.badw.de/	TLL X.2 1573ff
2	27		CPL 221
3	27		CPL 225

Table 3.8 Part of `DSS_linkToBibl` table from the `DSS_670` database

This is meant as a reminder to the scholars of what their argument is based upon and is useful for a future reader to assess the strength of the argument. This is in essence a part of the note process that scholars go through when publishing an edition in order to comment on their interpretations line by line and establish parallels between the text they are reading and other texts.

The next example illustrates how the scholars reading tablet 862 can make use of argumentation in the DSS prototype using again word 3 in line 1.

For the first letter S1 debates whether this reads *N* or *M*. An earlier example illustrated how S1 inputs *N* and *M* as two possible letters in the first character space (Appendix B). S1 then adds arguments for and against each letter both for his own records

to remind him why he made the decision that the letter was *M*, but also to make his reasoning process clear to the other two scholars.

In the letter editing pop-up box he clicks on the `<add arguments>` button for the letter *M* and adds the following arguments **for** and **against** *M*:

- 1) (-) The fourth stroke of the *M* has been lost
- 2) (+) The preceding *XII K* suggests that the word must be a month and the second letter looks mostly like an *A*, meaning March or May (in latin).

For and **against** *N* are the following arguments:

- a) (+) The letter resembles *N* in some hands
- b) (-) This scribe writes *N* with a horizontal stroke, and then down, not up
- c) (-) We would expect the word to be a month based on the preceding *XII K*. If the first letter is *N* we would expect it to be followed by *OV*, but this is not the case.

The next step is to link these arguments to supporting information such as references in the same document, links to external resources and bibliographical links.

In this example S1 can link argument **b)** to another instance of *N* in the same document. This would mean that if the *N* that this argument was linked to changed, the DSS could highlight this and give S1 a chance to change his reading if he wished to.

In the same way arguments **2)** and **c)** are based on the idea that the word is a part of a date. An assumption that relies on the preceding letters, *XII K*. Therefore these two arguments can be linked to the words *XII* and *K*.

The scholars can furthermore add a link from arguments **2)** and **c)** to an external resource about the Roman Calendars to back up these two arguments.

Finally, the scholars can add a bibliographic link from argument **a)** to another set of documents, which showed examples of the letter *N* written in a similar way to the one in

this instance.

3.6 SUMMARY

This chapter demonstrates the development of the DSS prototype. Chapter 2 illustrated through Terras' reading levels, how interpretation of a document is carried out on several levels from the physical attributes of the document, through the meaning of a single letter, to the meaning of the document as a whole. Terras used these levels to examine three scholars' reading process and found that the scholars operated particularly on the levels concerning the features of characters and identification of characters and words. Therefore, these three levels were used as a framework for the network of interpretation around which this DSS prototype is built.

First this chapter discussed how the DSS prototype will work using the 'flow of information' model (fig. 3.4) and database model (fig. 3.5). The DSS will need to adhere to certain specifications if it is to be a useful tool for scholars. A working DSS must be able to provide an environment that enables scholars to make decisions based on argumentation. It needs to have the option to edit words, lines and characters. There must be an option to see or input the transcription and define between lines, words and characters. The DSS works with argumentation so it is essential to have the capacity to assign arguments for and against each line, word or character and to be able to assign evidence to those arguments. Finally, arguments should be linked to the text so that if any text changes the argument will pop up and notify the scholar.

The chapter then illustrated how it would be possible to tackle input into the DSS. The method used in the prototype is to input information directly into the MySQL database combined with a pop-up frontend interface.

The structure of the document was then handled. This consists of sections, lines, words and characters. These were mainly input in one go through the MySQL database

backend. Following this was the meaning of the document. The prototype only handles meaning in the shape of letters. Several letters can be attested to each character space but only one can be resolved as the interpreted meaning of that character space.

As mentioned earlier, argumentation provides the best method for organizing/visualizing decisions for the MDTSuite DSS. In this prototype DSS I have added argumentation for letters only. Each letter can have as many arguments **for** and **against** as the scholar finds necessary. This does not affect the value of the letter nor whether it is resolved or not.

Each argument can again be linked to aiding material such as bibliographic references, Web links, or other parts of the same document.

4. THE DSS IN CONTEXT

Chapter 3 demonstrates the DSS prototype using two Vindolanda tablets, 670 and 862, as examples. The prototype is built around section A.i of tablet 670 and this section is used to demonstrate the development throughout Chapter 3. Tablet 862 on the other hand is used as an example to illustrate how the three scholars who are collaborating on the reading of this particular tablet could potentially use the prototype.

The purpose of this current chapter is to show how a DSS can become even more powerful by engaging with XML and Web Service technologies, especially in the data-rich field of ancient documents. First it takes a closer look at the datasets that scholars can search through in the prototype suggestion search. The datasets included in the suggestion search were previously divided according to two different types of searches that might be conducted by a scholar. Concordance searches are done on other document datasets (e.g. Vindolanda Tablets, Inscriptions of Roman Tripolitania or Inscriptions of Aphrodisias) in order to find parallels that can support an argument for a reading in a certain context. Lexical searches, on the other hand, often serve the purpose of verifying whether a certain reading is technically possible, in other words whether the words can be used in that language. This chapter will first use Perseus Digital Library as an example to illustrate how lexical resources can be searched. The next step will be to look at how concordance datasets can be created by using the Vindolanda tablets as an example.

Once the Vindolanda Tablets dataset was created it was necessary to find a mechanism to search the dataset in a meaningful way. The APPELLO Web Service was developed for this purpose. APPELLO is a RESTful Web Service that uses a URL to send different methods to a server that then retrieves the result from the dataset in question.

Following this is a demonstration on how these different search types have been integrated into the DSS prototype through the suggestion engine.

Finally, this chapter will demonstrate how the XML output from the DSS prototype can potentially be re-used, both with APPELLO to become a part of an additional search engine that can be searched for parallels, but also as an integral part of an interactive website. The latter will be illustrated through the new Vindolanda Tablets Online II (VTO2) website.

4.1 LEXICAL RESOURCES

Lexical resources such as dictionaries are invaluable for scholars reading ancient documents. By referring to them in their notes, scholars can adduce the fact that certain words are known in a particular context and that a reading is technically possible and can be justified. The field of classics has been very active at digitising textual and linguistic materials and, in recent years, at making them available online (Bagnall 2009). Among the larger resources are the Perseus Digital Library which includes Classical as well as Arabic, Germanic, Renaissance and 19th Century materials; the Papyrus Portal¹, which searches through all online Papyri in German collections and the Advanced Papyrological Information System², which does the same for mainly American papyrological material. These applications all function as portals enabling users to perform one search and retrieve results from across many decentralised digital datasets. This is a great improvement, as now users do not have to visit every single digital dataset's own website to find the information they are looking for.

But if users wish to re-use the data they have received they would have to rely on copying and pasting this information from their search results into their own spreadsheets, documents, or databases. Instead it would be very useful to be able to access it through an interoperable format such as XML (Roued Olsen 2007). Furthermore, in order for an

¹ <http://www.papyrusportal.de/>

² <http://www.columbia.edu/cu/lweb/projects/digital/apis/>

application such as a DSS to send a pattern to one of these huge datasets and return a list of words fitting this pattern, it needs to be able to connect to a Web Service of some sort. The following will search the dictionaries available through Perseus Digital Library, in particular the Latin dictionaries Lewis & Short and Thomas Cooper, which are used in the DSS prototype.

4.1.1 PERSEUS DIGITAL LIBRARY

Perseus Digital Library is one of the most advanced digital collections of classical texts available at this point in time and it can be accessed both directly from the Perseus website and now also in a faster version via PhiloLogic³. It contains nearly 8 million Greek and 5.5 million Latin words. Through Perseus it is possible to search the Latin dictionaries and with the further aid of the Archimedes project⁴ it is possible to build a URL that can retrieve dictionary results from Perseus in XML format.

This can in essence be thought of as a Web Service even though it does not use best practice in the creation of the URL or the resulting XML. It is clearly not developed for use in external projects. This example will search for the Latin word *frater* ('brother') in Perseus. If the word *frater* is added to the end of the rather cryptic search URL (fig. 4.1) the XML result for *frater* in both Lewis and Short and Thomas Cooper will be returned (fig. 4.2).

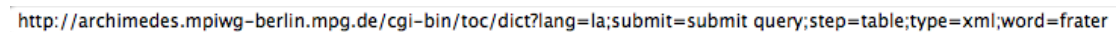


Fig. 4.1 URL sent to Perseus to retrieve lexicon results for the word 'frater'

The resulting XML contains the main tag `<opt>` with the attribute `WORD = "frater"`. This is followed by the results, each in their own `<RES>` tag. Each `<RES>` tag contains the following notable attributes: `AUTHOR` (the author of the dictionary), `LINK` (the URL for the

³ <http://perseus.uchicago.edu/>

⁴ <http://archimedes2.mpiwg-berlin.mpg.de/>

full result page in XML), `DICT` (the dictionary name) and `LEM` (the lemma we are looking for, in this case *frater*).

This result can then be transformed with an XSLT stylesheet and output in a DSS or another application (see 4.4).

```
-<opt WORD="frater">
<RES AUTHOR="Thomas Cooper" BACK="http%3A%2F%2Farchimedes.mpiwg-berlin.mpg.de%2Fcgi-
bin%2Ftoc%2Fdict%3Fflang%3Dla%3Bsubmit%3Dsubmit%2520query%3Bstep%3Dtable%3Btype%3Dxml%3Bword%3Dfrater" DATE="1584"
DICT="Thesaurus Linguae Romanae et Britannicae" DICTID="d002" HEAD="frater" LEM="frater" LINK="http://archimedes.mpiwg-berlin.mpg.de/cgi-bin
/toc/dict?step=entry;head=frater;dict=d002;inword=frater" PROJECT="" WORD="frater"/>
<RES AUTHOR="Lewis and Short" BACK="http%3A%2F%2Farchimedes.mpiwg-berlin.mpg.de%2Fcgi-
bin%2Ftoc%2Fdict%3Fflang%3Dla%3Bsubmit%3Dsubmit%2520query%3Bstep%3Dtable%3Btype%3Dxml%3Bword%3Dfrater" DATE="" DICT="Latin
Lexicon" DICTID="d003" HEAD="fra_ter" LEM="frater" LINK="http://archimedes.mpiwg-berlin.mpg.de/cgi-bin/toc/dict?step=entry;head=fra_ter;
dict=d003;inword=frater" PROJECT="" WORD="frater"/>
</opt>
```

Fig. 4.2 XML received from Perseus for lexicon results of the word 'frater'

4.2 CONCORDANCES

A search in concordances is a search for parallels to help justify a specific reading in a document. It is common practice for scholars to list other documents of relevance that use a word in a similar fashion in the notes of their edition.

Previously, these corpora would have been searched manually through the indexes in monographs and journals in which they were published. Now more and more corpora have become available online for searching and displaying through stand-alone websites. The question is, how is such a digital dataset created out of an originally paper-based database? The Vindolanda Tablets project faced this issue when deciding to publish tablets online. There are several solutions to this and some were attempted for the first version of the website Vindolanda Tablets Online (VTO) which is based on a 4D database system⁵ and a limited amount of XML. For the new version of this website, Vindolanda Tablets II (VTO2) a purely EpiDoc XML based system was adopted. This meant that each of the 736 tablets had to be encoded using XML. In the following I will introduce EpiDoc XML and demonstrate how it can be used to add traditional Leiden encoding to transcripts. This is followed by an introduction to the concept of contextual encoding and a demonstration of

⁵ <http://www.4d.com/>

how this has been used to enrich the XML. Finally, I will demonstrate a script developed for the Vindolanda Tablets, which uses pattern recognition to transform the previous text files into EpiDoc XML.

4.2.1 EpiDOC

Tom Elliott came up with the concept for EpiDoc when he, as a graduate student in the late 1990s, published his work on epigraphical encoding using XML. By the summer of 1999 Elliott and his colleagues were using the Text Encoding Initiative (TEI) to implement XML into epigraphic work (Elliott 2008). However, the TEI Guidelines were developed for texts in general and not for epigraphic texts specifically and this led to the guidelines for the application of TEI to epigraphic texts; in other words the first EpiDoc guidelines. These guidelines were written in January 2001 with the assistance of a group of scholars, among others Charles Crowther from CSAD, who wished to use the TEI guidelines for the Vindolanda Tablets Online (VTO) website. Another epigrapher, Charlotte Roueché, soon became interested in the use of EpiDoc for the publication of the Inscriptions of Aphrodisias (InsAph). It is largely as a part of the InsAph project that the EpiDoc guidelines have been enhanced and circulated (Bodard 2008).

The EpiDoc Guidelines are available online and the newest stylesheet for EpiDoc was retrieved with help from the InsAph project⁶.

4.2.2 LEIDEN CONVENTION ENCODING

Even though most published editions now use the Leiden Conventions there are still differences in the way that is chosen to display the transcripts using brackets and symbols. This means that there will also be different ways of converting the Leiden Conventions into EpiDoc tags. This research uses the conventions described in the

⁶ Many thanks to Gabriel Bodard, Centre for Computing in the Humanities, Kings College London for providing the latest EpiDoc schema and stylesheets as well as guidance in their use.

beginning of Tab.Vindol.III (Table 4.1) to find appropriate tags for each bracket or symbol. This exercise was made more difficult by the editions using symbols and brackets (table 4.2: 13, 14 and 15) not listed in the conventions (Table 4.1). In these cases it was important first to understand the meaning of the symbol and then approximate the most appropriate EpiDoc tag. For example [abc] (table 4.2: 13) indicates that the characters abc were lost and supplied by the editor. For this the <supplied> tag and the attribute reason = "lost" was used.

i, ii	designate separate columns of text following the original layout.
. . .	indicates that the text is broken or incomplete at the top or bottom.
<i>m</i> ¹ , <i>m</i> ²	distinguish different hands in the text.
[]	indicates a lacuna in the text.
[c.4]	estimate of the number of letters missing in a lacuna.
<i>uacat</i>	a space left by the scribe on the tablet.
[[abc]]	letters crossed out or erased by the scribe.
`abc`	letters or words added by the scribe above the line.
<abc>	letters erroneously omitted by the scribe.
{abc}	superfluous letters written by the scribe.
ꜱꜾꜿ	doubtful or partially preserved letters.
.....	subscript dots represents traces of letters visible on the tablet, which have been left unread.
()	expansion or resolution of an abbreviation or symbol, e.g. <i>prae(ecto)</i> , (<i>centurio</i>).

Table 4.1: The editing conventions used in Tab.Vindol.III (Bowman and Thomas 2003: 19).

The <gap> tag is a common occurrence in the EpiDoc encoding of the Vindolanda ink tablets. The <gap> tag can be used for two different reasons. The first reason is `lost` (table 4.2: 1, 3, 4 and 14). This indicates that there should be text in this space, which is lost due to damage to the tablet. The second reason is `illegible` (table 4.2: 11 and 15), which means that there will be traces of characters that could not be read by the editor.

This will be indicated in the conventions by a dot for each character (table 4.2: 11) or in cases where the extent is unknown by the word `traces` (table 4.2: 15). The latter conventions for gaps where the extent is unknown (table 4.2: 15) and the one due to loss where the extent is known exactly (table 4.2: 14) are not found in the Vindolanda conventions list either.

There are three further markings, which the convention list does not cover (table 4.2: 16, 17 and 18). The first one is a downward arrow, meaning that the text has been written across the grain of the wood. This has been added as the attribute `type` to the `<ab>` tag for each section it has occurred in. The interpunct⁷ and the checkmark are also not covered by the conventions list because they are not actual conventions but rather marks made by the scribe. Nevertheless, they have still been encoded by using the `<seg>` tag with the attribute `type` as either `interpunct` or `checkmark`.

The rest of the encoding was more straightforward: `<handshift>` tag when a new hand is introduced (table 4.2: 2); `<space>` tag for a space (table 4.2: 5); `<del>` tag for text erased by the scribe (table 4.2: 6); `<hi>` tag with attribute `superscript="supralinear"` for text added above the line by the scribe (table 4.2: 7); `<supplied>` tag with attribute `reason="omitted"` for text omitted by the scribe (**Fejl! Henvisningskilde ikke fundet.**: 8); `<sic>` tag with attribute `n="superfluous"` for text thought to be superfluous by the editor (table 4.2: 9); `<unclear>` tag for doubtful or broken characters (table 4.2: 10); `<abbr>` and `<ex>` tags inside the `<expan>` tag for abbreviations where the editor has supplied the remaining characters for the abbreviated word (table 4.2: 12).

The Vindolanda tablet XML documents were first encoded according to an older version of the EpiDoc schema. In 2010, work began on a newer version of an EpiDoc schema that would be fully compliant with TEI P5. With the aid of Gabriel Bodard I

⁷ An interpunct (·) is a medial dot sometimes used particularly in ancient Latin documents to indicate word division.

transformed all the previously encoded XML documents to fit this new schema⁸.

The changes were mainly in the header of the XML documents. As an example, a well-used attribute such as `id` became `xml:id`.

1.	. . .	<gap extent="unknown" reason="lost" unit="line" />
2.	M^l, m^2	<handShift new="hand2" n="2"/>
3.	[]	<gap extent="unknown" reason="lost" />
4.	[c.4]	<gap extent="3" precision="low" reason="lost" unit="character"/>
5.	uacat	<space unit="line" />
6.	[[abc]]	<del status="unremarkable">abc
7.	`abc`	<hi read="superscript">abc</hi>
8.	<abc>	<supplied reason="omitted">abc</supplied>
9.	{abc}	<sic n="superfluous">abc</sic>
10.	Aḃç	<unclear>abc</unclear>
11.		<gap extent="6" reason="illegible" unit="character"/>
12.	()	<expan><abbr>praef</abbr><ex>ecto</ex></expan>
13.	[abc]	<supplied reason="lost">abc</supplied>
14.	[...]	<gap extent="3" reason="lost" unit="character" />
15.	traces	<gap reason="illegible" extent="unknown" unit="character"/>
16.	↓	<ab type="vertical"> -attribute added to the ab tag
17.	·	<seg type="interpunct"/>
18.	├	<seg type="checkmark"/>

Table 4.2: The EpiDoc tags used in the Vindolanda tablets

The EpiDoc XML schema that was used to encode the Vindolanda documents is now known as the experimental P5 version⁹. While all the Vindolanda tablet XML documents now validate to this schema there are areas in which this encoding differs from other projects encoded to the same schema. This is mainly due to the continuously

⁸ I want to note that any mistakes in the encoding and any redundancies are completely down to me.

⁹ Experimental EpiDoc P5 Schema: <http://epidoc.googlecode.com/files/exp-epidoc.rng>

experimental nature of EpiDoc encoding. At the time the Vindolanda tablet XML documents were converted to this new P5 version of the EpiDoc schema it was only available in an experimental stage. There is now an official version¹⁰ of this schema available and a test validation of a couple of Vindolanda tablet XML documents against the official schema have been successful.

4.2.3 CONTEXTUAL ENCODING

The initial idea behind contextual encoding was to use the concept of keywords from the VTO website. This resulted in marking up different aspects of the transcription to use in a word search. In 2005 Hippiisley worked on what he called the ECEE (Extended Contextualized EpiDoc Encoding) project for his MA thesis (Hippiisley 2005). He describes the current VTO website as a ‘hypertext retrieval system that acts like a traditional library catalogue, recalling a list of tablet records in which the search term occurs’ (Hippiisley 2005: 13). With the ECEE project he aimed to ‘create a different hypertext retrieval system that allows users to view each occurrence of a word and to direct them to the exact location in which that term occurs in the text’ (Hippiisley 2005:13).

Hippiisley’s idea of contextual encoding was inspired by the InsAph project, which already encoded Greek and Latin words, persons and other indexing terms such as official posts, professions, place names and administrative units. This was then used to create an online set of indices¹¹. For example, when looking up the Greek name, *Μάμας* the result shows that this name is mentioned in document 11.415, section i, line 1 (fig. 4.3).

Figure removed because of missing permissions, see URL:
<http://insaph.kcl.ac.uk/iaph2007/inscriptions/indices/word/word-grcM.html>

Fig. 4.3 Screenshot of results page from I Aph2007 for words beginning with ‘M’ (Reynolds et al. 2007)

¹⁰ Latest Official EpiDoc P5 Schema: <http://www.stoa.org/epidoc/schema/latest/tei-epidoc.rng>

¹¹ Indices page for I Aph2007: <http://insaph.kcl.ac.uk/iaph2007/inscriptions/indices/index.html>

The document ID 11.415.i.1 is linked to the document. Clicking on this link will transfer the user directly to the word in the document (fig. 4.4).

Hippisley chose ten texts from Tab.Vindol.II (Bowman and Thomas 1994), which were first encoded with EpiDoc XML and then further encoded to include contextual

Figure removed because of missing permissions, see URL:
<http://insaph.kcl.ac.uk/iaph2007/iAph110415.html>

Fig. 4.4 Screenshot of text 11.415.i.1 from LAph2007 (Reynolds et al. 2007)

The contextual encoding included personal names, for which he used the `<persName>` tag enveloping the `<name>` tag. This thesis includes the `<persName>` tag but does not make use of the `<name>` tag. The `type` attribute is also used in Hippisley's example with the added `key` attribute as a unique key without spaces to index the personal names with.

```
<persName key="Cerialis_Flauius" type="praefectus">
  Flauius Cerialis
</persName>
```

Hippisley's project used tablets that were already on VTO. This provided him with the keywords from the website. He therefore encoded specific subjects (e.g. foodstuffs, clothing, profession and buildings) as contextual encoding. He used the `<rs>` tag with the attributes `type` and `key` to encode information from VTO subject browsing into the XML documents. Furthermore, each word was encoded individually with the `<w>` (i.e. word) tag and the *lemma* attribute with the Lemma¹² of the word in Latin.

```
<rs type="foodstuff" key="salt"><w lemma="sal">salis</w></rs>
```

¹² A lemma is the root form of a word.

This thesis differs in its approach mainly because it is concerned also with Tab.Vindol.III., which has not had keywords added in the same way as Tab.Vindol.II. Therefore, this attempt at contextual encoding uses the indices in the back of Tab.Vindol.III to add contextual information to each word in the transcription.

This research has used the `<w>` tag for words from the index of general Latin words for which there is a Lemma. The attribute `subtype` indicates an under-type of the lemma, which was found in the index. The need for adding the subtype of the lemma is debatable. Nevertheless, it is an integral part of the indices and in an attempt to be true to the indices of Tab.Vindol.III we have kept them.

```
<w type="lemma" lemma="idem" subtype="eodem">eodem</w>
```

The other Latin words in the transcriptions, which do not have a place in the index of general Latin words and thus do not have a lemma, can instead be found in one of the other indices. Personal names have been covered above and this leaves dates, geography, consuls, military and official terms and abbreviations. This is illustrated with the military/official term `decurio`. This word is enclosed by `<rs>` tags and has the attributes `key="decurio"` and `type="militaryOfficial"`. In this way any instance of this term can be found in the texts by searching for `<rs>` tags of the type `militaryOfficial` with this key.

```
<rs key="decurio" type="militaryOfficial">decurio</rs>
```

The `key` attribute is something most of the encoding has in common and is used in the same way as with the personal names above.

The `<rs>` (i.e. reference string) tag is used as in Hippiisley's project where the attribute `type` covers the type of reference string (e.g. consuls or military and official terms).

4.2.4 TRANSFORMING ENCODING TO XML

Regardless of all the advantages of XML encoding there is one issue that must be

highlighted. XML is mainly a computer-readable language. Humans can in principle understand it, but it is space consuming and confusing to look at. For example `<supplied reason="lost">abc</supplied>` gives us more information than `[abc]`, but the latter is easier to read and much shorter to write. I do not think scholars will ever be persuaded to record their working transcriptions in XML. It is much more likely that the future will present applications like the DSS that will write XML for the scholars behind the scenes. For now, most digitisation projects will most probably begin with text files using Leiden System brackets and symbols that then subsequently have to be converted to XML.

There is an application developed for this purpose, CHET-C¹³ that converts epigraphical text into EpiDoc XML fragments. However, it currently found suitable only for small fragments of very well formed text. Furthermore, for the Vindolanda Tablets project there were 736 tablets that needed transforming and to copy and paste every fragment into CHET-C would have been a very tedious task.

The solution was to develop a new PHP script using regular expressions and pattern recognition. This transforms the text version of a full edition with transcript, description, translation and commentary into an EpiDoc compliant XML document. This approach has moreover been successfully tested on the MAMA XI documents.

The PHP script developed as a part of this research produced an encoding of the transcription together with the EpiDoc header and divisions into description, translation and commentary. By using the raw text versions of each tablet provided by the CSAD it was possible to run this PHP script and thus automate a part of the encoding.

However, this method only enabled us to automatically create XML documents containing the structure and the Leiden encoding. Even so, the Leiden encoding had to be manually checked afterwards, as the original use of symbols and brackets was inconsistent.

¹³ Chapel Hill Electronic Text Converter (CHET-C):
<http://www.stoa.org/projects/epidoc/stable/chetc-js/chetc.html>

Using pattern recognition necessitates that there is a consistent pattern to recognise.

This is an example of how the PHP script works using tablet 291 section i, line 2 which in the original text file with Leiden symbols and brackets is written like this:

```
[sa]<unclear>l</unclear>[u]<unclear>te</unclear>m
```

Note that Charles Crowther from the CSAD kindly transformed under-dots into `<unclear>` tags before he sent me the texts. These `<unclear>` tags remain unchanged through the transformation process.

In this line the script exchanged the square brackets around the letters for the `<supplied>` tag with the attribute `reason="lost"` (table 4.2:13). The pattern in this case is ‘letters enclosed by single square brackets’ (i.e. [sa]). Once this pattern is found, the value (i.e. *sa*) is extracted and added to the tags as such:

```
<supplied reason="lost">sa</supplied>
<unclear>l</unclear>
<supplied reason="lost">u</supplied>
<unclear>te</unclear>
m
```

The next step is contextual encoding which required scholarly attention. The indices in the back of Tab.Vindol.II provided the basis for this encoding but again these indices had not been compiled consistently and thus the input of contextual encoding tags required someone with a good knowledge of Latin. Thus the final encoding of the above word becomes:

```
<w type="lemma" lemma="salus" subtype="salutem">
  <supplied reason="lost">sa</supplied>
  <unclear>l</unclear>
  <supplied reason="lost">u</supplied>
  <unclear>te</unclear>
  m
</w>
```

4.3 APPELLO WEB SERVICE WORD SEARCH

The World Wide Web Consortium (W3C) defines a Web Service as ‘a software system designed to support interoperable machine-to-machine interaction over a network’ (Haas and Brown 2004). In other words Web Services enable heterogeneous systems to

communicate, solving the need for interoperability between these systems. Often XML is used to format the message, as it can be understood by most systems.

The Web has irrevocably changed the way researchers work by giving access to an enormous amount of data. However, there is now a new trend of embracing the Semantic Web by providing interoperability between all this data through technologies that enable reusability (Hall *et al.* 2009). An example of this could be a weather Web Service which can be used by, for example, travel websites. Instead of everyone pointing their browser towards the meteorological office's website when they want the latest weather forecast for their holiday destination, the travel website can include the information through a weather Web Service showing their users instant updates for their particular destination without requiring them to leave the travel website (Abeyasinghe 2008).

Developing a Web Service for the Vindolanda ink tablets will enable users to re-use the data in their own applications (online and offline) in a way that would otherwise be tedious and involve copying and pasting from other websites. At the moment XML documents for each Vindolanda ink tablet are available on VTO for download. This is useful for scholars who wish to view and use each XML document. However, by using APPELLO, users could request all tablets with the word *pullus* (Latin for *chicken*), or a list of all occurrences of the words in the tablets that begin with the characters *pul*. They would then receive the answer as XML, which can be reformatted with the aid of an XSLT stylesheet for the appropriate application.

Another useful aspect of using a Web Service that can extract words and other terms from the Vindolanda Tablets is that this can be used to recreate an index for the entire corpus. Indexing is in itself a very tedious and difficult task (Juola 2008: 78-79) and this provided a method of automatically creating a very thorough and most importantly searchable index of the corpus.

This research uses the RESTful (Representational State Transfer) architecture to

develop APPELLO as it is currently the simplest and most lightweight Web Service architecture. RESTful software architecture was introduced by Roy Fielding in his doctoral thesis in 2000 (Fielding 2000).

The key principles of REST software architecture are as follows:

- Every resource must have a unique ID (e.g. a URI).
- There must be established relationships between resources (interconnectivity).
- Must use standards (e.g. HTTP and XML).
- Resources can have multiple representations for different application states.
- Communication using HTTP should be stateless, meaning that it should not rely on information from previous communication.

HTTP is the transport protocol for Web Services and can be used to access any type of resource on the Web. When using HTTP a uniform resource identifier (URI) is sent which includes the address of the Web Service and any parameters that will influence the result. It is possible to perform different actions with HTTP (e.g. retrieve, send or delete resources) (Abeyasinghe 2008; Fielding 2000; Richardson and Ruby 2007). However, APPELLO is only developed to retrieve data.

This development uses the Zend Framework¹⁴, which is a library of scripts, to build a RESTful Web Service.

4.3.1 METHODS

For each request a unique URL is required which will include the address of the APPELLO Web Service and a set of parameters.

`http://appello-vto2.csad.ox.ac.uk?method=get_tablets`

Each request has a parameter called `method`, and these methods influence the returned result. The `get_tablets` method returns a list of all the tablets within the knowledge base with their `tabletID`, inventory number, author and title. The `tabletID` can

¹⁴ <http://framework.zend.com/>

be used as a parameter in the method `get_tablet` in order to receive the full XML for a certain tablet. This provides interconnectivity between the different requests in the Web Service (fig. 4.5).

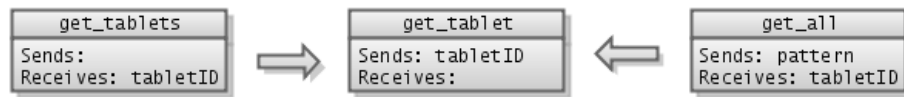


Fig. 4.5 Model of the interconnectivity between the methods.

```

- <item n="581">
- <ab type="images">
- <figure>
  <head>581f</head>
  </figure>
- <figure>
  <head>581b</head>
  </figure>
</ab>
<ident type="inventory">93.1474I-V</ident>
<ident type="category">Accounts</ident>
<title>Vindolanda writing tablet number 581</title>
<ref>?method=get_tablet&tabletID=581</ref>
- <bibl n="Tab.Vindol.III">
  A.K.Bowman, J.D.Thomas, The Vindolanda Writing Tablets (Tabulae Vindolandenses III). London: British Museum Publications, 2003
</bibl>
- <ab type="measurements">
  <width unit="mm">40</width>
  <height unit="mm">93</height>
</ab>
<ab type="plates">2-3</ab>
<ab type="archLocation">SG (bonfire site)</ab>
<ab type="archPeriod">3</ab>
</item>
  
```

Fig. 4.6 XML result for the APPELLO method `get_tablets` displaying the record for tablet 581

The Vindolanda Tablet 581 is used as an example in the following. A search using the `get_tablets` method returns a list of tablets (fig. 4.6). Tablet 581 is linked to two images, 581f (of the front) and 581b (of the back). This XML record also includes other information such as the inventory number (i.e. 93.1474I-V), the location of the tablet on the site (i.e. SG Bonfiresite) and the period to which the tablet is attributed (i.e. 3). The XML document returned is a TEI compliant XML document. The first tag is `<TEI>` followed by a header `<teiHeader>` that contains information about the resulting XML document itself (e.g. the title):

```
<title> APPELLO method: Get Tablets </title>
```

On the other hand the method `get_tablet` and the parameter `tabletID=581` returns the entire EpiDoc encoded XML document for tablet 581 (fig. 4.7). As EpiDoc is a type of TEI this document is also a TEI formatted document beginning with the `<TEI>` tag.

```
- <TEI n="581">
  <!-- HEADER -->
  - <teiHeader>
    - <fileDesc>
      - <titleStmt>
        - <title>
          Vindolanda writing tablet number
          <rs type="tabletNum" key="581" n="1">581</rs>
        </title>
      </titleStmt>
```

Fig. 4.7 XML result of the APPELLO method `get_tablet` and parameter `tabletID = 581`

The methods `get_word`, `get_rs`, `get_person`, `get_geog` and `get_all`, will return all the words in each category. They are also interoperable with the `get_tablet` method as they return a `tabletID` together with the number of occurrences of each word in the tablet. This example will search the method `get_words` with `pattern = pullus` (fig. 4.8). The lemma *pullus* occurs 15 times in tablet 581, three times in tablet 582 and once in tablet 616 and 679. However, the word occurs in different variations each time. In tablet 582 the variant is *pulli* and *pullos*. The lemma binds the two variants of this word together with a common reference. Again the resulting XML document is encoded with TEI.

```
- <item n="582">
  <ident type="tabletID">582</ident>
  <num type="amount">2</num>
  <w n="1">pulli</w>
  <w n="2">pullos</w>
</item>
- <item n="616">
  <ident type="tabletID">616</ident>
  <num type="amount">1</num>
  <w n="1">pullos</w>
</item>
- <item n="679">
  <ident type="tabletID">679</ident>
  <num type="amount">1</num>
  <w n="1">pullum</w>
</item>
```

Fig. 4.8 XML result of the APPELLO method `get_words` and parameter `pattern = pullus`

4.3.2 PATTERNS

Some of the methods described above have a second optional parameter called `pattern`, which enables users to specify a pattern that the words returned must include. If, as in the above example, the parameter is `pattern = pullus` combined with `method = get_words`, APPELLO will return all the instances of the lemma *pullus*. However, using `pattern = pul` returns all the words with this pattern, including all instances of *pullus*.

```
- <text>
  - <body>
    - <list type="words">
      + <item n="idem"></item>
      + <item n="adimo"></item>
      + <item n="viridis"></item>
      + <item n="quidem"></item>
    </list>
  </body>
</text>
```

Fig. 4.9 XML result from APPELLO method `get_words` with parameter `pattern = *[bd]em`

Furthermore, the pattern enables searches for incomplete character sequences. The symbol `*` is a character wildcard. On the other hand, square brackets surrounding a set of letters signify that in one certain character space it is possible to have either of the letters in the square brackets. For example, `[bd]` means that either *b* or *d* can be in this character space. Therefore, a search for the pattern `*[bd]em` will return a variety of words, among others, the word *idem* (fig. 4.9).

4.4 SUGGESTION ENGINE FOR DSS

As I established earlier, searching through huge datasets is another thing that computers are better at than humans. Searching through millions of records to find one that might match the current interpretation is a near impossible task for a human so this is where a computer can be really beneficial.

There are two different types of searches used by scholars. The first is a search through dictionaries or lexical resources. The reason for this type of search is often to see if

a word exists in the first place. In other words, to decide whether the use of the word is technically possible.

The second type of search is that of concordances. In this case scholars would search through comparable corpora to find parallels for the use of a word in certain contexts.

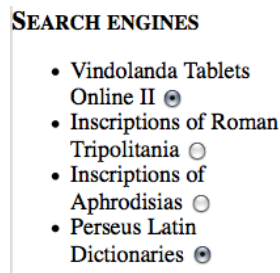


Fig. 4.10 Screenshot of the search engine selector in the DSS prototype

Both types of searches are incorporated into the DSS as suggestion engines. In the left sidebar (fig. 4.10) on the main DSS view it is possible to select and deselect the resources that can be used in a search. In this prototype, searching suggestion engines can only be done on the word level.

The suggestion engines that can be used in this prototype are all stored in the table `DSS_engines` (table 4.3). Here each search engine has its own `ID`. The field `webservice` stores the URL with the parameters needed to search for the word. The only thing missing in each of these URLs is the value of the last parameter – the word that is going to be searched for. In the first example the URL is:

```
http://appello-vto2.csad.ox.ac.uk?method=get_word&pattern=
```

The search term is added to the end of the URL to the parameter `pattern` in the DSS script.

The field `document` stores a universal link to the search engines result page. In the first case the URL is:

```
http://vto2.classics.ox.ac.uk/index.php/tablets/search-for-tablets?
submit=View&tablet=
```

If a tablet number from the Vindolanda Tablets is added to the end of this URL it will become a link to the online version of this document. The other search engines do not have a document field value listed. This is either because these documents are not available online or because as is the case with the last one (i.e. Perseus Latin Dictionaries) the link to the relevant web resource is encoded into the search result. The field `title` contains the title of the search engine and the field `used` can have the value 1 (i.e. this search engine is being used in the prototype) or 0 (i.e. the search engine is not being used). The final field `suggestionType` can have either one of the values `concordance` or `lexicon` depending on the type of search engine as described above.

ID	webservice	document	title	used	suggestionType
1	http://appello-vto2.csad.ox.ac.uk?method=get_word&...	http://vto2.classics.ox.ac.uk/index.php/tablets/se...	Vindolanda Tablets Online II	1	concordance
2	http://localhost/appello_irt/?method=get_word&patt...		Inscriptions of Roman Tripolitania	0	concordance
3	http://localhost/appello_insaph/?method=get_word&p...		Inscriptions of Aphrodisias	0	concordance
5	http://archimedes.mpiwg-berlin.mpg.de/cgi-bin/toc/...		Perseus Latin Dictionaries	1	lexicon

Table 4.3 DSS_engines table from the DSS_670 database

Using this suggestion, search functionality requires that there are datasets online that can be searched. However, even if there are datasets available online this DSS prototype may still not be able to use them, as they may only be searchable through a HTML form on the datasets website.

This is why the word search Web Service, APPELLO, has been developed in order to show how Web Service technology could be used on datasets encoded in EpiDoc TEI. Depending on the search engines selected on the left side, a pattern will be sent off to each of them and the results that fit this pattern will return as XML. The XML retrieved is formatted as TEI with a tree of `<list>` tags and `<item>` tags. The prototype uses an XSLT style sheet to transform this into a list that can be displayed to the user in the word editing

pop-up box.

As an example, to demonstrate the use of the in-built APPELLO pattern search, the Vindolanda Tablets will be searched for the word *facio* (word 2 in line 5, tablet 670) (fig. 4.11) (See Example 3, Appendix C). The yellow highlight of *fa* signifies that these two letters are proposed but not resolved. The prototype converts this to the pattern ***cio*. The symbol *** signifies to the search engine that the first two characters can be any letter. This pattern will return a longer list of words with anything from *facio* to *lucio*. Here it is up to scholars to evaluate which of the suggestions would fit best into the context of this document and this phrase.



Fig. 4.11 Screenshot of line 5 in tablet 670 in the DSS prototype

The search engine returns the result as XML (fig. 4.12), which is then transformed with the XSLT stylesheet into a list of lemmas and their sub-lemmas (fig. 4.13).

```

- <list type="words">
  - <item n="facio">
    <ident type="lemma">facio</ident>
  - <list type="tablets">
    - <item n="156">
      <ident type="tabletID">156</ident>
      <num type="amount">1</num>
      <w n="1">faciendum</w>
    </item>
  
```

Fig. 4.12 XML returned from a search of the Vindolanda Tablets APPELLO with the search pattern ***cio*

Results from: List of words from Vindolanda Writing Tablets

- **lemma:** Lucio [580](#),
- **lemma:** adicio
 - **sub-lemma:** adicierunt [615](#),
- **lemma:** afficio
 - **sub-lemma:** adficio [613](#),
- **lemma:** cal(e)facio [645](#),
- **lemma:** efficio [645](#), [657](#),
- **lemma:** ex(s)arcio [233](#),
- **lemma:** facio [156](#), [157](#), [230](#), [256](#), [270](#), [291](#), [297](#), [311](#), [343](#), [348](#), [371](#), [652](#), [670](#),

Fig. 4.13 Screenshot of the result of a search of the Vindolanda Tablets APPELLO with the search pattern '**cio' displayed in the DSS prototype

Another pattern feature is the square brackets: `[abc]` indicating a character space, that can contain either of the letters *a*, *b*, or *c*. This can be used in relation with the word *cogn.tis* to clarify which of the three letters *a*, *u*, or *i* would fit best in the fifth character space. The pattern for this would be `*ogn[aiu]t*s`. Again `*` takes the place of the uncertain letters and `[aiu]` replaces the fifth character space.

As the list of results (fig. 4.13) demonstrates, the prototype provides a link to the full context of the words that are retrieved during the search. This link is either created with the URL from the `document` field (table 4.3) or from a URL inherent in the XML result. In the first example with the search for `**cio` in the Vindolanda documents, the lemma *Lucio* is found in tablet 580. Clicking on the link 580 opens up tablet 580 on the Vindolanda Tablets Online II Web resource (fig. 4.14).

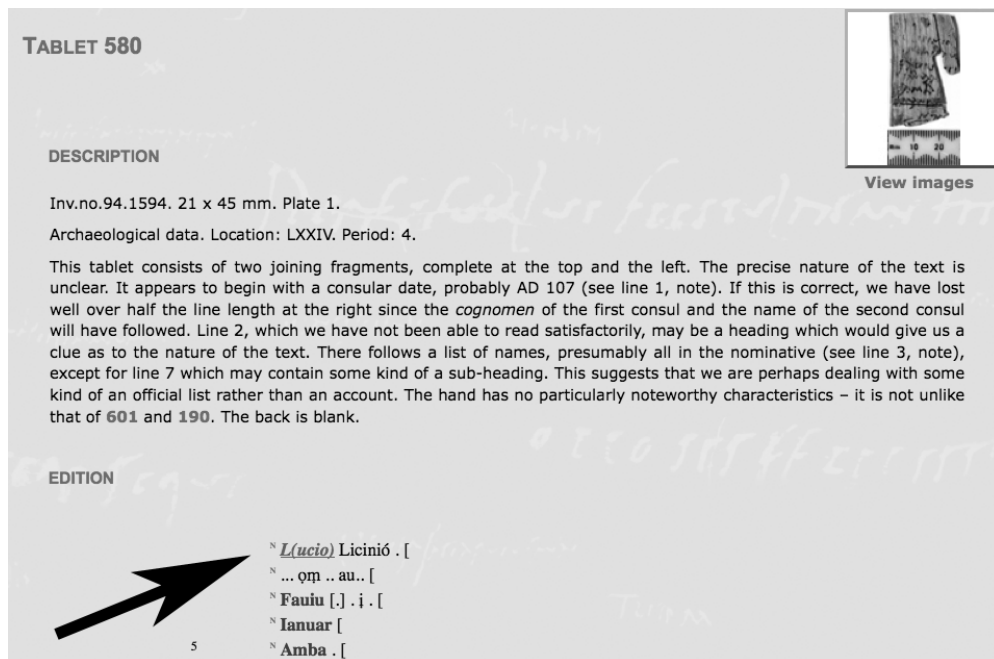


Fig. 4.14 Screenshot of the record for tablet 580 on the Vindolanda Tablets Online II website. Used with permission of the Centre for the Study of Ancient Documents, University of Oxford.

As another example the ‘Perseus Latin Dictionaries’ search engine can be selected and by clicking on the word *fratri* in line 1, tablet 670 a search through the lexicon resource for the word *fratri* is initiated (See Example 4, Appendix C). The result would be the word *frater* in either the ‘Thomas Cooper’¹⁵ or the ‘Lewis and Short’¹⁶ lexicons¹⁷ (fig. 4.15). These two results are also linked to an online resource for the word *frater* that displays the meaning of the word, *brother* (fig. 4.16). This example is again quite simplified but it does show how external lexicons can be used to back-up the technical possibility of an interpretation.

As previously mentioned there are different reasons to search related text and lexicons and tablet 862 will now be used to demonstrate when and how scholars would use these search methods.

¹⁵ Thesaurus Linguae Romanae et Britannicae

¹⁶ A Latin Dictionary

¹⁷ While Lewis and Short is a widely used Latin dictionary for modern scholars Thomas Cooper is more of an antiquarian item now. However, these are the two Latin dictionaries published by Perseus at the time of this prototype development, which explains why they are used in this example.



Fig. 4.15 Screenshot from the DSS prototype of the word edit pop-up for the word 'fratri' and the suggestion search result

Figure removed because of missing permissions, see URL:

<http://www.perseus.tufts.edu/hopper/text?doc=Perseus%3Atext%3A1999.04.0059%3Aentry%3Dfrater>

Fig. 4.16 Screenshot of the Perseus Latin Dictionaries page for the word 'frater' in the Lewis and Short Latin Lexicon

The following example is from line 9, which in the final edition (Appendix A) reads:

Line 9: item ferrum [p]roductum

In order to justify the use of the word *[p]roductum* where so many of the letters can not be clearly read (hence the under-dots and square brackets) the scholars resort to a lexical search of *productum*.

In the DSS, clicking and opening the word editing pop-up box and searching for this particular word in the lexicon chosen in the main view left sidebar accomplishes this search. In this example the scholars could choose to search the Perseus Latin Dictionaries for any results relating to *productum*. This search would not return any results because it is

too technical a term. The scholars could instead try searching again through Lewis and Short for the derived adjective *productilis* and find a definition of this: “*of beaten work, of highly wrought metal*” (fig. 4.17), confirming that the word *productum* is technically possible. However, it does not show whether this word would necessarily work in this particular context.

Figure removed because of missing permissions, see URL:
<http://www.perseus.tufts.edu/hopper/text?doc=Perseus%3Atext%3A1999.04.0059%3Aentry%3Dproductilis>

Fig. 4.17 Screenshot of the Perseus Latin Dictionaries page for the word ‘productilis in the Lewis and Short Latin Lexicon

For this the scholars would have to conduct a concordance search of other corpora for parallels. In this case they searched Juvenal¹⁸ and found in *Sat.* XV, 165-6 the use of *produxisse* ‘in the sense of forging ‘deadly iron’ into weapons’ (Appendix A). In the same way a search in Cyprian¹⁹ returns in Ep. 69.8 ‘*turibula quoque ipsa ... conflata atque igne purgata in laminas ductiles producuntur* (‘[pagan] censers are melted and purified by fire, before being drawn out into metal sheets’)’ (Appendix A). These two search results illustrate parallels to the use of *productum* in the context of metalwork. The scholars can then add these different search results to the arguments for the word *productum* as bibliographic links by clicking on the <Show Arguments> button, which opens the argument editing pop-up box. Here they first click the <+ Add an argument> button and add in the text box their argument, which is that ‘the word *productum* works in the context of

¹⁸ Juvenal is the English name for the Roman poet, Decimus Iunius Iuvenalis (late 1st – early 2nd Century CE), author of the Satires (Sat.).

¹⁹ Cyprian is the English name for the Bishop of Carthage, Thascius Caecilius Cyprianus (3rd Century CE), author of a number of epistles (Ep.)

metalwork’. They then click the `<argument for>` radio button to indicate that this is an argument for the word *productum*. Then they click the `<Add Bibliographic link>` button and add the bibliographic reference (i.e. Juvenal Sat. XV, 165-6) and optionally a link to this reference if it was available on the Web. Once added this link would be displayed as ‘Bibliographic link: Juvenal Sat. XV, 165-6’, possibly linked to the relevant Web page.

4.5 OUTPUT FROM THE DSS

Chapter 2 discusses the task of publishing editions and how this could be a rather tedious task for scholars. With the use of XML, editions became much easier to publish and re-publish in different formats including monographs and websites. However, producing XML can in itself be a tedious task too.

In section 4.2.4 I demonstrated the script I wrote to transform the Vindolanda Tablet editions from text-based files to EpiDoc encoded XML documents. This did speed up the process of creating XML considerably, however it still required scholars to manually add contextual encoding.

For the DSS I envisage a system where the document that is being worked on can automatically be exported as XML or any other format that would be useful to the scholars.

Tablet 670 will be used to demonstrate how an EpiDoc encoded XML document is built up of different parts (i.e. `<div>` tags) in the same way that a finished edition is. However, let me first note that as tablet 670 is a Vindolanda document it only demonstrates how the editions for Vindolanda tablets are published. Editions for different resources sometimes use different elements or name similar elements differently. The Vindolanda editions most often have the following elements: description (including physical attributes and the context where it was found), transcription of the text read, translation, and line-by-line commentary (Appendix A).

This structure is replicated in the XML document with the TEI/EpiDoc tags: `<div`

```
type="description" xml:lang="eng">, <div type="edition" xml:lang="lat">, <div
type="translation" xml:lang="eng">, and <div type="commentary" xml:lang="eng">.
```

The current XML output available in the prototype only exports the structure and meaning of the document through the DBtoTEI Web Service (fig. 4.18).

```
- <l n="1">
  - <w n="1">
    - <c n="1">
      <g ana="#resolved">M</g>
    </c>
```

Fig. 4.18 XML returned from the DBtoTEI Web Service for part of line 1

This DBtoTEI structured XML is rather more detailed than that of the same line (line 1, tablet 670) and word in the EpiDoc encoded XML for tablet 670 (fig. 4.19). Nevertheless, this former XML could easily be transformed into the latter by removing the `<g>` and `<c>` tags and adding a more contextual tag (e.g. `<persName>`) instead of `<w>`. So in essence outputting the actual transcription of the document from the DSS prototype is fairly simple and as this is the most tedious element of creating an EpiDoc or TEI encoded XML document for an edition it would definitely be worthwhile to pursue this area further.

```
<lb xml:id="line.i-1" n="1"/>
<persName key="martius" nymRef="Martius" n="1"
rendition="Martius"
rend="librarius">Martius</persName>
```

Fig. 4.19 XML from line 1 of the original edition on tablet 670

Another part of an EpiDoc encoded XML document that could easily be output from the DSS prototype is the line-by-line commentary. The arguments for and against each word and letter are already linked to each line and can therefore be easily output as line-by-line commentary. Depending on how each argument is written, this commentary may need editing to make it flow better.

Outputting the translation and description of each document as XML is not covered by the prototype as there is currently no way to input this into the DSS. However, in the

EpiDoc XML document these are mainly just pieces of text with surrounding `<div>` tags therefore they are not nearly as tedious to input as the transcription and the commentary.

The major advantage of having a DSS application that can export an interpretation as XML is the way in which such XML can be re-used. This is demonstrated by first explaining how the XML encoded Vindolanda tablets have been re-used through APPELLO. Followed by a demonstration of how these same XML documents together with APPELLO enable a more interactive website by using the new Vindolanda Tablets Online II (VTO2) website as an example.

4.5.1 RE-USE OUTPUT WITH APPELLO

Once the Vindolanda tablets were encoded with both Leiden conventions and contextual encoding the next step was to find a method of searching through them and extracting the index information to create an interactive indexing system. The method that came to mind was inspired by my MSc dissertation *Heritage Portals and cross-border data interoperability* (Roued Olsen 2007) and resulted in the development of the Web Service, APPELLO.

Developing APPELLO to search through the Vindolanda tablets required a clear strategy of what information scholars wished to be able to search for in each tablet and what information they should be able to extract and display in the search result.

The functionality of APPELLO has already been described in 4.3. The following will demonstrate how the Vindolanda Tablets XML documents have been re-used with APPELLO by reverting back to the example in 4.3.1 where the APPELLO method `get_words` was used to find instances of the word *pullus*. The live URL for this is:

http://appello-vto2.csad.ox.ac.uk/?method=get_word&pattern=pullus

The search result is produced through a PHP script that searches through each and every Vindolanda Tablet for `<w>` tags that contain the letter combination *pullus* in the

lemma attribute. The result of this search is then added to the TEI tags `<item>` and returned as a part of the APPELLO result (fig. 4.8).

In the above example the PHP script would search each tablet in the Vindolanda dataset until it reached the first instance of a `<w>` tag where the attribute `lemma="pullus"`:

```
<w type="lemma" lemma="pullus" >pulli</w>
```

The PHP script adds this new instance to the resulting XML document as a new `<item>` tag with the attribute `n="581"` for the tablet number. Each instance of the `<w>` tag with `lemma="pullus"` in the tablet is then added to a `<w>` tag within the `<item n="581">` tag of the APPELLO XML:

```
<item n="581">
  <ident type="tabletID">581</ident>
  <num type="amount">15</num>
  <w n="1">pulli</w>
  <w n="2">pull-</w>
  ...
  <w n="15">pulli</w>
</item>
```

By using an XSLT stylesheet, this XML that has been retrieved can be represented through HTML as a list, a drop-down menu or in any other format desired. The following chapter will demonstrate how this has been used to build the new VTO2 website.

4.5.2 RE-USE OUTPUT WITH INTERACTIVE WEBSITES

The original Vindolanda Tablets Online (VTO) website²⁰ was launched in 2003 as an online exhibition and a way of publishing the editions of Tab.Vindol.II. The editions were complete with high quality digital images and included the transcriptions, translations and notes for each edition (fig.4.20).

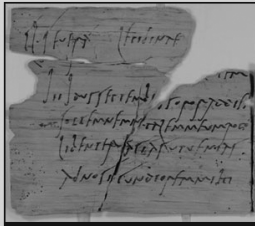
²⁰ Project directors: Alan Bowman, Charles Crowther and John Pearce. Team: Paul Groves, Jessica Ratcliff, Joseph Talbot, Frédérique Landuyt, Peter Haarer, Margaret Sasanow, and David Thomas.

Vindolanda Tablets Online Tablets Exhibition Reference Help

search results | new search

Tablet 291

Image: Leaf No. 1 (front)



open image zooming viewer

Vindolanda Inventory No. 85.057

Introduction

This diptych contains a letter to Sulpicia Lepidina from Claudia Severa, wife of Aelius Broccus, sending Lepidina a warm invitation to visit her for her (Severa's) birthday (on the celebration of birthdays by private individuals see RE VII, 1142-4) and appending greetings to Cerialis from herself and greetings from her husband. The elegant script in which this letter is written is also probably to be recognised in 243, 244 and 248. The letters are slim, with marked ascenders and descenders, and very little use of ligature. There is occasional use of the apex mark for which see pp.57-61, above. In the present text the use is not always in long quantities. It is quite certain that the author is Severa herself, adding a brief message and the closing greeting in her own hand as she also does in 292 and 293. Almost certainly, therefore, these are the earliest known examples of writing in Latin by a woman.

Latin text

i

1 Cl(audia) · Seuerā Lepidinae [suae

2 [sa]l[u]tem

3 iii Idus Septembr[e]s soror ad diem

4 sollemnem natalem meum rogō

5 libenter faciās ut uenias

6 ad nos iucundiores mihi

ii

7 [diem] interuentū tuo facturā si

8 [-].[-c.2]s uacat

9 Cerial[em t]uum salutā Aelius meus .[-]

10 et filioli saluant uacat

11 m2uacat sperabo te soror

12 uale soror: anima

13 mea ita ualeam

14 karissima et haue

Back

15 m1 Sulpiciae Lepidinae

16 Cerialis

17 a S[e]uera

Notes

open notes viewer

TVII Abbreviations and Bibliography

addenda

print-friendly tablet view

English translation

"Claudia Severa to her Lepidina greetings. On 11 September, sister, for the day of the celebration of my birthday, I give you a warm invitation to make sure that you come to us, to make the day more enjoyable for me by your arrival, if you are present (?). Give my greetings to your Cerialis. My Aelius and my little son send him (?) their greetings. (2nd hand) I shall expect you, sister. Farewell, sister, my dearest soul, as I hope to prosper, and hail. (Back, 1st hand) To Sulpicia Lepidina, wife of Cerialis, from Severa."

Fig. 4.20 Screenshot of tablet 291 on the VTO website. Used with permission of the Centre for the Study of Ancient Documents, University of Oxford

Vindolanda Tablets Online Tablets Exhibition Reference Help

The fort plan

you are here: home | exhibition | Forts and military life | **The fort plan**

Vindolanda and its setting

History

Forts and military life

The fort plan

Soldier's lives - military routines

Soldiers and builders

Manufacture and repair

Transport and supplies

Diet and dining

Clothing

Birthdays and gods

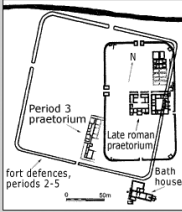
People

Documents

Reading the tablets

about this exhibition

The writing tablets have been excavated from levels associated with the pre-Hadrianic, timber forts at Vindolanda. It is difficult to reconstruct the plans of these because the remains of the later Roman stone forts and the vicus lie over them. The figure shows the excavated buildings and a possible outline of the period 3 fort, although only the line of the southern and western fortifications has been established with confidence. When we compare this fort with the later stone fort at Vindolanda and with Roman forts elsewhere, the position of the *praetorium* seems unusual. We might expect it to lie in the central strip, east-west across the fort, to one side of the headquarters building, rather than on the road to the south gate.



Click on the image for a larger version.

Image details:

The period 3 fort (outline), in relation to the overlying stone built fort. The figure shows the known positions of two buildings occupied in period 3, the praetorium within the fort and the bath house, outside its south-east corner.

Image ownership:

Vindolanda Trust, with modifications

Fig. 4.21 Screenshot of a page in the online exhibition on the VTO website. Used with permission of the Centre for the Study of Ancient Documents, University of Oxford

The exhibition included in the website thoroughly introduces the tablets, their content and context (fig. 4.21).

The VTO website is intended for a wide audience ranging from anyone with an interest in the classical world to experienced scholars with a particular interest in particular editions.

```
<div type="edition" lang="lat">
  <ab n="text">
    <pb n="i"/>
    <lb id="line-1" n="1" /> Cl(audia) &middot; Seuer&aacute; Lepidinae [suae
    <lb id="line-2" n="2" /> [sa]<unclear>l</unclear>[u]<unclear>te</unclear>m
```

Fig. 4.22 Screenshot of a part of the XML document for Vindolanda tablet 291 as it was encoded for the VTO website. Used with permission of the Centre for the Study of Ancient Documents, University of Oxford.

The transcriptions were first encoded in a low detailed-version of EpiDoc (fig. 4.22) and then split up and added to a 4D database. Mark-up was used to divide the text into lines and paragraphs but leaving Leiden convention encoding such as square brackets in place instead of replacing them with the respective tags. The only exception to this rule was the use of the `<unclear>` tag for under-dotted letters.

Fig. 4.23 Screenshot of the search facilities of the VTO website. Used with permission of the Centre for the Study of Ancient Documents, University of Oxford.

The website enables users to either search or browse through the tablets. They can search through the Latin text (i.e. transcription) or general text throughout the document (fig.4.23).

Browsing provides lists of *highlights, subjects, categories, people, places, types, publication numbers* and *military terms*. By using this browse facility, tablets that contain references to clothes can be displayed and this can be further narrowed down to references to socks (fig. 4.24) to find one particularly well-known letter (tablet 346).

Clothing	25 in total
<u>All results</u>	
Clothing results by subheadings:	
<u>belt</u>	1
<u>bootnails</u>	1
<u>buskin</u>	1
<u>cloak</u>	7
<u>overcoats</u>	1
<u>sandals</u>	1
<u>shoes</u>	1
<u>socks</u>	1
<u>thongs</u>	1
<u>tunic</u>	5
<u>underpants</u>	1
<u>vest</u>	2

Fig. 4.24 Screenshot of the browsing facility of the VTO website. Used with permission of the Centre for the Study of Ancient Documents, University of Oxford.

This browsing functionality has been manually added to the 4D relational database and extending this to more tablets would mean a great deal of manual input in the database.

The following will demonstrate how the VTO2 website works through the use of the new encoding of the tablets, the APPELLO Web Service and AJAX Live Search functionality (See Example 5, Appendix C). The first page is Tablets > Search for Tablets (fig.4.25).



Fig. 4.25 Screenshot of the basic tablet search on the VTO2 website. Used with permission of the Centre for the Study of Ancient Documents, University of Oxford.

On this page the APPELLO method `get_tablets` is used to create a drop down menu with tablet numbers and inventory numbers that link directly to the tablet views

themselves. If the tablet number is known then this is the fastest method of retrieving the tablet. If on the other hand the tablet number is not known, it is possible to browse tablets based on their *inventory number*, *category*, *publication*, *plates*, *archaeological find location* or *archaeological find period* from the [Tablets > View all tablets](#) page.

The header enables sorting by *Inventory*, *Category*, *Reference*, *Plates*, *Arch. Location* or *Arch. Period* (fig.4.26).

View all							
Tablet ID	Inventory	Category	Reference	Dimensions	Plates	Arch. Location	Arch. Period
178	85.043	Accounts	Tab.Vindol.II (ref)	89mm x 69mm	XVC		
179	85.252	Accounts	Tab.Vindol.II (ref)	180mm x 37mm			
180	88.943	Accounts	Tab.Vindol.II (ref)	77mm x 264mm	VIII		
181	88.944	Accounts	Tab.Vindol.II (ref)	70mm x 137mm	X		
182	88.947	Accounts	Tab.Vindol.II (ref)	174mm x 77mm	IX		
183	89.978	Accounts	Tab.Vindol.II (ref)	47mm x 68mm			
184	87.622	Accounts	Tab.Vindol.II (ref)	210mm x 70mm	XII		
185	88.950	Accounts	Tab.Vindol.II (ref)	60mm x 171mm	XI		
186	87.567	Accounts	Tab.Vindol.II (ref)	60mm x 167mm	VIII		

Fig. 4.26 Screenshot of the browsing facility on the VTO2 website. Used with permission of the Centre for the Study of Ancient Documents, University of Oxford.

It is also possible to filter the tablets by *Category*, *Reference*, *Plates*, *Arch. Location* or *Arch. Period* by clicking on the relevant term. For example, to see all tablets in period 3 the user can click on 3 in the column *Arch. Period* (fig.4.27).

View all

Tablet ID	Inventory	Category	Reference	Dimensions	Plates	Arch. Location	Arch. Period
581	93.1474I-V	Accounts	Tab.Vindol.III (ref)	40mm x 93mm	2-3	SG (bonfire site)	3
582	93.1299a-b + 93.1298a	Accounts	Tab.Vindol.III (ref)	190mm x 30mm	4	SG/NE (intervallum road)	3
583	93.1474a	Accounts	Tab.Vindol.III (ref)	37mm x 86mm	4	SG (bonfire site)	3
584	93.1480	Accounts	Tab.Vindol.III (ref)	36mm x 88mm		SG (bonfire site)	3

Fig. 4.27 Screenshot of the browsing facility on the VTO2 website filtered to show tablets from archaeological period 3 only. Used with permission of the Centre for the Study of Ancient Documents, University of Oxford.

The tablet IDs in the left most column are all linked to the tablet view. This browsing list uses the APPELLO method `get_tablets` in the same way as the former page.

The greatest strength of VTO2 lies in the index searcher demonstrated on the page `Indices > General Index of Latin words`. This facility uses the APPELLO method `get_words` together with AJAX Live Search to search the tablets for Latin words (i.e. words in `<w>` tags) as letters are typed into the search box, in the manner now familiar from Google Instant²¹.

The following example is a search for tablets containing the Latin word *soror* ('sister'). By first typing *so* the search engine returns suggestions with words containing *so* (fig. 4.28). By adding *ro* to the end of this the results narrow down to words containing *soro*. In this example *soror* is the only word containing the second letter combination.

GENERAL INDEX OF LATIN WORDS

consobrinus

consobrinus 761(1)

consolor 663(1)

soror 291(3) 292(2) 293(1) 310(1) 389(1) 635(1) 661(1)

Fig. 4.28 Screenshot of index searching facility on the VTO2 website. Used with permission of the Centre for the Study of Ancient Documents, University of Oxford.

²¹ <http://www.google.com/instant/>

This type of search using AJAX Live Search can be a powerful tool when searching for a particular word where there is uncertainty about the spelling or perhaps different variants of the word. As a further example it is possible to search for *sum* and receive all the variations of *sum* (fig. 4.29).

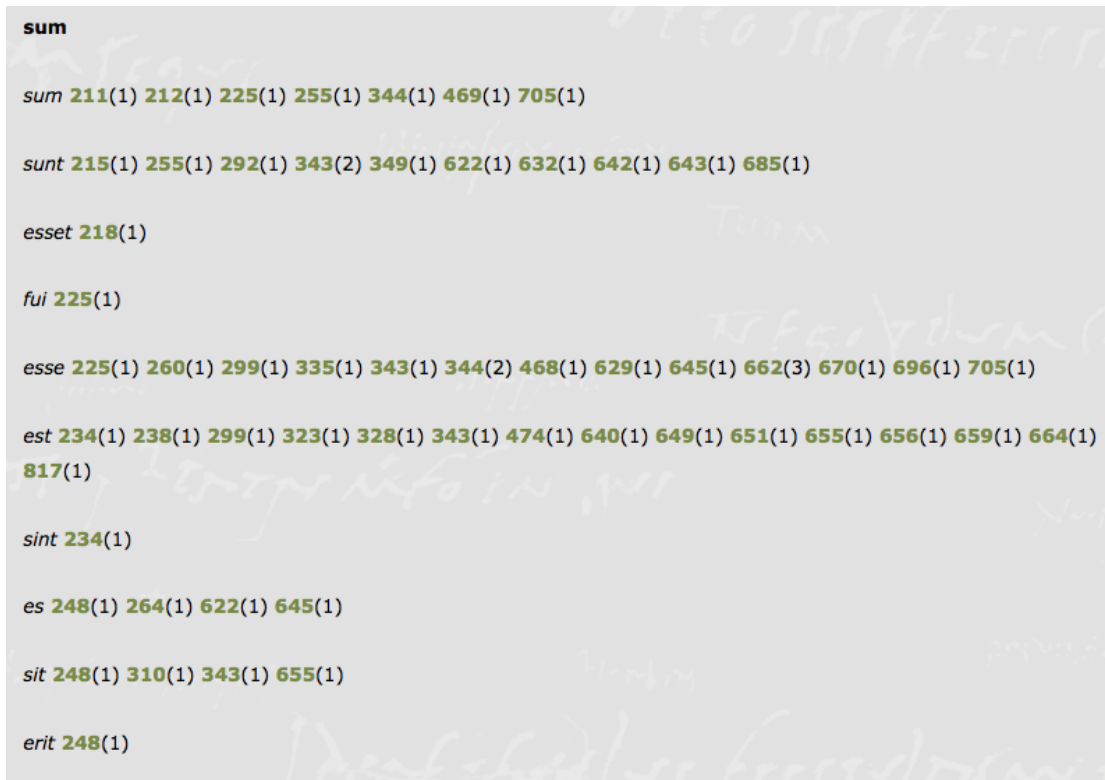


Fig. 4.29 Screenshot of the result from a search on the VTO2 website. Used with permission of the Centre for the Study of Ancient Documents, University of Oxford.

Each result listed includes the tablets containing the word and the occurrences of the word in each tablet. The *tabletIDs* are linked to the first occurrence of the word in the tablet (fig. 4.30).

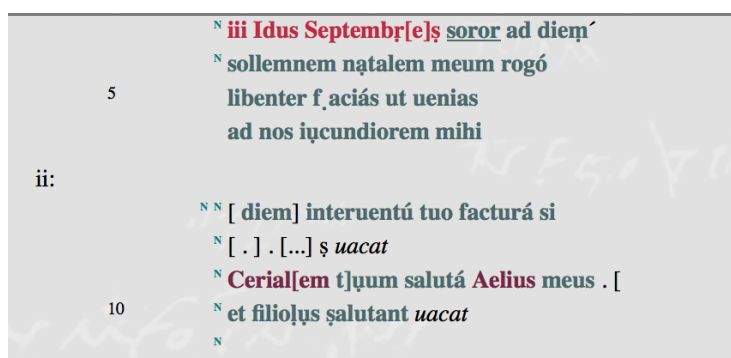


Fig. 4.30 Close-up screenshot of tablet 291 on the VTO2 website. Used with permission of the Centre for the Study of Ancient Documents, University of Oxford.

When clicked, the user is brought to a tablet view that uses the XML document retrieved with APPELLO method `get_tablet`.

The XSLT style sheet lists the different parts of the document (i.e. `<div>` tags on the page) starting with the figures. In the example above there are two images, one showing the front of the tablet and one showing the back. The images open up in a separate pop-up viewer using the Lightbox 2²² application.

This is followed by the description in which bibliographical references (yellow) and other tablet numbers (turquoise) are highlighted and linked to the bibliography page or the relevant tablet view (fig. 4.31).



Fig. 4.31 Screenshot of tablet 291 on the VTO2 website. Used with permission of the Centre for the Study of Ancient Documents, University of Oxford.

²² <http://www.huddletogogether.com/projects/lightbox2/>

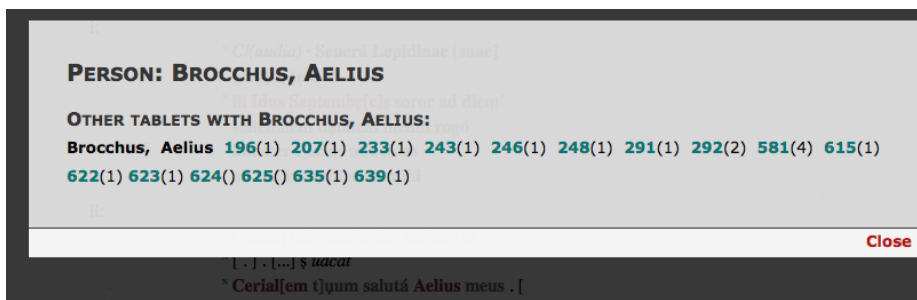


Fig. 4.32 Screenshot of the pop-up activated by clicking the name *Aelius* (line 9) in tablet 219 on the VTO2 website. Used with permission of the Centre for the Study of Ancient Documents, University of Oxford.

Next the transcription of the text is listed under the header, *Edition*. The words in the transcription are colour-coded to fit with the index in which they can be looked up. This is achieved by transforming the context of the different contextual encoding tags into HTML `<a>` tags with different classes. As an example the name *Aelius* (ii. 9) (fig. 4.30) was originally in a `<persname>` tag in the XML document retrieved via APPELLO. It is now in `<a>` tag with the attribute `class="purple"`. The name *Aelius* is furthermore a link that activates a pop-up listing all other occurrences of the name *Aelius* with a link to the tablet in which they occur (fig. 4.32).

The edition is followed by the translation, if there is any. The final part of this view is the commentary, a set of line-by-line notes. These notes can also be viewed in a pop-up view for each line through the *N* links in the left hand margin of the line (fig. 4.33).

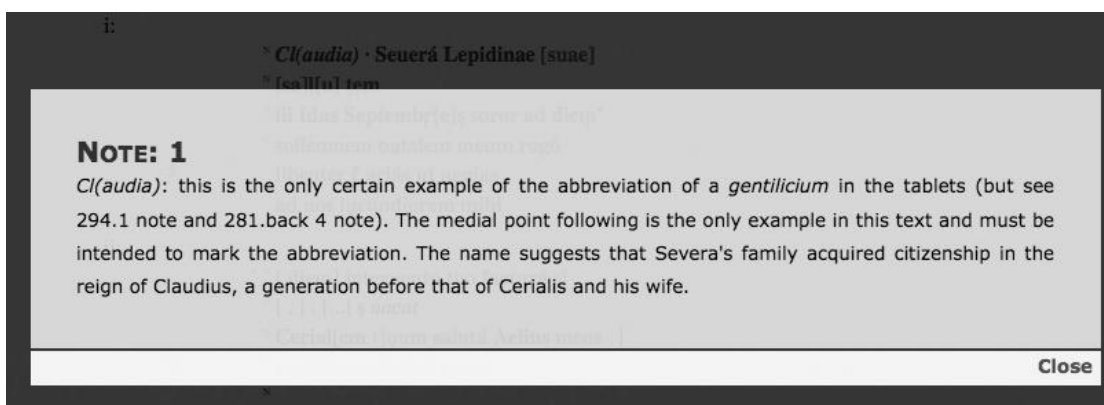


Fig. 4.33 Screenshot of the pop-up box showing note 1 in tablet 291 on the VTO2 website, activated by clicking the 'N' icon to the left of line 1. Used with permission of the Centre for the Study of Ancient Documents, University of Oxford.

4.5 SUMMARY

The purpose of Chapter 4 was to demonstrate how the DSS can be used in the context of other IT tools such as Web Services and XML. The main focus of this chapter is on the word search capabilities that are used in the DSS prototype to suggest possible words based on a pattern of a few letters. This search facility has been divided into lexical searches and concordances.

Lexical searches can be used by scholars to justify that a word is generally known and thus technically possible. Perseus Digital Library was used to illustrate such a search. Concordances on the other hand are used to show how a certain word can be placed in context. These are generally justified by referring to other published editions. The Vindolanda Tablets illustrated how such documents could be encoded with EpiDoc XML to contain Leiden Conventions and contextual information. Furthermore, I have demonstrated the development of the APPELLO Web Service word search of words throughout the entire XML dataset. In essence, APPELLO sends a URL with a method and possible pattern to the Web Service and returns the result of the search requested as TEI XML.

Then, the word suggestion engine developed for the DSS prototype was demonstrated using the Vindolanda Tablets APPELLO to find matching words for the characters input into the DSS prototype interface.

Finally, this chapter demonstrated how edition based datasets can be re-used with APPELLO to build interactive websites by making use of AJAX technology. This was illustrated first by presenting the previous VTO website and then exhibiting how the new VTO2 website has significantly improved user interaction with the Vindolanda Tablets.

5. CONCLUSION

This thesis has answered the research question; can the IT technology known as Decision Support Systems (DSS) be used as a tool to aid documentary scholars with the interpretation of ancient documents? The answer, derived from the achievement of the different aims of the thesis, is that DSS can indeed be a valuable tool in this context.

The first aim of the thesis was to understand the cognitive process of reading ancient documents and to examine how an interpretation is formed. This was accomplished in Chapter 2 through a literature review, which explored the history of reading ancient documents, from the first recorded attempts, in the first century CE, to read Egyptian pictographic texts and translate them into Greek, to modern day ancient document editing.

The methods, and more particularly the tools, have evolved between then and now and those working in the field of interpreting ancient documents have been keen to include current technology in the methodology.

Chapter 2 illustrated the difficulties involved in understanding the cognitive process of reading ancient documents, due to the inability of experts (i.e. the scholars), though engrossed in their tasks, to explain their work process, a phenomenon known as the 'paradox of expertise'. One scholar, Youtie, however, did take a great interest in the psychological aspects of reading and deciphering, forming for him part of what he called "private papyrology" (i.e. the actual reading of ancient documents). Terras recently carried this forward in her analysis, using different knowledge elicitation techniques, of the reading process of three scholars. She concluded that the scholarly reading process cycles through several reading levels, such as the discussion of features of character and the identification of sequences of characters. Terras furthermore concluded that each of the three scholars approached the reading in two different ways, described by Tarte as the

palaeographical/kinaesthetic and the philological/cruciverbalistic approach¹.

Furthermore, Terras concluded in particular that scholars moving between the different levels of reading operate within three of the reading levels: features of characters (level 1), identification of characters (level 2), and identification of possible words (level 4).

I used this perspective to develop the network of interpretations (fig 2.2) with the steps 0-3 ranging from low-level steps, such as "these three stroke fragments are a character," (step 0), to higher level steps, such as "these five letters can make up the word *primo*," (step 2). The network of interpretation was used to bridge the gap between previous work on understanding the reading of ancient documents and the development of the DSS prototype.

The second aim was to understand what requirements are needed of IT tools for the task of aiding interpretation in Humanities disciplines. In Chapter 2, I examined the current attitude towards IT in Humanities and concluded that the most important requirement for such IT tools is that they should be developed in close co-operation with the user-community and with the user-community's special requirements in mind. In light of this, I discussed the arguments for and against an automated IT tool for the reading of ancient documents. The conclusion was that such an automated system would not be appropriate for this particular application, as the reading of ancient documents is an act of interpretation, and interpretation is a purely human endeavour.

Nevertheless, I did identify certain aspects of reading ancient documents that can be significantly aided by IT technology. One of these, where the Decision Support System would be particularly useful, is the remembering of complex reasoning. Furthermore, IT

¹ See 2.1.1 for an explanation of these two terms.

technology could be very useful in the task of searching huge datasets, as proposed in Chapter 4.

The third aim was to test the concept of using a DSS in an interpretation scenario. In Chapter 3, I demonstrated my development of the DSS prototype and concluded that is a successful proof of concept. The use of argumentation in the DSS prototype is a valid and practical way of storing different decisions, along with arguments for and against these decisions.

The final aim was to understand how a DSS fits into the context of other services and systems present in Digital Humanities, particularly how the IT technology known as Web Services could provide valuable support to the DSS. Chapter 4 described the work I have done in regard to XML encoding of ancient document editions and my development of a Web Service that acts as a search engine for this dataset. Furthermore, in Chapter 4 I demonstrated how such a search engine could be used as a part of the DSS in order to, for example, suggest words to the reader during the editing of texts, whilst also being used in the publishing of ancient document editions online.

In the following section I will address the outcome of the thesis, illustrating how I tackled the subject and, further, how it relates to the wider study of handwritten texts.

I will discuss the choices I made during the course of my research, specifically the methodological choice to use argumentation for recording decisions, and the technical choice to use a MySQL database or EpiDoc encoded XML. Finally, I will look at how this research can benefit a wider community, whether using a DSS can change the way scholars work in the future, and if so, how this can be done.

Following this is a discussion of how a possible future DSS could be developed by employing a user interface that incorporates the concept of layers. My recommendations will cover methods of input, storage, exporting options, and editing the structure of a text

document. A significant disadvantage of the current prototype is the inability to incorporate the condition of the document, i.e. the incapability of adding the type of encoding to an edition that would traditionally be added using Leiden Conventions. I will discuss how this could be included in a future system, and illustrate a more user-friendly system for organising and extending argumentation.

This is followed by a demonstration of how a DSS can provide a wealth of data about the reading process, enabling subsequent analysis and an improved understanding of the work of scholars.

5.1 OUTCOME OF THE THESIS

This research set out to show how readers of ancient documents would find a DSS useful for aiding the interpretation process. It has established that a DSS can provide much needed help to scholars when attempting to recall their own complex reasoning. Chapter 3 successfully demonstrated this by developing the DSS prototype, using Vindolanda Tablets 670 and 862 as examples. Among other things, the prototype illustrates that the effectiveness of such a system lies in the successful development of a user-friendly interface. The LAIRAH² survey recommendations stress the importance of a well-designed user interface and liken the lack of one to ‘producing an expensive car but neglecting to provide a steering wheel, thus rendering it impossible to drive’ (Warwick *et al.* 2008: 100).

The survey concludes that, when the interface compares badly to the type of interfaces users have become accustomed to through their use of commercial resources, the digital resource is less likely to be continually used after development (Warwick *et al.* 2008).

² Log Analysis of Internet Resources in the Arts and Humanities
<http://www.ucl.ac.uk/infostudies/research/circa/lairah/>

One element that the prototype did not manage to demonstrate was an appropriate input interface. However, interaction with scholars during the development of the prototype made it very clear that a DSS would be of very little value to the community without such an interface, as it would hinder rather than help the reading process.

Whilst Terras focused on the entire reading process in her reading levels, I have instead looked closer at the levels where she identified the most activity during the reading process - levels 1, 2 and 4. The DSS prototype was developed to provide assistance primarily on these levels and for this reason I constructed the network of interpretation to illustrate how scholars can, for example, move between deciding the structure (i.e. lines, words and character) of a document and the meaning of a word.

The prototype DSS enabled scholars, at any point during the process, to record their interpretations, along with any arguments that may lie behind them. This very successfully demonstrated how argumentation can play a substantial role in the DSS. Arguments can be both **for** and **against** the interpretation in question, though scholars were not asked to note the emphasis they put on each argument. The ability to add arguments **for** and **against** each interpretation, whether at a lower or higher level, would be very advantageous for scholars. Furthermore, the use of argumentation in this way eliminates the need for quantifying or qualifying each interpretation in terms of its relative importance amongst other arguments.

The MDTSuite DSS for multidisciplinary teams on colorectal cancer inspired the DSS prototype. Scholars and medical practitioners are similar in that neither group can be persuaded to express the weight that they put on an argument in a way that can be calculated, and it is for this reason that in both cases using argumentation simply **for** and **against** is a better option.

However, scholars and medical practitioners are dissimilar in that, while the

decisions scholars make are recorded primarily for their own benefit, medical practitioners record their decisions for the use of other medical practitioners, and for legal reasons (Austin 2008).

As well as remembering complex reasoning when reading documents, I also aimed to demonstrate how to tackle difficult tasks that occur during the process, for example searching through huge datasets (see 2.4.2) and publishing the final editions (see 2.4.4).

Chapter 4 successfully demonstrated enhancements that can help make a DSS system even more powerful through the use of Web Services and XML. APPELLO is a word search Web Service that, based on parameters sent to it through a URL, can return a list of words and their context as reusable XML. The prototype DSS demonstrated how this can be used to offer suggestions for letters and words.

The Vindolanda encoding project and subsequent interactive website Vindolanda Tablets Online II (VTO2) illustrates the vast number of possibilities that arises from the ability to export editions in XML from the DSS. VTO2 is developed to provide the user with a more interactive experience and a sense of connectivity between different tablets. This is all accomplished with a combination of the APPELLO Web Service accessing the corpora's XML dataset and AJAX technology.

I am generally very happy with the technological choices I made for the DSS prototype, APPELLO and the Vindolanda tablets encoding. I chose to use a combination of PHP, JavaScript and AJAX for the interface and this has the further advantage that the code can be re-used by other open source developers in the future. Furthermore, although I believe that a more modern storage solution (e.g. RDF) would be advantageous for future DSS developments, I am satisfied that MySQL adequately served the purpose for the DSS prototype.

From the beginning, I decided to use XML encoding together with the TEI and

EpiDoc schemas and this proved successful. For example, APPELLO is developed for EpiDoc and can therefore, with only minor adjustments, be made to work with any EpiDoc based dataset, such as the Roman Tripolitania Project or MAMA XI.

The use of computers to aid interpretations in Humanities has formed a part of the discussion in this thesis. I have made it very clear that I do not suggest that Humanities is the environment for experimenting with developing systems that can make scholars obsolete by replicating the task of interpretation. The conclusion I have arrived at here is that a computer can never be designed to truly interpret anything of its own volition, but that it can be made to mimic the act of interpretation by providing it with a set of rules from which it can draw conclusions in a way that resembles interpretation.

On the other hand there are so many other tasks that scholars find laborious and time-consuming, and this thesis has shown that it is here that the use of IT can make a significant difference.

The introduction mentioned how a DSS can be developed to be of use to the wider field of interpreters of any written document. This thesis has focused on ancient documents, specifically those in Latin and Greek, but there is no reason why any of the developments resulting from this research should not be made to work with other document types, from Egyptian pictograms to 19th century letters. It would require adjustments, but the general principles would be the same. The dataset used to search for suggestions would be different and in some cases the structure of the document would also need to be altered, but it would still produce a network of interpretations at different stages and these could still be attached to arguments **for** and **against**, structured similarly to the DSS prototype. APPELLO itself could also be adjusted to work with a more general TEI schema instead of EpiDoc or even a completely different XML schema altogether.

It is probable that in the future scholars will increasingly take it for granted that

they will have IT available to aid their work. I think that there already exists diminishing patience with systems that require a change in working habits. I foresee that in the future scholars will want to spend less time in front of the computer filling out fields that they consider superfluous. I also think that there will be an expectation for all new published editions to be available and searchable online. This means that any developments in this field will need to focus on two vital elements:

The first of these vital elements is the development of easy, user-friendly interfaces that can remember scholars' past actions and will suggest the same actions in future interpretations. This system should involve minimal interaction so that scholars will not need to sit and click and type all the time.

The second vital need is for the system to act as an automated publishing system in a variety of standards. I am convinced that there will be a growing need to encode editions in the future, but there will also be need to be faster methods of doing this.

As I mentioned in the introduction, most work places have already embraced IT as an integral part of their working practice, and the study of ancient documents is no exception. Using a computer is now an essential part of research, and this has affected the way computers and IT solutions are viewed as work tools. The mystery has gone from them. Increasing numbers of scholars, like everyone else, expect IT tools to adapt to the way they work. This observation has been a vital driver of this thesis, and the DSS has been developed using research on scholars' actual working practice, as opposed to developing a system that scholars would have to adapt their working practices to use.

I do, however, acknowledge that this research is based on a very small group of scholars. Further development of a fully functional DSS will need to carry out a user study, focusing on how different groups of scholars work and how this compares to the examples in this thesis.

The remaining questions deal with how I envisage the role of IT for future scholars and how I think the study of ancient documents will look if the use of a DSS becomes commonplace.

Firstly, the approach taken by the eSAD team in the development of IT tools for the study of ancient documents is one that is already predominant in the field of Digital Humanities. In the future I would expect to see an increasing number of projects similar to the eSAD project, all starting with a study of the users for whom they wish to develop tools. This means that the future should see an increasing number of IT tools, probably open source, specifically developed for a target group of scholars.

In essence this is a very good thing. However, there could be a downside to this progress; there is a risk that it will mean many smaller projects, each working on its own system, many of which may actually overlap. Looking at the recent past, it is noticeable how many different systems have been or are being developed for some purpose of annotating images of documents (see 2.3.1). They each overlap on some of their functionality and differ on others. I am convinced that in the long run the limited resources of Humanities could be put to better use, through better collaboration between projects such as these.

This prompts me to describe an ideal future scenario in which there is widespread use of a DSS, developed through greater collaboration, using the most modern technology. The user interface is lightweight and when a scholar wishes to input data and record decisions this interface requires very little interaction. The output is compliant of all possible schemas and enables scholars to publish their edition in the format of choice with the click of a button.

All scholars using the DSS are able to interact with each other and co-operate and collaborate on different interpretation projects. Each scholar can be a member of a different

project group or can work alone. A scholar can work on many different projects at a time.

The DSS is a part of one Web application that that can be logged in to by scholars across the world. When beginning a project, scholars can choose between templates for different texts such as Latin texts, 18th Century letters or Nordic Runes. Depending on their own interests, scholars can be a member of different interest groups with which they can choose to share parts of the current interpretation. Once an edition is satisfactorily completed it can be categorised and finally published.

On publication, the edition can automatically become part of a Web Service feed for its particular category. Everyone else, subscribing to the feed or using it as a suggestion engine in his or her own interpretation project, now also has access to this published edition.

In this scenario the DSS is just part of a bigger interpretation management system, and as a result I expect the interpretation of documents, ancient or modern, to be a much more interactive pursuit. The way editions are published would be completely changed by this new system as all resources needed would be gathered in one accessible location through the aggregation of many different Web Service feeds.

Finally, as an additional benefit of all these combined developments, scholars are released from the mechanics of data entry, allowing more time for the actual reading of their documents - the focus of their own specific expertise.

The tools and prototypes developed throughout this thesis have shown how IT in general and a DSS in particular can help achieve this future ideal.

5.2 RECOMMENDATIONS FOR A FUTURE DSS IMPLEMENTATION

This part of the concluding chapter will illustrate how a future DSS could be developed. First it will describe a concept for an overall user system that can incorporate elements developed as a part of the eSAD project. This DSS will be layer-based and will enable a more intuitive interface for inputting the structure of the document. Furthermore, it will enable a smoother integration of external suggestion engines such as feature detection, character recognition and word search.

The development of the layer-based application has been the subject of some discussion in the context of the eSAD project. As a part of the project, different tools have been developed and redeveloped to aid the reading of damaged documents. The DSS prototype is one of these tools, working alongside tools to enhance photos of stylus tablets, in particular to remove woodgrain, and tools for letter recognition. As a result of close co-operation with the BVREH project we have been able to discuss the possibility of using, as a basis for this layer-based application, a Virtual Research Environment (VRE) similar to the one developed for the BVREH project.

The system would combine four types of layer (fig. 5.1). The first type is the ‘image layer’. This would primarily consist of the original image but could also include shadow stereo or PTM images of the document if they exist. Further, an image that has been enhanced by the code made available by the eSAD project could be included, such as an image with wood grain removed or illumination corrected (i.e. the enhanced layer).

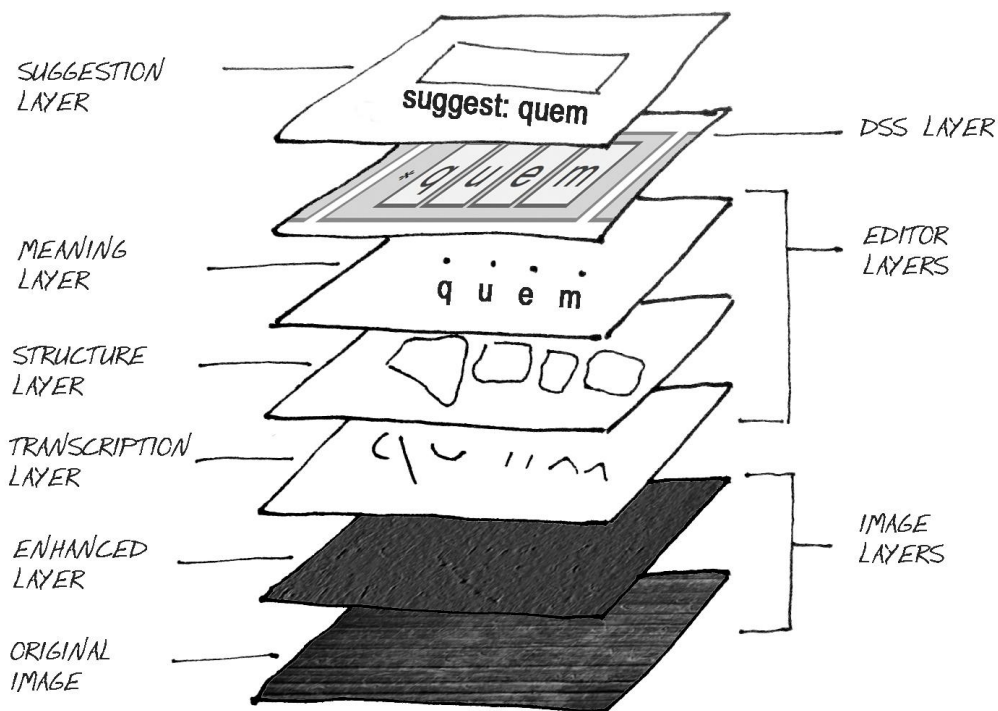


Fig 5.1 Drawing of the layer-based DSS concept. Images used with permission of the Centre for the Study of Ancient Documents, University of Oxford.

The second type of layer is the ‘editor layer’, so called because these layers can be edited by scholars. The ‘editor layer’ includes three sub-types: the ‘transcription layer’, the ‘structure layer’ and the ‘meaning layer’. The ‘transcription layer’ is one that would enable kinaesthetically minded scholars³ to include their own drawing of what they see in the document. The eSAD project has been experimenting with the use of an LCD drawing-tablet and pen for the enhancement of this task (Tarte 2011a: 240). Above this is the ‘structure layer’ in which scholars can draw polygons around lines, words and characters to define the structure of the document. Finally, there is the ‘meaning layer’ in which scholars can input points and attach letters to each point.

These three types of ‘editor layer’ will together be used to extract data to create a

³ See page 2.1.1 for a description of kinaesthetically minded scholars.

third type of layer - the ‘DSS layer’. The overall idea is that the layers are not static in any way. Scholars can go back and forth at any time and make changes to the ‘editor layers’, automatically updating the ‘DSS layer’. The ‘DSS layer’ will work in a similar way to the DSS prototype demonstrated in Chapter 3.

Finally, there is the ‘suggestion layer’. Using the ‘transcription layer’ the ‘character suggestion layer’ can retrieve information from character recognition software and display the most likely result in the character spaces. Working from the ‘meaning layer’ the ‘word suggestion layer’ can display the most likely words available. It is important to stress at this point, however, that these layers should never be viewed as more than mere suggestions intended to inspire scholars who might otherwise be unable to make progress with their reading. They should never be mistaken for automated readings in their own right.

The layers should also be more or less transparent, enabling the user to work on one layer while viewing another beneath it. For example, users could edit the transcription layer by arranging it above an enhanced image layer and tracing what they see on the layer below.

While I have been happy with the storage solution developed for the DSS prototype (i.e. MySQL database) it would be far from ideal for a larger DSS development. Austin (2008: 137) realised that in order to optimise the DSS developed for his thesis (MDTSuite), he needed to use a more flexible storage system. He chose to use a triple store/RDF system based on an ontology he developed specifically for this purpose.

This system stores triplets such as ‘`pizza hasTopping cheese`’, to use an example from the well-known Protégé tutorial demonstrating how to develop an ontology (Horridge *et al.* 2004). When implemented, this system is faster than the relational database used for the DSS prototype, primarily because the data is easier to extract. Furthermore, as RDF,

the dataset takes up less storage space because it is built without the redundancies that usually accompany a relational database.

Another storage possibility would be to encode all the information as XML and store it in an XML database such as eXist-db⁴.

The input method used in the prototype was a very basic version, consisting of input by clicking buttons. For a future system I envisage another two options; one is a drag and drop application where it would be possible to build the structure and add content to a document by dragging the elements into a sandbox; the other option is the use of image annotation to create interlinked polygons, each polygon representing a different structural element, to be filled out as the work proceeds.

I will use tablet 862 to demonstrate how each of these three methods would work. In line 4 of the tablet, S1 (Appendix B) first proposes the following reading:

Line 4: [S]VPER IDEM ACTAM AD VECTV

The final reading of this line is as follows (Appendix A):

Line 4: q̣cridem · ƒactam ad uetu-

Notice that while the last three words remain practically unchanged the first two words in S1's proposed reading (Appendix B) have become one word by the time of the final collaborative reading (Appendix A). Furthermore, in his first reading S1 did find that there was no letter before *ACTAM*, but somewhere during the process of interpretation a letter was added (i.e. ƒ), though with the under-dot symbol signifying that it should be read as unsure or indistinct. Any DSS must take into account the need to re-shuffle words and letters, or sometimes the entire structure of the document. By this I mean that the first two words (i.e. [S]VPER IDEM) need to be re-shuffled into one word (i.e. q̣cridem) and it must

⁴ <http://exist-db.org/>

be possible to add a letter (i.e. f) to the word ACTAM. In the following we will see how, with these three input methods, this becomes possible.

The first method consists simply of the clicking of buttons (fig. 5.2). This would be a perfectly viable, if not slightly tedious from a user's point of view, solution to inputting the structure of a document. It would require scholars to click a button each time they wished to add a character box or a new line.

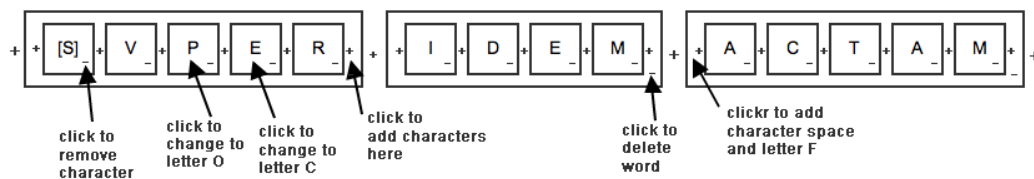


Fig. 5.2 Drawing of how the 'button clicking' input method could work in line 4, tablet 862

Using the above example, S1 would initially click the <+> buttons to add the number of words and letters that he needs for his interpretation. If he then wishes to change the structure and make the first word longer and remove the second word altogether he could add more letters to the first word and delete the second word using the <-> button.

The second method of input is one where the main view of the document would resemble a sandbox and where structural elements could be dragged and dropped and shuffled around at will (fig. 5.3). Other elements of the interpretation process could also be dragged and dropped on top of the structure to relate them to each other.

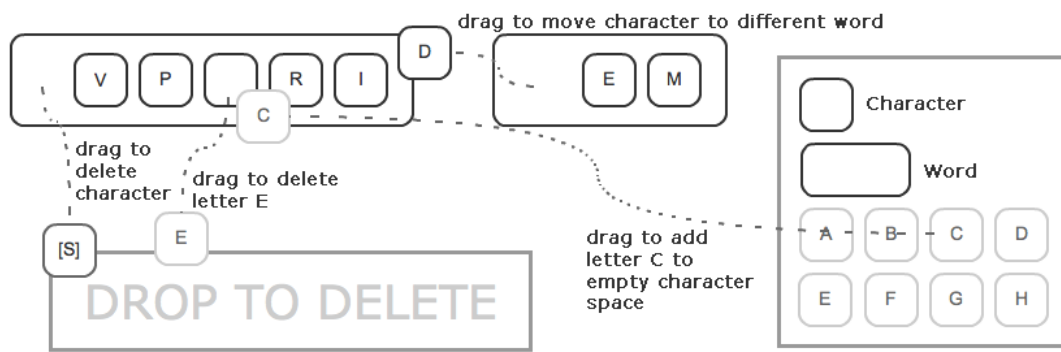


Fig. 5.3 Drawing of how the 'drag and drop' input method could work in line 4, tablet 862

Using the example here, S1 would drag two word elements from the element box over to the line 4 element. He would then drag five character elements over to the first word element and four character elements over to the second word element. This would then be populated with letters by dragging them from the element box over to the intended character elements.

When the reading changes the scholars can then drag the character elements from the second word directly over to the first word. To get rid of the second word, the word element would be dragged out and dropped at the bottom where it would vanish. Some of the letters in S1's version of the first word would also need to be changed; this could be achieved by dragging the letter E to the bottom and dragging a new letter C into the same character element.

The third and last method is one of image annotation (fig. 5.4). In this scenario scholars would draw polygons directly onto the image to indicate the structure of the document. These polygons could then be adjusted and moved around depending on the changes in interpretation. Furthermore, other elements of the interpretation process, such as the arguments for and against an interpretation could be linked to these polygons.

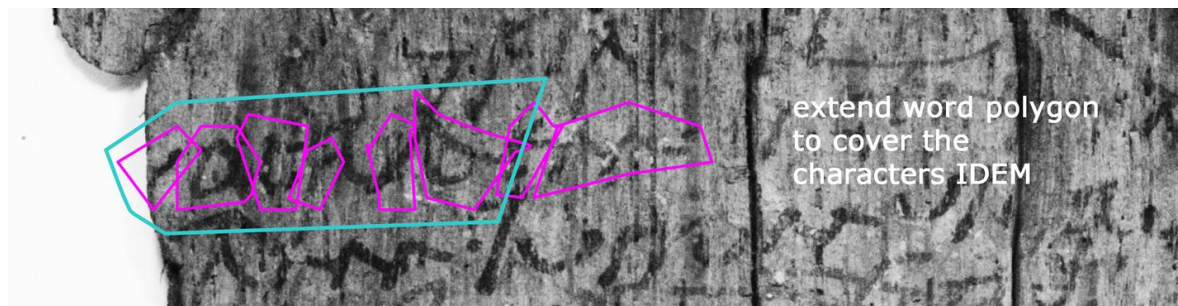


Fig. 5.4 Drawing of how the 'image annotation' input method could work in line 4, tablet 862. 3. Image of tablet used with permission of the Centre for the Study of Ancient Documents, University of Oxford.

Using the example of line 4 again, S1 would begin by drawing a polygon around all the strokes he thought were a part of line 4. He would then draw another polygon around what he thought made up the first word and another around the second word. Within these words S1 would draw polygons around the characters and label these polygons with the letters and other relevant information.

When this reading changes, the scholars could edit each polygon by adjusting the polygon sides. For example, the polygon for the second word would be deleted leaving four character polygons without a word polygon surrounding them. Then the first word polygon could be extended to cover these four extra character polygons. The first character polygon in the first word would be deleted too and the labels for the current first, second and third character would be edited to reflect their changed reading.

For use in the future layer-based system I recommend a combination of the last two of these three options. The image annotation option would work very well in a 'structure layer', while the drag and drop option would be ideal for the DSS and meaning layers. This combination would create a very intuitive system, requiring minimal interaction for the user.

As mentioned previously, I would expect any future system to enable output as EpiDoc or TEI XML, vital because online publication is the mode of the future. This

means that, beyond the need to remember complex reasoning, there is as great a need to export any such interpretation process as a functional edition ready to be published online.

Chapter 4 closely examined the encoding of the state of a document through the previous use of Leiden symbols and brackets, and now through EpiDoc. It has not previously been deemed necessary to add this functionality to the DSS prototype in order to illustrate the usefulness of a DSS to the reading of ancient documents. However, any development of a system to aid the reading of ancient documents should incorporate this functionality in order that the text can be exported and published, ideally using the Leiden

Conventions because this is the format with which users are most familiar. Using EpiDoc XML to encode is one method that would ensure this capability. For example, if there is a gap in the text this should be stored in the DSS in such a way that it can be exported as EpiDoc XML: `<gap/>`. Using a XSLT stylesheet, this can then be transformed for publication as `[ ]`, which is the standard Leiden Convention for indicating a gap in the text.

Ideally this export of a provisional interpretation can then be attached directly to a Web Service from which it can be re-used to search datasets in future readings. This brings the reading process to a full circle.

Another part of the DSS that would need work in order to be fully functional is the organisation of a user interface for the argumentation. I will use the concept of the drag and drop system as an example to show how this could work. Using this system, arguments **for** and **against** a letter are amongst the elements that could be dragged and dropped on top of the letter combinations for which the argument was used.

However, this would first require the letters to be dragged and dropped onto each character element. In addition to this capability it would be ideal if the letters connected to each character element and their level of resolve was visible from the main DSS view.

Currently, the letters and their resolve are indicated with a colour coding system⁵.

5.3 UNDERSTANDING THE INTERPRETATION PROCESS WITH DSS

Up to this point I have focused on how to develop a DSS and how an understanding of the interpretation process is necessary in order to build a system that can be a useful aid in this process. However, I will now argue that once such a system has been developed and put into use, it can provide a valuable repository of information on the cognitive process of reading ancient documents, the construction of interpretation, and the meaning of a document. If developed as suggested above, the DSS will collate the arguments used by scholars to remind/convince themselves and each other why they read a piece of text in a certain way.

This can act in a similar fashion to the TAP exercise conducted by Terras (2002) as a part of her thesis. However, since it is not the intention of the DSS to force scholars to record every argument, it could be argued that, in comparison to a TAP exercise, the DSS will provide a rather incomplete record of the interpretation process. On the other hand, scholars may be more likely to record the complex reasoning that could be forgotten between one reading session and the next, or they might record enough to emphasise the value of their argument to the scholars with whom they may be collaborating. Whilst this leaves out the argumentation that seems obvious to the scholars themselves or that they deem will be obvious to their colleagues, it will provide a valuable insight into how the interpretation process was built up, particularly if each argument is marked with its author and a time stamp, enabling the extraction of a timeline of argumentation and interpretation.

With such a dataset it would be possible to conduct a whole new type of analysis on how interpreting ancient documents is done on a much wider scale.

⁵ See 3.4 for more information about how the letters are tackled in the DSS prototype.

BIBLIOGRAPHY

- Abeyesinghe, S. (2008) *RESTful PHP Web Services*. Birmingham: PACKT Publishing.
- Adams, J. N. (1995) The Language of the Vindolanda Writing Tablets: An Interim Report. *The Journal of Roman Studies*, 85, 86-134.
- Ainsworth, P. and Meredith, M. (2009) e-Science for Medievalists: Options, Challenges, Solutions and Opportunities. *Digital Humanities Quarterly*, 3 (4).
- APIS Website (2011) *About the APIS Project*.
<http://www.columbia.edu/cu/lweb/projects/digital/apis/about.html> [09.11.2011]
- Austin, M. (2008) *Information Integration and Decision Support for Multidisciplinary Team Meetings on Colorectal Cancer*, D.Phil Thesis, Department of Engineering Science, University of Oxford.
- Austin, M., Kelly, M. and Brady, M. (2008) The Benefit of an Ontological Patient Model in Clinical Decision Support. IN: *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*. Menlo Park: AAAI Press, 1872-1872.
- Bagnall, R. S. (1995) *Reading Papyri, Writing Ancient History*. London: Routledge.
- Bagnall, R. S. (1997) Imaging of Papyri: A Strategic Overview. *Literary and Linguistic Computing*, 12 (3), 153-154.
- Bagnall, R. S. (2009) *The Oxford Handbook of Papyrology*. Oxford: Oxford University Press.
- Barrett, A. (2005) The Information-Seeking Habits of graduate Student Researchers in the Humanities. *The Journal of Academic Librarianship*, 3 (4), 324-331.
- Bearman, G. H. and Spiro, S. I. (1996) Archaeological Applications of Advanced imaging Techniques. *The Biblical Archaeologist*, 59 (1), 56-66.
- Bell, M. Z. (1985) Why Expert systems Fail. *The Journal of the Operational Research Society*, 36 (7), 613-619.
- Bell, B. (2007) *Minimus, Starting out in Latin*. Cambridge: Cambridge University Press.
- Benton, J., Gillespie, A. and Soha, J. (1979) Digital Image-Processing Applied to the Photography of Manuscripts. *Scriptorium*, 33, 40-55.
- Biber, D., Conrad, S. and Reppen, R. (1994) Corpus-based Approaches to Issues in Applied Linguistics. *Applied Linguistics*, 15 (2), 169-189.
- Bidez, J. and Drachmann, A. B. (1932) *Emploi des signes critiques, disposition de l'apparat dans les éditions savantes de textes grecs et latins, conseils et recommandations*. Paris: É. Champion for Union académique internationale.
- Bidwell, P. (1997) *Roman Forts in Britain*. London: English Heritage.

- Birley, R. (1997) The Vindolanda Bonfire. *Current Archaeology*, 13, 348-357.
- Bischoff, B. (1990) *Latin Palaeography: Antiquity and the Middle Ages*, translated by D. O. Croinin and D. Ganz. Cambridge: Cambridge University Press.
- Bodard, G. (2008) The Inscriptions of Aphrodisias as Electronic Publication: A User's Perspective and a Proposed Paradigm. *Digital Medievalist*, 4.
<http://www.digitalmedievalist.org/journal/4/bodard/> [09.11.2011]
- Bodard, G. and Mahoney, S. (eds.) (2010) *Digital Research in the Study of Classical Antiquity*. London: Ashgate.
- Boden, M. A. (1990) The Social Impact of Artificial Intelligence. IN: Kurzweil, R. (ed.) *The Age of Intelligent Machines*. Cambridge, Mass: M.I.T. Press, 450-453.
- Bonzio, R. (2011) Father Busa, Pioneer of Computing in Humanities with Index Thomisticus, dies at 98. *Forbes*, 11 August 2011.
<http://www.forbes.com/sites/robertobonzio/2011/08/11/father-busa-pioneer-of-computing-in-humanities-dies-at-98/> [09.11.2011]
- Boose, J. H. (1986) *Expertise Transfer for Expert System Design*. New York:Elsevier.
- Borgman, C. (2009) The Digital Future is Now: A Call to Action for Humanities. *Digital Humanities Quarterly*, 3 (4).
- Bowman, A. K. (1994) *Life and Letters on the Roman Frontier: Vindolanda and its People*. New York: Routledge.
- Bowman, A. K. (2002) Recolonising Egypt. IN: Wiseman, T. P. (ed.) *Classics in progress – Essays on ancient Greece and Rome*. Oxford: Oxford University Press, 193-223.
- Bowman, A. K., Brady, M. and Tomlin, R. S. O. (1997) Imaging Incised Documents. *Literary and Linguistic Computing*, 12 (3), 169-176.
- Bowman, A. K. and Brady, M. (2005) *Images and Artefacts of the Ancient World*. London: British Academy.
- Bowman, A. K., Crowther, C. V., Kirkham, R. and Pybus, J. (2010b) A Virtual Research Environment for the Study of Ancient Documents and Manuscripts. IN: Bodard, G. and Mahoney, S. (eds.) *Digital Research in the Study of Classical Antiquity*. London: Ashgate, 87-103.
- Bowman, A. K. and Thomas, J. D. (1983) *Vindolanda: The Latin Writing Tablets, Tab.Vindol.I*. London: Society for the Promotion of Roman Studies.
- Bowman, A. K. and Thomas, J. D. (1994) *The Vindolanda writing-tablets : (Tabulae Vindolandenses II), Tab.Vindol.II*. London: British Museum Press.
- Bowman, A. K. and Thomas, J. D. (2003) *The Vindolanda writing-tablets (Tabulae Vindolandenses III), Tab.Vindol.III*. London: British Museum Press.

- Bowman, A. K., Thomas, J. D. and Tomlin, R. S. O. (2010a) The Vindolanda Writing-Tablets (Tabulae Vindolandenses IV, Part 1), Tab.Vindol.IV. *Britannia*, 41, 187-224.
- Bowman, A. K., Tomlin, R. S. O., and Worp, K. A. (2009) Emptio bovis Frisica: the 'Frisian ox sale' reconsidered. *The Journal of Roman Studies*, 99, 156-170.
- Brown, M. P. (1998) *Writing and Scripts: History and Techniques*. London: British Library.
- Boyle, L. E. (2001) *Integral Palaeography*. Turnhout: Brepols.
- Brady, M., Pan, X., Schenk, V. U. B., Terras, M., Robertson, P., and Molton, N. (2005) Shadow stereo, image filtering and constraint propagation IN: Bowman, A. K. and Brady, M. (eds.) *Images and Artefacts of the Ancient World*. London: British Academy, 15-30.
- British Museum Website (2011) *Our Top Ten British Treasures*.
http://www.britishmuseum.org/explore/online_tours/britain/our_top_ten_british_treasures/the_vindolanda_tablets.aspx [09.11.2011]
- Busa, R. (1980) The Annals of Humanities Computing: The Index Thomisticus. *Computers and the Humanities*, 14, 83-90.
- Cayless, H. (2008) *Linking Page Images to Transcriptions with SVG*.
<http://www.unc.edu/~hcayless/img2xml/presentation.html> [09.11.2011]
- Cayless, H., Roueché, C., Elliott, T., and Bodard, G. (2009) Epigraphy in 2017. *Digital Humanities Quarterly*, 3 (1). <http://digitalhumanities.org/dhq/vol/3/1/000030/000030.html> [09.11.2011]
- Chabries, D. M., Booras, S. W. and Bearman, G. H. (2003) Imaging the Past: Recent Applications of Multispectral Imaging Technology to Deciphering Manuscripts. *Antiquity*, 77 (296), 359-372.
- Ciula, A. (2005) Digital Palaeography: Using the Digital Representation of Medieval Script to Support Palaeographic Analysis. *Digital Medievalist*, 1.
<http://www.digitalmedievalist.org/journal/1.1/ciula/> [09.11.2011]
- Ciula, A. (2009) The Palaeographical Method Under the Light of a Digital Approach. IN: Rehbein, M., Sahle, P. and Schaßan, T. *Codicology and Palaeography in the Digital Age*. Norderstedt: Books on Demand, 219-235.
- Coats, P. K. (1988) Why Expert Systems Fail. *Financial Management*, 17 (3), 77-86.
- Crane, G. (2004) Classics and the Computer: An End of the History. IN Schreibman, S., Siemens, R. and Unsworth, J. (eds.) *A Companion to Digital Humanities*. Oxford: Blackwell, 46-55.
- David, M. and van Groningen, B. A. (1952) *Papyrological Primer*. Leyden: E. J. Brill.
- Davis, T. (2007) The Practice of Handwriting Identification. *The Library*, 8 (3), 251-276.

- Deegan, M. and Sutherland K. (2009) *Transferred Illusions: Digital Technology and the Forms of Print*. London: Ashgate.
- de La Flor, G., Luff, P., Jirotko, M., Pybus, J., Kirkham, R. and Carusi, A. (2009) The case of the disappearing ox: seeing through digital images to an analysis of ancient texts. IN: *CHI '10 Proceedings of the 28th international conference on Human factors in computing systems*. New York: ACM.
- Digital Humanities Manifesto 2.0 (2009) UCLA.
http://www.humanitiesblast.com/manifesto/Manifesto_V2.pdf [09.11.2011]
- Dow, S. (1969) *Conventions in Editing: A Suggested Reformulation of the Leiden System*. Duke University.
- Earl, G., Basford, P., Bischoff, A., Bowman, A., Crowther, C., Dahl, J., Hodgson, M., Isaksen, L., Kotoula, E., Martinez, K., Pagi, H. and Piquette, K. E. (2011) *Reflectance Transformation Imaging Systems for Ancient Documentary Artifacts*.
http://ewic.bcs.org/upload/pdf/ewic_ev11_s8paper3.pdf [09.11.2011]
- Elliott, T. (2008) *EpiDoc Website: About EpiDoc*. <http://epidoc.sourceforge.net> [09.11.2011]
- Elliott, T., Bodard, G., Crowther, C. (2007) *Transcription wherein the character would not be unambiguous outside its context*. <http://www.stoa.org/epidoc/gl/5/unclear.html> [09.11.2011]
- Eom, S. B., Lee, S. M., Kim, E. B. and Somarajan, C. (1998) A Survey of Decision Support System Applications (1988-1994). *Journal of the Operational Research Society*, 49, 109-120.
- Eysenk, M. W. and Keane, M. T. (2010) *Cognitive Psychology: A Student Handbook*, Sixth Edition. New York: Psychology Press.
- Feigenbaum, E. A. and McCorduck, P. (1983) *The Fifth Generation: Artificial Intelligence and Japan's Computer Challenge to the World*. London: Michael Joseph.
- Fielding, R. T. (2000) *Architectural Styles and the Design of Network-based Software Architectures*. D.Phil Thesis, Information and Computer Science, University of California.
<http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm> [09.11.2011]
- Flanders, J., Pies, W. and Terras, M. (2007) Welcome to Digital Humanities Quarterly. *Digital Humanities Quarterly*, 1 (1).
<http://digitalhumanities.org/dhq/vol/1/1/000007/000007.html#> [09.11.2011]
- Fox, J., Krause, P., and Elvang-Gøransson, M. (1996) Argumentation as a General Framework for Uncertain Reasoning. IN: Heckerman, D. and Mamdani, E. H. (eds.) *UAI '93: Proceedings of the Ninth Annual Conference on Uncertainty in Artificial Intelligence, July 9-11, 1993, The Catholic University of America, USA*. San Fransisco: Morgan Kaufmann Publishers, 428-434.

- Fulford, M. G., O'Riordan, E. J., Clarke, A. and Rains, M. (2010) Silchester Roman Town: Developing Virtual Research Practice 1997-2008. IN: Bodard, G. and Mahoney, S. (eds.) *Digital Research in the Study of Classical Antiquity*. London: Ashgate, 15-34.
- Gaceb, D., Eglin, V., Lebourgeois, F. and Emptoz, H. (2008) Improvement of Postal Mail Sorting System. *International Journal of Document Analysis and Recognition (IJDAR)*, 11 (2), 67-80.
- Galey, A. and Ruecker, S. (2010) How a prototype argues. *Literary and Linguistic Computing*, 25 (4), 405-424.
- Gallo, I. (1986) *Greek and Latin Papyrology*. London: Institute of Classical Studies.
- Gareth, J. J. (2005) *Ajax: A New Approach to Web Applications*.
<http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications>
 [09.11.2011]
- Gonzales, R. C. and Woods, R. E. (2008) *Digital Image Processing*. New Jersey: Pearson Prentice Hall.
- Gordon, A. E. (1983) *Illustrated Introduction to Latin Epigraphy*. Berkeley: University of California Press.
- Gordon, C. H. (1958) Minoan Linear A. *Journal of Near Eastern Studies*, 17 (4), 245-255.
- Gorry, G. A. and Morton, M. S. S. (1971) A Framework for Management Information Systems. *Sloan Management Review*, 13 (1), 55-70.
- Goskar, T. A. and Earl, G. (2010) Polynomial Texture Mapping for Archaeologists, *British Archaeology*, 111, 28-31.
- Gregory, R. L. (1998) *Eye and Brain: The Psychology of Seeing*. Oxford: Oxford University Press.
- Grenfell, B. P. and Hunt, A. S. (1898) *The Oxyrhynchus Papyri: Part I*. London: Egypt Exploration Fund.
- Guichard, M. (2002) The Precursors of the Written Word in the Danube River Basin. IN: Christin, A. M. (ed.) *A History of Writing : From Hieroglyph to Multimedia*. Paris: Flammarion, 17-19.
- Haas, H. and Brown, A. (2004) *Web Services Glossary*. <http://www.w3.org/TR/ws-gloss/>
 [09.11.2011]
- Hall, W., De Roure, D., and Shadbolt, N. (2009) The evolution of the Web and implications for eResearch. *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, 367 (1890), 991-1001.
- Harmond, P. and King, D. (1985) *Expert Systems*. New York: John Wiley & Sons
- Hart, A. (1986) *Knowledge Acquisition for Expert Systems*. London: Kogan Page.

- Hippisley, D. (2005) *Encoding the Vindolanda Tablets: An Investigation in Contextual Encoding using XML and the EpiDoc Standards*, Masters in Electronic Communication and Publishing, University College London.
- Hockey, S. (2004) The History of Humanities Computing. IN Schreibman, S., Siemens, R. and Unsworth, J. (eds.) *A Companion to Digital Humanities*. Oxford: Blackwell, 3-19.
- Hondius, J. J. E. (1923) *Supplementum Epigraphicum Graecum*, Vol. 1. Amsterdam: Lugduni Batavorum.
- Hondius, J. J. E. (1929) *Supplementum Epigraphicum Graecum*, Vol. 4. Amsterdam: Lugduni Batavorum.
- Hondius, J. J. E. (1934) *Supplementum Epigraphicum Graecum*, Vol. 7. Amsterdam: Lugduni Batavorum.
- Horridge, M., Knublauch, H., Rector, A., Stevens, R., and Wroe, C. (2004) *A Practical Guide to Building Ontologies using the Protégé-OWL Plugin and CO-ODE Tools Edition 1.0*. Paper by The University of Manchester.
http://owl.cs.manchester.ac.uk/tutorials/protegeowltutorial/resources/ProtegeOWLTutorialP4_v1_1.pdf [09.11.2011]
- Hunt, A. S. (1932) A note on the transliteration of papyri, *Chronique d'Égypte*, 7, 272-274.
- IDP Wiki (2011) *Integrating Digital Papyrology: Duke Databank of Documentary Papyri (DDBDP)*. <http://idp.atlantides.org/trac/idp/wiki/DDBDP> [09.11.2011]
- Ibeji, M. (2001) *BBC – History – Ancient History in depth: Vindolanda*.
http://bbc.co.uk/history/ancient/romans/vindolanda_01.shtml [09.11.2011]
- Jannidis, F. (2009) TEI in a crystal ball. *Literary and Linguistics Computing*, 24 (3), 253-265.
- Johnson, P. E. (1983) What kind of expert should a system be?. *The Journal of Medicine and Philosophy*, 8 (1), 77-97.
- Jonscher, C. (2000) *Wired Life: Who Are We in the Digital Age?* London: Anchor.
- Juola, P. (2008) Killer Applications in Digital Humanities. *Literary and Linguistics Computing*, 23 (1), 73-83.
- Keen, P. G. W. and Morton, M. S. S. (1978) *Decision support systems : an organizational perspective*. London: Addison-Wesley.
- Keenan, J.G. (2009) The History of the Discipline. IN: Bagnall, R. S. *The Oxford Handbook of Papyrology*. Oxford: Oxford University Press: 59-78.
- Klein, M. R. and Methlie, L. B. (1995) *Decision Support Systems: With Applications in Business*. New York: John Wiley & Sons.
- Kurzweil, R. (1990) *The Age of Intelligent Machines*. Cambridge, Mass: M.I.T. Press

- Lock, G. (2003) *Using Computers in Archaeology: Towards Virtual Pasts*. London: Routledge.
- Mahoney, A. (2006) Epigraphy. IN: Burnard, L., O'Brien O'Keeffe, K. and Unsworth, J. *Electronic Textual Editing*. http://www.tei-c.org/About/Archive_new/ETE/Preview/mahoney.xml [09.11.2011]
- Manguel, A. (1997) *A History of Reading*. London: Flamingo
- Mann, M. (1949) Reading Machine Spells Out Loud. *Popular Science*, February 1949, 154 (2), 125-127.
- McClelland, J. L. and Rumelhart, D. E. (1981) An Interactive Activation Model of Context Effects in Letter Perception: Part 1. An Account of Basic Findings. *Psychological Review*, 88 (5), 375-407.
- McDonough, J. (2009) XML, Interoperability and the Social Construction of Markup Languages: The Library Example. *Digital Humanities Quarterly*, 3 (3).
- McGraw, K. L. and Harbison-Briggs, K. (1989) *Knowledge Acquisition: Principles and Guidelines*. New Jersey: Prentice-Hall
- Molton, N., Pan, X., Brady, M., Bowman, A. K., Crowther, C., and Tomlin, R. (2003) Visual enhancement of incised text. *Pattern Recognition*, 36, 1031-1043.
- Moran, M. (2009) *Ontology for Ancient Documents*. Engineering Science Fourth Year Project, University of Oxford.
- Nagy, G. (1965) Observations on the Sign-Grouping and Vocabulary of Linear A. *American Journal of Archaeology*, 69 (4), 295-330.
- Oakhill, J. and Garnham, A. (1988) *Becoming a Skilled Reader*. Oxford: Basil Blackwell.
- O'Donnell, D. P. (2009) Back to the future: what digital editors can learn from print editorial practice. *Literary and Linguistics Computing*, 24 (1), 113-125.
- Pan, X. B., Brady, M., Bowman, A. K., Crowther, C., and Tomlin, R. S. O. (2004) Enhancement and feature extraction for images of incised and ink texts. *Image and Vision Computing*, 22, 443-451.
- Pasquali, G. (1936) *In memoria di Girolamo Vitelli*. Firenze: F. Le Monnier.
- Perseus Website (2011) *About the Perseus Digital Library*. <http://www.perseus.tufts.edu/hopper/about> [09.11.2011]
- Pestman, P. W. (1990) *The New Papyrological Primer*. Leiden: E. J. Brill.
- Piez, W. (2005) SVG Visualization of TEI Texts. *ACH/ALLC Conference 2005*. <http://pear.hcmc.uvic.ca:8080/ach/site/xhtml.xq?id=214> [09.11.2011]
- Pitts, M. (2009a) Staffordshire Gold. *British Archaeology*, Nov/Dec 2009, 14-21.

- Pitts, M. (2009b) Anglo-Saxon gold hoard is the biggest – and could get bigger. *Guardian* 25. September 2009, Main section, page 6.
<http://www.guardian.co.uk/uk/2009/sep/24/staffordshire-anglo-saxon-gold-find>
 [09.11.2011]
- Porter, D. C., Resides, D. and Walsh, J. (2009) Text-Image Linking Environment (TILE). *Digital Humanities 2009 Conference Abstracts*, 388-390.
http://www.mith2.umd.edu/dh09/wp-content/uploads/dh09_conferencepreceedings_final.pdf [09.11.2011]
- Pritchett, W. K. (1955) Dotted Letters in Greek Epigraphy. *American Journal of Archaeology*, 59 (1), 55-61.
- Rendsburg, G. A. (1996) “Someone will succeed in Deciphering Minoan”: Cyrus H. Gordon and Minoan Linear A. *The Biblical Archaeologist*, 59 (1), 36-43.
- Reynolds, J., Roueché, C. and Bodard, G. (2007) *Inscriptions of Aphrodisias, IAPH2007*.
<http://insaph.kcl.ac.uk/iaph2007> [09.11.2011]
- Richardson, L. and Ruby, S. (2007) *RESTful Web Services*. Cambridge: O'Reilly
- Robertson, P. (2001) *A Self-Adaptive Architecture for Image Understanding*. D.Phil Thesis, Department of Engineering Science, University of Oxford.
- Roueché, C. (2004) *Aphrodisias in Late Antiquity: The Late Roman and Byzantine Inscriptions, revised second edition, ala2004*. <http://insaph.kcl.ac.uk/ala2004> [09.11.2011]
- Roueché, C. and Flanders, J. (2006) *Introduction for Epigraphers*.
<http://epidoc.sourceforge.net/IntroEpigraphers.shtml> [09.11.2011]
- Roued Olsen, H. (2007) *Heritage Portals and Cross-border Data Interoperability*. MSc in Archaeological Computing, University of Southampton.
- Roued, H. (2009) Textual Analysis using XML: Understanding Ancient Textual Corpora. Paper for the 5th IEEE International Conference on e-Science, 9 –11 December 2009, Oxford, UK.
http://esad.classics.ox.ac.uk/index.php?option=com_docman&task=doc_details&gid=30&Itemid=78 [09.11.2011]
- Roued-Cunliffe, H. (2010) Towards an Interpretation Support System for Reading Ancient Documents. *Literary and Linguistics Computing*, 25 (4), 365-379.
- Russell, S. and Norvik, P. (2010) *Artificial Intelligence: A modern approach*. Boston: Pearson.
- Sauter, V. (1997) *Decision support systems: an applied managerial approach*. New York: John Wiley and Sons.
- Schenk, V. U. B. (2001) *Visual Identification of Fine Surface Incisions*. D.Phil Thesis, Department of Engineering Science, University of Oxford.

- Schenk, V. U. B. and Brady, M. (2003) Visual Identification of Fine Surface Incisions in Incised Roman Stylus Tablets. IN: Mukherjee, D. P. and Pal, S. *Proceedings of the Fifth International Conference - Advances in Pattern Recognition ICAPR 2003*. Mumbai: Allied Publishers Pvt. Ltd
- Schmidt, D. (2010) The inadequacy of embedded markup for cultural heritage texts. *Literary and Linguistics Computing*, 25 (3), 337-356.
- Schomaker, L. (2007) Advances in Writer Identification and Verification. *Ninth International Conference on Document Analysis and Recognition 2007*.
- Schreibman, S., Siemens, R. and Unsworth, J. (eds.) (2004) *A Companion to Digital Humanities*. Oxford: Blackwell.
- Schubert, P. (2009) Editing a Papyrus. IN: Bagnall, R. S. *The Oxford Handbook of Papyrology*. Oxford: Oxford University Press: 197-215.
- Siemens, R., Leitch, C., Blake, A., Armstrong, K. and Willinsky, J. (2009) "It May Change my Understanding of the Field": Understanding Reading Tools for Scholars and Professional Readers. *Digital Humanities Quarterly*, 3 (4).
- Sol, H. G., Takkenberg, C. A. T., de Vries Robbé, P. (1987) Introduction. IN: Sol, H. G. *Expert systems and artificial intelligence in decision support systems. Proceedings of the Second Mini Euroconference, Lunteren, The Netherlands, 17-20 November 1985*. Dordrecht: D. Reidel, 1-8.
- Smith, A. H. (1938) The Photography of Manuscripts. *London Mediaeval Studies*. 1 (2), 179 – 201.
- Smith, F. (2004) *Understanding Reading: A Psycholinguistic Analysis of Reading and Learning to Read*. London: Lawrence Erlbaum Associates.
- Sprague, R. H. and Carlson, E. D. (1982) *Building effective decision support systems*. N.J.: Prentice-Hall.
- Sprague, R. H. and Watson, H. J. (1989) *Decision support systems: putting theory into practice*. London: Prentice Hall International.
- Stokes, P. A. (2007) Palaeography and Image-Processing: Some Solutions and Problems. *Digital Medievalist*, 3. <http://www.digitalmedievalist.org/journal/3/stokes/> [09.11.2011]
- Stokes, P. A. (2009) Computer-Aided Palaeography, Present and Future. IN: Rehbein, M., Sahle, P. and Schaßan, T. *Codicology and Palaeography in the Digital Age*. Norderstedt: Books on Demand, 313-342.
- Srihari, S. N. (1993) Recognition of Handwriting and Machine-printed Text for Postal Address Interpretation. *Pattern Recognition Letters*, 14, 291-302.
- Srihari, S., Huang, C. and Srinivasan, H. (2008) On the Discriminability of the Handwriting of Twins. *Journal of Forensic Science*, 53 (2), 430-446.

- Svensson, P. (2009) Humanities Computing as Digital Humanities. *Digital Humanities Quarterly*, 3 (3). <http://digitalhumanities.org/dhq/vol/3/3/000065/000065.html> [09.11.2011]
- Svensson, P. (2010) The Landscape of Digital Humanities. *Digital Humanities Quarterly*, 4 (1). <http://digitalhumanities.org/dhq/vol/4/1/000080/000080.html> [09.11.2011]
- Tarte, S. (2011a) Papyrological Investigations: Transferring Perception and Interpretation into the Digital World. *Literary and Linguistics Computing*, 26 (2), 233-247.
- Tarte, S. (2011b) Digitizing the Act of Papyrological Interpretation: Negotiating Spurious Exactitude and Genuine Uncertainty. *Literary and Linguistics Computing*, 26 (3), 349-358.
- Tarte, S., Brady, M., Bowman, A. K., and Terras, M. (2011). Image Capture and Processing for Enhancing the Legibility of Incised Texts. IN Vahtikari, V., Hakkarainen, M., Nurminen, A., *Eikonopoiia. Digital Imaging of Ancient Textual Heritage, Technological Challenges and Solutions*. Commentationes Humanarum Litterarum 129, Helsinki, Academy of Finland, 173-88.
- Tarte, S., Wallom, D., Hu, P., Tang, K., and Ma, T. (2009) An Image Processing Portal and Web-service for the Study of Ancient Documents. IN: *Proceedings of Paper the 5th IEEE International Conference on e-Science, 9 –11 December 2009, Oxford, UK*, 14-19.
- Terras, M. (2002) Image to Interpretation: *Towards an Intelligent System to Aid Historians in the Reading of the Vindolanda Texts*. D.Phil Thesis, Department of Engineering Science, University of Oxford.
- Terras, M. (2005) Reading the Readers: Modelling Complex Humanities Processes to Build Cognitive Systems. *Literary and Linguistic Computing*, 20, 41-59.
- Terras, M. (2006) Image to Interpretation: *An Intelligent System to Aid Historians in Reading the Vindolanda Texts*. Oxford: Oxford University Press.
- Terras, M. (2010) The Digital Classicist: Disciplinary Focus and Interdisciplinary Vision. IN: Bodard, G. and Mahoney, S. (eds.) *Digital Research in the Study of Classical Antiquity*. London: Ashgate, 171-189.
- Terras, M. (2011) Present, Not Voting: Digital Humanities in the Panopticon: Closing Plenary Speech, Digital Humanities 2010. *Literary and Linguistic Computing*, 26 (3), 257-269.
- Terras, M. and Robertson, P. (2005) Using Artificial Intelligence to Read Ancient Roman Texts. *Human IT*, 7 (3). <http://etjanst.hb.se/bhs/ith/3-7/mtpr.pdf> [09.11.2011]
- Tile Website (2011) *Text-Image Linking Environment: About*. <http://mith.umd.edu/tile/about/> [09.11.2011]
- Tracy, S. V. and Papaodysseus, C. (2009) The Study of Hands on Greek Inscriptions: The Need for a Digital Approach. *American Journal of Archaeology*, 113, 99-102.
- Trismegistos Website (2011) *About Trismegistos*. <http://www.trismegistos.org/about.php> [09.11.2011]

- Tupman, C. (2010) Contextual Epigraphy and XML: Digital Publication and its Application to the Study of Inscribed Funerary Monuments. IN: Bodard, G. and Mahoney, S. (eds.) *Digital Research in the Study of Classical Antiquity*. London: Ashgate, 73-84.
- Turban, E. and Watkins, P. R. (1986) Integrating Expert Systems and Decision Support Systems. *MIS Quarterly*, 10 (2), 121-136.
- Turing, A. M. (1950) Computing machinery and intelligence. *Mind*, 59, 433-460.
- Turing, A. M. (1963) Computing Machinery and Intelligence. IN: Feigenbaum, E. A. and Feldman, J. *Computers and Thought*. New York: McGraw-Hill, 11-35.
- Turner, E. G. (1973) *The Papyrologist at Work*. London: William Clowes & Sons.
- van Groningen, B. A. (1932a) Projet d'unification des systemes de signes critiques. *Chronique d'Egypte*, 7, 262-269.
- van Groningen, B. A. (1932b) De signis criticis in edendo adhibendis. *Mnemosyne*, 594, 362-365.
- Van Minnen, P. (1993) The Century of Papyrology (1892-1992). *The Bulletin of the American Society of Papyrologists*, 30, 5-18.
- Vollgraff, C. W. (1917) De tabella emptionis aetatis Traiani nuper in Frisia reperta. *Mnemosyne*, XLV, 341-52.
- Waldrop, M. M. (1985) Personal Computers on Campus. *Science*, 228 (4698), 438-444.
- Warwick, C., Fisher, C., Terras, M., Baker, M., Clarke, A., Fulford, M., Grove, M., O'Riordan, E., Rains, M. (2009) iTrench: A study of user reactions to the use of information technology in field archaeology. *Literary and Linguistics Computing*, 24 (2), 211-223.
- Warwick, C., Terras, M., Huntington, P. and Pappa, N. (2008). If You Build It Will They Come? The LAIRAH Study: Quantifying the Use of Online Resources in the Arts and Humanities through Statistical Analysis of User Log Data. *Literary and Linguistics Computing*, 23 (1), 85-102.
- Waterman, D. A. (1986) *A Guide to Expert Systems*. Reading: Addison-Wesley.
- Youtie, H. C. (1963) The Papyrologist: Artificer of Fact. *Greek, Roman, Byzantine Studies*, 4 (1), 19-32.
- Youtie, H. C. (1966) Text and Context in Transcribing Papyri. *Greek, Roman and Byzantine Studies*, 7 (3), 251-258.

APPENDIX A

TABLET 670

This edition of Vindolanda Tablet 670 was published in:

Bowman, A. K. and Thomas, J. D. (2003) *The Vindolanda writing-tablets (Tabulae Vindolandenses III)*. London: British Museum Press, 126-129.

Reproduced here with the permission of the authors.

DESCRIPTION

Inv.no.93.1215. 133 x 81 mm. Plates 22.

Archaeological data. Location: LXXX. Period: 6A laminate or top of Period 6 ditch.

Several fragments survive of a letter from Martius to Victor, which may well be mostly complete. In addition to what is transcribed below, there is a tiny scrap with traces of writing on one side only. The tablet is unusual in three respects, date, script and format.

Date: it was found in a much later stratum than most of the other tablets, a stratum which Robin Birley would date to the last quarter of the second century (cf. A.R.Birley (2002), 158). See also Introduction, pp. 11.

Script: it is written in a hand unlike any others from Vindolanda, a hand which is very cursive and at times distorts the letter-forms quite markedly, e.g. *esse* in i.3. *d* starts with a downward, wavy stroke which turns to form the bottom of the loop before a diagonal stroke cuts back through the first stroke (e.g. ii.4 *dabis*); *e* is often in the shape of a narrow v; *g* in i.7 has a remarkably long tail; *l* descends well below the line; *n* is written in two strokes, the lower one from top to bottom and the upper one from bottom to top; they are often not well joined and can look like a diagonal with a kink in the middle (e.g. i.8, *diligenter*); similarly *s*, except that the kink is in the opposite direction (e.g. *eis* in i.9); *p* is usually made with a loop; *u* can be a shallow curve, high up, in ligature with the letter following (in *agatur*, ii.1, it is appreciably reduced). Throughout there is considerable use of ligature, far more than is normal in the tablets; e.g. in *frater* (i.5) every letter is linked to the ones before and after. Particularly notable ligatures are *al* in *salute[m]* (i.2), *dt* in *quodte* (i.3), *ha* in *distrabant* (i.9). Such a script would be possible at the date to which the other tablets belong (ca. AD 90-120), but fits at least as well with the later date suggested by the archaeological evidence: it has some similarity to the variant of Old Roman Cursive to which Cencetti applied the word "strisciato" (cf. *Tab.Vindol.I*, p. 59); Cencetti considered this form to have come into common use around the middle of the second century, although he identified an early example in a papyrus of AD 103.

Format: the letter is written on two-thirds of one side (A) and two-thirds of the other side (B) of a single piece of wood; the other third of (A) was used for the address. We describe A as the front and B as the back. No notches or tie-holes are visible. Although there are several other letters written on both sides of the leaf, there is no other example (so far as we can tell) of a comparable format in the Vindolanda tablets. The tablet is best envisaged as consisting of three vertical sections, I, II and III. The first half of the letter, (i) below, was written on the front of I + II. The tablet was then turned over and the second half of the letter, (ii) below, was written on the back of III + II. The tablet was then folded in a

APPENDIX A

TABLET 670

This edition of Vindolanda Tablet 670 was published in:

Bowman, A. K. and Thomas, J. D. (2003) *The Vindolanda writing-tablets (Tabulae Vindolandenses III)*. London: British Museum Press, 126-129.

Reproduced here with the permission of the authors.

DESCRIPTION

Inv.no.93.1215. 133 x 81 mm. Plates 22.

Archaeological data. Location: LXXX. Period: 6A laminate or top of Period 6 ditch.

Several fragments survive of a letter from Martius to Victor, which may well be mostly complete. In addition to what is transcribed below, there is a tiny scrap with traces of writing on one side only. The tablet is unusual in three respects, date, script and format.

Date: it was found in a much later stratum than most of the other tablets, a stratum which Robin Birley would date to the last quarter of the second century (cf. A.R.Birley (2002), 158). See also Introduction, pp. 11.

Script: it is written in a hand unlike any others from Vindolanda, a hand which is very cursive and at times distorts the letter-forms quite markedly, e.g. *esse* in i.3. *d* starts with a downward, wavy stroke which turns to form the bottom of the loop before a diagonal stroke cuts back through the first stroke (e.g. ii.4 *dabis*); *e* is often in the shape of a narrow v; *g* in i.7 has a remarkably long tail; *l* descends well below the line; *n* is written in two strokes, the lower one from top to bottom and the upper one from bottom to top; they are often not well joined and can look like a diagonal with a kink in the middle (e.g. i.8, *diligenter*); similarly *s*, except that the kink is in the opposite direction (e.g. *eis* in i.9); *p* is usually made with a loop; *u* can be a shallow curve, high up, in ligature with the letter following (in *agatur*, ii.1, it is appreciably reduced). Throughout there is considerable use of ligature, far more than is normal in the tablets; e.g. in *frater* (i.5) every letter is linked to the ones before and after. Particularly notable ligatures are *al* in *salute[m]* (i.2), *dt* in *quodte* (i.3), *ha* in *distrabant* (i.9). Such a script would be possible at the date to which the other tablets belong (ca. AD 90-120), but fits at least as well with the later date suggested by the archaeological evidence: it has some similarity to the variant of Old Roman Cursive to which Cencetti applied the word "strisciato" (cf. *Tab.Vindol.I*, p. 59); Cencetti considered this form to have come into common use around the middle of the second century, although he identified an early example in a papyrus of AD 103.

Format: the letter is written on two-thirds of one side (A) and two-thirds of the other side (B) of a single piece of wood; the other third of (A) was used for the address. We describe A as the front and B as the back. No notches or tie-holes are visible. Although there are several other letters written on both sides of the leaf, there is no other example (so far as we can tell) of a comparable format in the Vindolanda tablets. The tablet is best envisaged as consisting of three vertical sections, I, II and III. The first half of the letter, (i) below, was written on the front of I + II. The tablet was then turned over and the second half of the letter, (ii) below, was written on the back of III + II. The tablet was then folded in a

concertina fashion so that the front of III and the back of I were on the outside. The address was written on the front of III. See the diagram below. Thus when the letter was received and opened, the front of I + II, (i) below, would have been revealed first; the recipient would then have turned the whole tablet over and read (ii), III back + II back, all in the order intended. Why the sender should have used such a strange format it is impossible to tell. There is a further oddity in that there are some three lines of faded writing, which we are unable to read, on the back of I. We wonder whether this is from an earlier text which was partially washed off.

The first column is apparently concerned with the appointment of the addressee as agent in a private legal matter, perhaps to do with an inheritance (see the notes). The most notable feature of the second column is the reference to a hitherto unknown place *Bremesio*, see ii.3-4 note.

Since we have the final salutations in ii.7ff. but not the final greeting, it is probable that the tablet is almost but not quite complete at the foot. The other three sides are partially preserved.

EDITION

- A. i.
 Martius Victori fratri
 karissimo salutē[m]
 scīas me recte eṣṣe quod te
 inuicem facere cupio procur[a]-
 5. torem te facio frater ..[
 [
] cogn.tis patriṣ .[
]. diligenter neque a.a. [
]...id eis diṣtrahant ..[
 10.]...te.[.....]...[

- B. ii.
 et quid cirç[a] eas res agatur
 peto per ocç[asi]onem scribis
 mihi si oc[cas]ionem Brēme-
 şione non h[abeb]is dabis Catarac-
 5. toni Dürm[u]leterano ..
 Hario in .[c. 12
 fuera[m] [c.5 P]roçulum
 et familiam [c.4]..onidiçem
 filiam .[c.6] Valentinum
 10. uexill[arium] et]...anum

- A.Address:
 Çoriṣ
 Victori
 eq(uiti) ar[m]...
 a [M]artio ..br

TRANSLATION

'Martius to Victor, his most dear brother, greetings. Know that all is well with me and I wish that the same may be true for you. I am making you agent, brother, the relatives(?) of my(?) father carefully nor they sell(?) anything(?) for them and write to me, I ask, what is being done about those matters when you have the chance. If you do not have the chance (to write) from Bremesio(?), give (your letter) at Cataractonium to Durmius(?) the veteran or(?) to Harius we had been. [Greet?] Proculus and (his?) family and your (?) daughter and Valentinus the *uexillarius* and -anus (Address) [Deliver] at Coria(?) to Victor, cavalryman, armourer, from Martius, clerk (?).'

COMMENTARY

- A.i.1. The name Martius has not previously occurred in the tablets. It is worth remarking that *RIB* 937 (Old Penrith, undated) records a stone set up by a man named Martius for his father (cf. A.i.4-5 note) who was an *emeritus* of the *ala Petriana* (the *ala* is attested in several tablets, see 700.back note). A Victor is attested in 180, 182, 608 and perhaps 394 and 694, but all these are from earlier levels.
- A.i.2. *karissimo*: the word does not occur elsewhere in a salutation in the tablets, but cf. CEL 77 and 141.
- A.i.3-4. We are confident of the reading even though little remains of *çi* in *sçias* and *eşşe* is so cursively written that it could be read quite differently. The only point where a different reading seems at all likely is if *tu ... facias* were substituted for *te ... facere*: the letter after *fac* could well be *i*, but the letter following has the characteristic curved top-stroke of *r*. There is a very similar expression at the start of another Vindolanda letter, 311.i.3-5: *ut scias me recte ualere quod te inuicem fecisse cupio*. It is something of an epistolary cliché to begin a letter with *scias* (or equivalent): see Cugusi (1983), 79, and CEL I, p. 24. For *recte esse* see OLD s.v. *recte* 9 and Cugusi (1983), 273; and for the whole phrase *scias ... esse* cf. CEL 142.47-48 (= P.Mich. VIII 468), *scias domo perb[e]ne omnia recte esse*. On the whole of this sentence see now Adams (2002).
- A.i.4-5. The *p* of *procur[a]torem* lacks the loop which is found elsewhere in *p* in this hand. For *facio procuratorem* cf. TLL X.2 1576.20. It is most probable that procurator here is being used of a private legal capacity, for which see TLL X.2 1573ff.; cf. especially the *procurator ex testamento* appointed by a soldier in CPL 221 (perhaps also CPL 225). One wonders whether the two men really were brothers and that *frater* is not in this letter a mere term of affection; but this cannot be the case if *patrīs mei* is right in 7. Perhaps Martius wishes to make Victor his procurator in order to prevent relatives of his father getting their hands on his father's property which he has inherited.
- A.i.6. The other side proves that there is a line almost wholly missing at this point (corresponding to the right-hand half of ii.6 plus the back of I).
- A.i.7. There is little doubt that the first four letters which survive in this line are *cogn* and that the next but one is *t*; thus *çognat-* or *çognit-* seems inevitable. The reading, however, seems to be *çognut*, for which we have no suggestion. If we are to read *çognat* or *çognit*, we have to assume that the first stroke of the apparent *u* is really a descender from a letter in the (lost) line above. A reference to *cognati patris* would make good sense with *procurator* in a private legal capacity; see the note to lines 4-5 above. After *patrīs*, *m* is a possibility, i.e. *m[ei]*; we have also considered *n[ostri]*, but there would not be room to fit it all on this line and the trace before *diligenter* in 8 does not look like *i*.
- A.i.7-9. Although not much can have been lost at left and right in these lines, we are unable to suggest any connected sense.

- A.i.8. After *neque* (which looks reasonably secure) it might be possible to read *agam[*, but we do not see how the sentence can have run.
- A.i.9. *uid* before *eis distrahant* looks possible, and *q* before it is perhaps also possible. For *distraho* meaning "sell" see OLD s.v.4 and TLL V.1 1542-3.
- B.ii.1. At the start *eş* would be an easier reading, but *eť* is possible and would seem to offer a more probable construction.
- B.ii.1ff. The way this passage is to be understood is discussed in Adams (2003), §6, whose suggestions (in particular that *peto* is to be taken as parenthetical and *scribis* as for the imperative) we have followed in our translation. The translation offered in A.R.Birley (2002), 159 is based on earlier readings of ours, which we have now improved after inspection of the original.
- B.ii.2-3. We had previously suggested *opt[i]onem* here in both lines, but inspection of the original made it clear that there was a larger gap between the fragments. We are reasonably confident that *occasionem* was the word used in both places. For the phrase *per occasionem* see OLD s.v. *occasio* 1b. *scribis*: we cannot read *scribes* (future for imperative, as *dabis* in line 4); on the use of the present indicative for the imperative see Adams (2003), 00.
- B.ii.3-4. *Br̥eme|sione*: we earlier read *Br̥emeñio* and took this to refer to the fort Brementium at High Rochester. The first letter in line 4 could indeed be *n*, though *s* is more probable, but the letters following *io* can hardly be read as anything other than *ne* in this hand; this would seem to make the reading *Br̥emesione* (or less probably *Br̥emeñione*) inevitable. We might think of *Bremesio* (or *Brementio*) as a hitherto unattested alternative name for *Brementium* (other place-names in Britain beginning *Brem-* are known; see PNRB s.vv.), but we do not want a reference here to High Rochester, if we have read the back correctly, since it is north of *Coria* (Corbridge). Our understanding is that the letter was sent to Victor at Corbridge who was on his way south to Catterick (Cataractonium). *Bremesio* therefore should be the name of a place on the route between Corbridge and Catterick. Prof. Malcolm Todd (per litt.) has suggested to us that *Bremesio* may have been the Roman name for Piercebridge, which was not previously known, and we find this suggestion attractive. On the Roman site there see Fitzpatrick and Scott (1999), 111-132; they state that there is no conclusive evidence for a fort or forts there in the second or earlier third centuries (p. 111) but that there is evidence for some military activity there in the mid-later second century (p. 114). The site would certainly have been of strategic importance.
- B.ii.4. *dabis* is no doubt future for imperative, with the object "your letter" being understood.
- B.ii.4-5. For other references to Cataractonium see *Tab. Vindol. II*, Index III s.v.
- B.ii.5. RNGCL recognises a *gentilicium* *Durmius*, though *Dur̥m[io]* looks a little short for the lacuna. At the end of the line *ueł* may be possible. For other references to veterans in the tablets see **593**.ii.2 note.
- B.ii.6. Hario: if *uel* is right at the end of 5, this will be a complete name (rather than the end of one). Again, RNGCL recognises this as a *gentilicium*. We are also reminded of (H)arrius in Catullus 84.
- B.ii.6-7. The trace after *in* is indeterminate; a possible supplement would be in *ç[uius centuria] fueramus*.
- B.ii.7. The lacuna is probably just enough to accommodate *saluta*. It seems clear that we are dealing with greetings at this point and are therefore near the end of the letter.
- B.ii.7-9. If this tablet is rightly assigned to c. 175-200, Proculus and Valentinus mentioned here are not the same as those who occur in other tablets (for Proculus see **611**.i.7 note and for Valentinus **255**.3).

- B.ii.8. After *familiam* one expects *tuam* or *eius*. Presumably a female name followed at the end of the line; either *niđiçem* or *niđițem* looks to be a good reading, suggesting a Greek name, but we have found nothing suitable.
- B.ii.9. The trace after *filiam* is indecisive; either *t[uam* et or *e[ius* et is possible. The presence of someone's daughter in a military environment is noteworthy; cf. 643.back.
- B.ii.10. For other references to *uexillarii* in the tablets see Tab.Vindol.II, Index IV s.v. The name may end *-çianum*.
- A.Address. It is probable that the address is all written in the same hand as the letter.
- A.Address.1. The position of this line, in the top left-hand corner, shows that what we have here is the name of the place to which the letter is to be delivered (see Introduction, p. 17). Probably five letters have been written of which the first two are badly obscured by dirt; then *ri* is almost certain, followed by a damaged letter which suits *s*. Since what can be seen of the first two letters is compatible with *co*, we think it very probable that the place name is *Çoriş* (i.e. Corbridge, see 154.7 note). For other letters in the Vindolanda material addressed to Coria see 312 and the stilus tablet Inv.no. 87.722 with Tab.Vindol.II, p. 364. According to Breeze and Dobson (1985), 13 Corbridge was not reoccupied as a normal fort after the abandonment of the Antonine Wall, but military activity continued at the site. This would seem to be sufficient to justify supposing that a letter was addressed to Victor there. See further Bishop and Dore (1988), 137-9.
- A.Address.3. *eq(uiti) ar̄m.[*: there is no mark of abbreviation after *eq*. Clearly Victor belonged to an *ala* or a *cohors equitata*. A.R.Birley (2001b), 26, (2002), 158-9, tentatively suggests *pr(aef)] eq a Mar[tio*, but this is not possible: even if a *Mar[tio* could be read, there is no room at the end of 2 or the beginning of 3 to supply *pr(aef)*. We are reasonably confident of the reading *arm* after *eq*. We think it most likely that we have a reference to the post of *armorum custos* (on which see von Domaszewski (1967), 55-6, Tab.Vindon., 61, with further bibliography). The title must have been abbreviated, though it is not certain how. Roger Tomlin has drawn our attention to ILS 2186 = Speidel (1965), No. 56, which lists among *nomina turmae* an *AVR VICTOR ARM*; he was undoubtedly the troop's *armorum custos*, so the text proves that in the right context *arm* was enough for identification. However, there is certainly something written after *arm* in our text. In the address of Tab.Vindon. 38 the title is abbreviated *armoru(m) cus(todi)* (cf. Tab.Vindon. 23). Here we think the reading *ar̄mōr̄ç* may be possible, i.e. *armor(um) c(ustodi)*.
- A.Address.4. Written on the slant as usual for the name of the sender. The traces immediately after *M]ar̄tio* are badly abraded. We think them compatible with *li* and suggest that Martius was a *librarius*. The word generally just means "clerk" but in the military context can refer to a *principalis* in the *officium* of the *praefectus* of an *ala* or a *cohort*, see von Domaszewski (1967), 56, 59.



Tablet 670 Side A. Image used with permission of the Centre for the Study of Ancient Documents, University of Oxford and the British Museum.



Tablet 670 Side B. Image used with permission of the Centre for the Study of Ancient Documents, University of Oxford and the British Museum.

Tablet 862

This edition of Vindolanda Tablet 670 is published in:

Bowman, A. K., Thomas, J. D. and Tomlin, R. S. O. (2010) *The Vindolanda Writing-Tablets (Tabulae Vindolandenses IV, Part 1)*. Britannia, 41, 187-224.

Reproduced here with the permission of the authors.

Inv.no. T02.25. BM Registration No.: 2004,0501.64. Dimensions: 93 x113 mm. Period: II or III. Location: V02-19A, remains of buildings below rooms 1 and 2 of Period IV structure. A.R.Birley (2003), 107-8; (2009), No. 5.
Plate 000

DESCRIPTION

A leaf with 11 lines of text written across the grain. Two tie-holes and notches at the top end. The top right corner is lost, and with it the end of line 1. A small part of the left edge is lost, and with it part of the first letter of line 4 and the first letter of line 6. Otherwise almost complete, but now fragmented into four pieces. Unusually the wood is oak, and special difficulty is caused by broken vertical 'lines' which look like ink, but must be grain, although sometimes it seems that ink has also run into the fibres. We do not agree with Birley that 'most of the surviving lettering is particularly difficult to read', since we have had the benefit of improved scans; but since his own reading is little more than 'traces' and disconnected letters except in lines 1-2 and *faber* in 7, we have not thought it necessary to indicate where it diverges from our own. The hand is idiosyncratic. Most striking is *t*, with its extended cross-stroke and sometimes exaggerated size, especially when it is the first letter of 8 and 11. The upper diagonal of *a* is often prolonged to the left, cutting the previous letter and extending still further. *r* has a long sinuous descender, and like *t* is sometimes exaggerated in size, notably as the second letter of 8. As in **160** (see below), there is occasional use of interpunct.

The text is a note of work done in the *fabrica* by the century of Firmus, dated 20 April. The heading may refer explicitly to *fabri*, and it is certain that three men are named and identified as *faber*, preceded by two items for the first *faber* and two for the second pair of *fabri*. They were hardly the actual workmen (since there must have been many more from a century), but rather the qualified craftsmen in charge of each project. The best parallel for the tablet as a whole is an Egyptian papyrus, *P. Berlin* 6765 = *ChLA* X, 409, which records work in a legionary *fabrica* over two successive days. Each of its two columns is headed by the date (18 and 19 April), and itemises objects made or rather processed (for example shields 'flattened'); but personnel are not named, only entered collectively by military status (*immunes*, *cohortales* etc.). At Vindolanda the best parallel is **160**, too fragmentary for full interpretation, which appears to record names in the nominative, followed by their trade and century; it too refers to *faber* (A12, B4, B9) and probably to *[fe]rrum* ('iron', A13). *Tab. Vindol.* III, App. **155** may also be noted; it too is dated (24 April), and records the numbers of men *in officis*, engaged in various duties relating to building-work.

EDITION

↓
 Xii K(alendas) Maiā[s] fa.[]
 [o]pus · fabricae
 ·(centuria)· Firmi
 .ocridem · factam ad uetu-
 5 ram · iussu Musuruni (centurionis)
 [c]ircolas factas at.ş.ās n(umero) viii
 Huennius faber
 trausersaria facta n(umero) v
 item ferrum [p]roductum
 10 Andauer faber
 T- *vacat*
 -agomas faber
vacat

TRANSLATION

‘20 April, craftsmen for (?)
 the work of the workshop,
 century of Firmus.
 A *uocris* made for transport,
 by order of Musurunus, centurion.
 [c]ircolas made ... to the number of 8.
 Huennius(?) craftsman.
 cross-bars made, to the number of 5.
 Also, iron drawn out.
 Andauer craftsman,
 Tagomas craftsman.’

COMMENTARY

1. After *ma* there is a vertical streak coinciding with the grain, better suited to *i* than *r*, perhaps followed by the edge of *a*; thus a date in April, not February. After the date, between the second tie-hole and the broken edge, are two downstrokes, the second curving left, both cut by a diagonal; part of a loop follows. They cannot be *Batau[orum?]* as Birley suggests, but a truncated *fab[...]* resembling that in *fabricae* (2) is perfectly possible; the diagonal of *a* would be elongated, but not so acute, as usual. The word was hardly *fab[rica]*, in view of line 2, but reference to *fab[ri]* here would anticipate *faber* nicely in 7, 10 and 12. After it there would be space for *ad*, as the preposition before *opus*, of which there is no sign in 2; the latter was evidently centred to complete the heading.
3. The centurial sign (resembling a modern ‘7’) is unmistakable, with its descender

reaching the foot of the line below. The bold *i* which terminates *Firmi* overlaps with the descender of *r* from the line above, to the right of which are two vertical streaks in the fibre, but they are not in alignment with 3, and are probably not letters. A ‘century of Firmus’ is not attested, but in **180** one Firmus gives orders; he is evidently an officer, and the phrase *iussu Firmi* there (**180.23**, and perhaps also 12) is reminiscent of *iussu Musuruni (centurionis)* here in 5.

4. The first word presents problems of reading and interpretation. Only a curved stroke remains of the first letter, which is most likely *u*; then *o*, and *c* (like that in *fabricae*) rather than *p*; finally *-ridem*, the *i* being reduced to a streak in the fibre. Hence *uocridem*, the accusative of **uocris*, a noun not attested. In response to our enquiry, Dr. Paul Russell (University of Cambridge) kindly supplied the following note which we reproduce verbatim: “This is likely to be a Latinisation of a Celtic word. Assuming that the first letter is *u-*, then the stem *uocrid-* is probably to be identified with the stem of the Old Irish verb *fo-cridigedar*, parts of which are used to gloss compounds of the Latin verb *cingere* ‘bind’ in the Milan glosses on the Pauline epistles; see *Thesaurus Palaeohibernicus. A Collection of Old-Irish Glosses, Scholia, Prose and Verse*, ed. W. Stokes and J. Strachan, 2 vols (Cambridge, 1901-3), I.55 (fol. 27c5), I.70 (fol. 31c7), etc. The formation of verbs in *-(a)igidir* is usually denominative, and so the stem is probably nominal or adjectival. The same stem can also be found in the word for a belt, Old Cornish *grugis* (glossing *cingulum*), Old Breton *guecrissou*, Welsh *gwegrys*, Middle Irish *fochrus* < **wo-kris* (< **-krid-tu-*), and the simple uncompounded stem in Old Cornish *kreis* (glossing *camisia*), Middle Welsh *crys* ‘shirt’, Old Irish *criss* (*u*-stem) ‘belt, binding’ < **krid-tu-* (and perhaps also, as suggested by Schrijver, in the Cornish and Breton words for ‘middle’, Middle Cornish *kreys* and Breton *kreiz*). For the linguistic details, see J. Vendryes, *Lexique Étymologique de l’Irlandais Ancien C*, ed. E. Bachellery and P.-Y. Lambert (Dublin/Paris, 1987), C-236, s.v. *cridig-*, C-238, s.v. *cris*; for a different view, which may require modification if this analysis of *uocridem* is correct, see P. Schrijver, *Studies in British Celtic Historical Phonology* (Amsterdam, 1995), 321. The prefix **wo-* in such forms probably means ‘under’ (< **upo-*) and so semantically it has a similar structure to Latin *succingulum*, *succinctus*, etc. Thus, we would have stem meaning literally ‘under-binding’; in the insular Celtic languages it is used most often of clothing but there is no reason why it should not be used in other semantically similar contexts. In reference to a part of a wagon, it may be some kind of binding which holds the framework of the wagon together, or perhaps some kind of tarpaulin which covers the load and ties underneath.”

The next word ends in *actam* or *aptam* (but the letter before the obvious *t* looks more like *c* in *fabricae* than *p*), and the upward stroke of *f* can just be distinguished from the cross-stroke of *t* and the downstroke of *i* at the end of *Firmi* (3). This reading is supported by *factas* (6) and *facta* (8) in the corresponding entries below. Although *uecturam* / *uecturas* means ‘transport (costs)’ in **615.A.2** and **649.ii.12** and 14, as Russell’s note suggests, the sense here is more likely to be concrete, ‘a (vehicle for) transport’, as in **600.ii.2**, *ad capsum ueturae*, ‘for the body of the carriage’; the loss of *c* is a trivial Vulgarism (Adams (2003), 558-9).

5. For *iussu* with the name of the originating officer in the genitive, cf. **180.5**, 12 and 23, **858.8** (?); *RMR* 63 ii 4 and 5. The centurion’s name is not attested, but the name *Musuri[us?]* occurs on a military diploma found at Lyon (*CIL* XVI 147). The reading is certain except for the second vowel, which has been damaged in the break; but the traces and the spacing suit *u* better than *i*.
6. Like **uocris*, this seems to be another technical term, perhaps **[c]ircolas* for *circulos*, ‘circular’ objects apparently connected with transport. In view also of the *fabri* (7, 10, 12) and *ferrum* (9), we conjecture that they are iron hoops, ‘tyres’ made for wheels. According to Quintilian (I, 5.8), *cantus* was the word for ‘the iron-binding of wheels’

(*ferrum quo rotae uinciuntur*), but he regards it as not Latin; certainly it is rare, and there may have been a more homely synonym. Diocletian's Price Edict (15.31-40) quotes for wheeled vehicles 'without iron' (*sine ferro*); the Latin of this chapter is defective, but the Greek translation shows that iron fittings as well as tyres are meant: note especially 15.36, *u[ehicula cum canthis et ferr]aturis*. The adjective(?) after *factas* may refer to the fitting of these tyres, but we cannot decipher it.

The first digit of the numeral is cut by the descender of the centurial sign (5), and is certainly *v*, not *x*. A little after *viii*, and running at a different angle, there is apparently a fine diagonal stroke descending below the line; but we do not think it is part of the numeral, and are not even sure it is ink.

7. The name begins with *Hue-* and ends in *-us*, with five digits in between. There is some indication that they fall into two pairs each linked by a cross-stroke, followed by *i*, which makes *Huennius* an attractive reading, but not certain since the name is unattested (see, however, the nomen *Vennia* in Alföldy (1968), p.52, no. 134). The aspirated *u* identifies it as German, and perhaps it should also be read in **184.27**. *Huennius*(?), like Andauer (10) and Tagomas (11), is identified as a *faber*, which does not mean simply a workman in a *fabrica* (that would be *fabricensis*), but a specialist craftsman, an 'engineer' or 'smith', one of the many *immunes* listed by the Digest (L.6, 7). Evidently they were supervisors in charge of the various projects.
8. Initial *t* is exaggerated (as in 11), and the descender of *r* reaches the foot of 10 and the leaf-edge. The noun *transuersaria* (neuter plural) is used by Vitruvius (X, 14.1) for the cross-beams under a *testudo*, a strongly-built siege engine on wheels. In this context of transport and iron-working, *tra(ns)uersaria* would be iron braces 'across' a vehicle to hold it together, one of the iron fittings to which the Price Edict refers (see above, note to 6). *trauersaria* is an alternative form with reflexes in Romance (e.g. Fr. *travers*); the grammarian Velius Longus notes that *trans* was sometimes reduced to *tra-* in compounds (Keil, *Grammatici Latini* VII, p. 66).
There is a horizontal line above *n(umero) viii*, but this is not to mark the numeral; it extends from the *t* of *facta*. There is another just below, which belongs to the *t* of *[p]roductum*.
9. *item* is frequent in lists and accounts (at Vindolanda **180.10**, 31, 34 and 37; **182 i.7**) to introduce a new entry. *ferrum* (which in association with *faber* should probably be restored in **160.13**) is qualified here by a compound of *ductum*, for which there is sufficient trace of the prepositional *[p]ro*. This technical sense is omitted from standard dictionaries, but is evident in Juvenal, *Sat.* XV, 165-6, who uses *produxisse* (with the synonym *extendere*) in the sense of forging 'deadly iron' into weapons; and it is well illustrated by Cyprian, *Ep.* 69.8, *turibula quoque ipsa ... conflata atque igne purgata in laminas ductiles producuntur* ('[pagan] censers are melted and purified by fire, before being drawn out into metal sheets'). But dictionaries do recognize the derived adjective *productilis*, frequent in the Vulgate in the sense 'of beaten work, of highly wrought metal' (to quote Lewis and Short).
10. The name is not attested. The prefix *ande-* and the element *uero-s* are frequent in Celtic personal names, but in the circumstances this is more likely to be a German name.
- 11-12. Initial *t* is greatly exaggerated so as to fill half a line, the rest of which is blank; the name was completed on the line below. Tagomas is not the *uexillarius* (**861.25** note) since he is identified as *faber*.



Tablet 862 Front. Image used with permission of the Centre for the Study of Ancient Documents, University of Oxford and the British Museum.

APPENDIX B

The following are extracts from the notes sent between scholars S1, S2 and S3 during the process of interpreting Vindolanda Tablet 862. The final edition of tablet 862 is published in (Bowman *et al.* 2010) and reproduced in Appendix A.

Used with permission of the authors of Bowman, Thomas and Tomlin 2010.

SCHOLAR 1 (S1)

XII K MA[...] FAB[
OPVS FABRICAE
7 KARATI *horizontal line*
[S]VPER IDEM ACTAM AD VECTV
RAM IVSSV MVSIRVNI 7
[.]IRCOBAS ACTAS ANV N VIII
HVELDINVS FABER
T
RAMERA[G]AS FAB[ER] LIV *barred*
ITE[M] FERRVM P C *numeral*
ATOB[.]FER FA[B]ER
T
AGOMAS FABER

- 1: XII K Ma[...] There is not enough to decide between March and May. I seriously considered NOV, since the fourth stroke of M has been lost, and it resembles N (in some hands; but this scribe apparently writes N with a horizontal second stroke, and then down, not up; but his other examples of N are somewhat conjectural). M is followed by what looks like O, and all that prevents me from reading it is (i) quite good trace of the second stroke of A, (ii) no sign of the expected V. At the end of the line, S3 reads FAB, which may be right. (Certainly not Birley's BATAV, and I am moving away from my numeral XX[.] .) Difficulty is the F with its crossing horizontal resembles X, but the slope of the next letter is different. There may be the loop of B as well. Presumably fab[ri], 'smiths to the work of the smithy', although this must be resolved with the specification of fabri below. ad may have followed, or has been lost entirely at the beginning of 2.
- 4: S3 and I agree on the sequence DEM, and before it I see a vertical staining in the fibre appropriate to I (whether 'idem' or idem = 'item'). But it is possible that the scribe avoided the long descender of the centurial sign. The preceding letter is certainly R. Before that is C or P (better than my 'E', which requires a lost upper half). Before that I read an elaborate P, but O is better. And finally, at the very edge, a diagonal stroke appropriate to V – but which might also be initial S. However, SOCRIDEM is no help at all. At the end of the line, we both agree on vectu|ram, but he prefers ve(c)tu|ram, and certainly the spacing supports this.

SCHOLAR 2 (S2)

- 1:** Remains difficult and I can see nothing clear after *ma*; but since we must choose between March and May I also will go along with *maias*. But, as S1 says, we surely can't suppose the hole was drilled after the tablet was written (his diagonal stroke is certainly from *b* in the line below). At the right he points out there are two verticals but I would not accept his *h*. I would be prepared to read the first as *i* (cf. *iussu* in 5 for the hook at the top). Then I could read either *x* or *a* but don't see where either would lead (we do have a 9th cohort of Batavians, *ix c[oh bat]*, but this is desperate). Any ink before it?
- 4:** I agree entirely with S1's initial paragraph and also with S3 that we must have an unattested word. Is there a monograph on Roman transport vehicles? Essentially I agree with S1's analysis in the next two paragraphs; it is true that the 3rd letter is unlike *p* in *opus* (curved at the foot) so I would go along with *c* as to be preferred. It is the initial trace which bothers me and here again I agree it looks most like *u* and we have to rule out *a* - it's really very low in the line for *s*. I remain confident of *aptam*, not because it is a better reading than *actam* (I had originally read *factam* but *f* is not there), but because it makes such admirable sense. (S1's comment on this rather contradicts what he said before about *c* and *p* earlier in this line.) The rest of S1's note on this line is fine.
- 9:** I can't do this. Either *ferru* or *ferro* looks possible and I agree the line seems later to have *c* with a dash over it, i.e. a numeral; but it seems to have a huge *u/v* before it. Also there is something written after it and the line seems to end *a*. Could the dash in fact be the horizontal of one of his *t*'s and so a word ending *-ucta*? This only suggests *ducta* and how could it fit? 'tipped with iron'??

APPENDIX C

Videos of the examples from the DSS using tablet 670 are on the attached CD in the folder *Examples*. The videos are SVG format and should be viewable in a browser if Flash player is installed. They have been tested in the following browsers Safari, Firefox and Internet Explorer.

EXAMPLE 1: ADDING MEANING

Demonstrates with Tablet 670, line 7, first word, fifth character space how it is possible to input three different letters without any of them being resolved.

First the space is blue with a star (*) because there are no letters linked to it. The letter *a* is then added, but not resolved and the space turns yellow with the letter *a* in it. This is followed by the letter *i* being added, still no letters are resolved and the space turns red with a question mark in it. Finally the letter *u* is added and the letter *a* is resolved. The space is now green and has the letter *a* in it. Once one letter has been resolved it is not possible to resolve any more letters. However, in this example the scholars are not prepared to resolve any of the letters so the space is left red with a question mark.

EXAMPLE 2: ADDING ARGUMENTS

This example uses Tablet 670, line 4, fourth word, first letter to demonstrate how letters can be linked to arguments that again can be linked to characters and words in the document.

The first letter is an unresolved *p* (yellow). One of the arguments against this letter is “*The p of procur[a]torem lacks the loop which is found elsewhere in p in this hand*“. This argument is linked to another *p* in the document.

EXAMPLE 3: SEARCHING FOR CONCORDANCES

Tablet 670, line 5, second word is used to demonstrate how the Vindolanda Tablet APPELLO (word search Web Service) can be searched for concordance parallels to a word-pattern in the text.

The word in this case is *facio* – however, the letters *f* and *a* are not resolved (yellow) so the pattern is ***cio* where *** signifies a character space that can have any letter in it.

The search returns many results ranging from *Lucio* to *facio* but most importantly each result is linked to the tablet in which the word occurs so that it is possible to see the context in which the word has been used.

EXAMPLE 4: SEARCHING LEXICONS

In this example, Tablet 670, line 1, third word is used to search for lexical parallels. This example will search for the word *fratri* in Perseus Latin Dictionaries.

This returns two results, one from Thomas Cooper and one from Lewis & Short. Each result in turn links to the record for their particular word on the original lexical website where it is possible to read more about the word and what it means.

EXAMPLE 5: VINDOLANDA TABLETS ONLINE II WEBSITE

This is a demonstration of how the VTO2 website (<http://vto2.classics.ox.ac.uk/>) can be used.

From the front page the simplest search functionality can be reached through the menu `Tablets > Search for Tablets`. Here it is possible to search for a particular tablet if the tablet ID is known – for example Tablet 670.

If the tablet ID is not known it is possible to use the menu `Tablets > View all tablets` to browse through the tablets. Here it is possible to use sorting and filtering to find

a tablet or a set of tablets. It is possible to filter so that only the tablets in archaeological period 3 are listed and then to sort them by their old inventory number.

A third method of finding a particular tablet is to search through their content using one of the index menus such as the menu `Indices > General Index of Latin words`. When typing in the search word the results list automatically updates itself. This index search also retrieves variations of word as in the example with the word *sum*.

In the actual tablet view the published edition of the tablet is present. The images for each tablet can be viewed and the references are linked to either the bibliography or the other tablets. Each word also activates a pop-up box in which is listed other tablets with the same word. Furthermore the *N* markings at the beginning of some of the lines activate a pop-up with the commentary for that line. The commentary is also listed below the translation.

APPENDIX D

Code for the DSS prototype in the order of the file list (Appendix C) – all developed by Henriette Roued-Cunliffe. Each file is divided by a horizontal line. See original files on CD in folder DSS Prototype.

```
<?php
/**
 * index.php
 *
 * Index page for DSS Prototype
 *
 * @package    DSS prototype
 * @author     Henriette Roued-Cunliffe <henriette@roued.com>
 * @license    for OS license
 * @link       http://roued.com/e-doc
 * @since      File available since 2011
 */
// Start session
session_start();
// Include the config file
include_once('config.php');
?>
<!-- Start of the HTML head -->
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <!-- Add meta data -->
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <meta name="description" content="DSS prototype - <?php echo $document;
?>" />
    <meta name="keywords" content="DSS, prototype, <?php echo $keywords;
?>" />
    <meta name="authors" name="<?php echo $authors; ?>" />
    <!-- Add base address for all links -->
    <base href="<?php echo $base; ?>" />
    <base target="_blank" />
    <!-- Add the css style -->
    <link href="<?php echo $css; ?>" type="text/css" rel="stylesheet" />
    <!-- Add the js script -->
    <script type="text/javascript" src="<?php echo $scripts; ?>"></script>
    <!-- Title of this DSS prototype example -->
    <title>DSS Prototype - <?php echo $document; ?></title>
    <!-- End of the HTML head -->
  </head>
  <!-- Start of the HTML body - on load the js function view() is run -->
  <body onload="view();showEng('xml.php?do=get&get=eng');">
    <!-- dark screen overlay -->
    <div id="screenoverlay" style="visibility:hidden;"></div>
    <!-- argument box overlay -->
    <div id="argtopbox" style="visibility:hidden;">
      <div id="argbox"></div>
      <a class="close" href="#" onClick="closeArgTop();return
false;">Close</a>
    </div>
    <!-- suggestion box overlay -->
    <div id="sugtopbox" style="visibility:hidden;">
      <div id="sugbox"></div>
      <a class="close" href="#" onClick="closeSugTop();return
false;">Close</a>
    </div>
    <!-- first top box overlay -->
    <div id="topbox" style="visibility:hidden;">
      <div id="box"></div>
      <span class="error_msg" id="doSomethingText"></span>
      <a class="close" href="#" onClick="closeTop();return
false;">Close</a>
    </div>
```



```

<!-- left sidebar box -->
<div id="left">
    <div id="title"> <h1>DSS Prototype - <?php echo $document; ?></h1>
    </div>
    <div id="image">
        
    </div>
    <div id="image">
        
    </div>
    <div id="engbox"></div>
</div>
<!-- main box -->
<div id="main">
    <div id="main-view"></div>
</div>
</body>
</html>
<?php mysql_close($con); ?>

```

```

<?php
/**
 * config.php
 *
 * Config page for DSS
 *
 * @package    DSS prototype
 * @author     Henriette Roued-Cunliffe <henriette@roued.com>
 * @license    for OS license
 * @link       http://roued.com/e-doc
 * @since      File available since 2011
 */

// Name of the document used in this DSS project
$document = '670';
// Keywords for this specific project (comma separated)
$keywords = 'Vindolanda Tablet, CSAD, Latin, Roman';
// Authors of the DSS project (comma separated)
$authors = 'Henriette Roued-Cunliffe';
// Image
$image = '670f_a.jpg';

// ---- MySQL connection ---- //
$con = mysql_connect("127.0.0.1","root","17122002");
if (!$con)
{
    die('Could not connect: ' . mysql_error());
}
mysql_select_db("DSS_".$document, $con);

// ---- FILE LIST ---- //
// Default address for all links
$base = 'http://localhost/dss_'.$document.'/';
// File containing the js scripts
$scripts = 'DSS_scripts.js';
// Stylesheet for the DSS
$css = "DSS_style.css";
// DBtoTEI Web Service - for retrieving database fields as XML
$DBtoTEI = $base .'db-tei.php?method=DBtoTEI';
?>

```

```

/**
 * DSS_style.css
 *
 * Style page for DSS
 *
 * @package    DSS prototype
 * @author     Henriette Roued-Cunliffe <henriette@roued.com>
 * @license    for OS license
 * @link       http://roued.com/e-doc

```

```

* @since      File available since 2011
*/
/* General formatting */
h1 {
    font-size:110%;
    font-variant:small-caps;
}
h2 {
    font-size:100%;
    font-variant:small-caps;
}
p{
    font-size:90%;
}
/* Main view box */
#main{
    background-color:#FFFFFF;
    margin:50px auto;
    max-width:900px;
    min-width:900px;
    padding:0;
    width:80%;
}
.main-view{
    width:100%;
    float:left;
    border: #dddddd solid 5px;
    padding:10px;
    margin-bottom:10px;
}
/* Line formatting - in main view */
.line{
    margin: 30px 0 30px 0;
    padding-right:10px;
    width:100%;
    color:#003366;
    font-size:20px;
}
.linenr{
    font-variant:small-caps;
    font-size:13px;
    color:#444444;
    padding-right:10px;
}
/* Word formatting - in main view */
.w{
    border: 2px solid #777777;
    padding: 13px 10px 10px 3px;
    margin:0 3px 0 3px;
    background-color:#cccccc;
}
a.word{
    font-size:12px;
    padding:10px 4px 10px 4px;
    background-color:#cccccc;
    color:#000000;
}
/* Character formatting - in main view */
.c{
    border: 2px solid #555555;
    padding:4px;
    background-color:#eeeeee;
    color:#333333;
    font-family:Arial, Helvetica, sans-serif;
    margin-right:2px;
}
/* legend for main view */
.red{
    background-color:#f9996a;
}
.green{
    background-color:#b3e594;
}

```

```

.yellow{
  background-color:#f3e871;
}
.blue{
  background-color:#b2e7f8;
}
/* Left sidebar */
#left{
  border: #dddddd solid 5px;
  width:15%;
  float:left;
}
#title{
  padding:2%;
}
.image{
  width:98%;
  margin:1%;
}
/* Link view of the document */
.link-view{
  float:left;
  border-top: #888888 solid 1px;
  border-bottom: #888888 solid 1px;
  border-left: #888888 solid 1px;
  margin-top: 3px;
  padding:2px;
}
/* Line formatting - in link view */
.link-line{
  margin: 10px 0 20px 0;
  padding-right:3px;
  width:200%;
  color:#003366;
  font-size:20px;
}
.link-linenr{
  font-variant:small-caps;
  font-size:12px;
  color:#000000;
  padding-right:10px;
  position:relative;
}
/* Word formatting - in link view */
.link-w{
  border: 2px solid #777777;
  padding: 4px 4px 6px 2px;
  margin:0 2px 0 2px;
  background-color:#cccccc;
}
.link-w-highlight{
  border: 2px solid #8D4D43;
  padding: 4px 4px 6px 2px;
  margin:0 2px 0 2px;
  background-color:#d47465;
}
a.link-word{
  font-size:12px;
  padding:6px 4px 8px 4px;
  background-color:#cccccc;
  color:#000000;
}
/* Character formatting - in link view */
.link-c{
  background-color:#eeeeee;
  color:#333333;
  font-family:Arial, Helvetica, sans-serif;
  margin-right:2px;
  border: 2px solid #555555;
  font-size:12px;
  padding:3px;
}
.link-c-highlight{

```

```

border: 2px solid #8D4D43;
background-color:#d47465;
font-size:12px;
padding:3px;
font-family:Arial, Helvetica, sans-serif;
margin-right:2px;
}
/* Screenoverlay formatting */
#screenoverlay{
background-color: #000000;
opacity: .75;
filter: alpha(opacity=75);
position: fixed;
left: 0px;
top: 0px;
width: 100%;
height: 100%;
z-index: 1000;
}
/* showTop() formatting */
#topbox{
position: fixed;
left: 1000px;
top: 100px;
padding: 5px;
width: 400px;
height: 400px;
z-index: 1001;
}
#box{
background:#ffffff;
opacity: .85;
min-height: 80px;
padding:20px;
max-height:400px;
overflow:auto;
}
/* showArg() formatting */
#argtopbox{
position: fixed;
left: 300px;
top: 100px;
padding: 5px;
width: 400px;
height: 400px;
z-index: 1001;
}
#argbox{
background:#ffffff;
opacity: .85;
min-height: 80px;
padding:20px;
max-height:400px;
overflow: auto;
}
/* showSug() formatting */
#sugtopbox{
position: fixed;
left: 300px;
top: 100px;
padding: 5px;
width: 400px;
height: 400px;
z-index: 1001;
}
#sugbox{
background:#ffffff;
opacity: .85;
min-height: 80px;
padding:20px;
max-height:400px;
overflow: auto;
}

```

```

/* close boxes formatting */
a.close{
    padding: 2px 5px 5px 358px;
    position:relative;
    color:#B31A14;
    background:#ffffff;
    opacity: .85;
}



---



/**
 * DSS_script.js
 *
 * Script page for DSS
 *
 * @package    DSS prototype
 * @author     Henriette Roued-Cunliffe <henriette@roued.com>
 * @license    for OS license
 * @link       http://roued.com/e-doc
 * @since      File available since 2011
 */
/* ----- View Functions - created main view of the DSS ----- */
// delay for view1()
function view()
{
    var func="view1()";
    setTimeout(func, 200);
    document.getElementById("main-view").innerHTML = '<span></span>';
}
// runs stateChangedView() with view.php response
function view1() {
    // Make object
    xmlhttpView=GetXmlHttpRequestObject();
    // If the object is not created:
    if (xmlhttpView==null){
        alert ("Your browser does not support AJAX!");
        return;
    }
    // Request result from view.php
    var urlView="view.php";
    xmlhttpView.onreadystatechange=stateChangedView;
    xmlhttpView.open("POST",urlView,true);
    xmlhttpView.send(null);
}
//sends response text on to 'main-view'
function stateChangedView() {
    if (xmlhttpView.readyState==4){
        document.getElementById("main-view").innerHTML=xmlhttpView.responseText;
    }
}
/* ----- end of View Functions ----- */

/* ----- Engine Functions - creates the search engine box next to the main view -
--- */
// delay for showEng1()
function showEng(URL){
    var func="showEng1('"+URL+"')";
    setTimeout(func, 100);
    document.getElementById("engbox").innerHTML = '<span></span>';
}
// runs stateChangedEng() with 'get=eng' response
function showEng1(URL)
{
    URL=URL;
    // Make object
    xmlhttp=GetXmlHttpRequestObject();
    // If the object is not created:
    if (xmlhttp==null){

```

```

        alert ("Your browser does not support AJAX!");
        return;
    }
    xmlhttp.onreadystatechange=stateChangedEng;
    xmlhttp.open("POST",URL,true);
    xmlhttp.send(null);
}
// sends response text on to 'engbox'
function stateChangedEng() {
    if (xmlhttp.readyState==4){
        document.getElementById("engbox").innerHTML=xmlhttp.responseText;
    }else {
        document.getElementById("engbox").innerHTML = '<span></span>';
    }
}
/* ----- end of Engine Functions ----- */

/* ----- showTop Functions - displays things in pop-up boxes on top of the main view ----- */
// delay for showTop1()
function showTop(URL, index){
    var func="showTop1('"+URL+"','"+index+"')";
    setTimeout(func, 200);
    document.getElementById("box").innerHTML = '<span></span>';
}
// creates 'topbox' and 'screenverlay' and runs stateChangedTop() with response
function showTop1(URL, index){
    URL=URL+"&index="+index;
    // Make object
    xmlhttp=GetXmlHttpRequestObject();
    // If the object is not created:
    if (xmlhttp==null){
        alert ("Your browser does not support AJAX!");
        return;
    }
    //Position the topBox.
    boxWidth = 700;
    boxHeight = 200;
    screenWidth=document.all?document.body.clientWidth>window.innerWidth;
    screenHeight=document.all?document.body.clientHeight>window.innerHeight;
    xPos = (screenWidth - boxWidth) * 0.5;
    yPos = (screenHeight - boxHeight) * 0.2;
    document.getElementById('topbox').style.left=xPos;
    document.getElementById('topbox').style.top=yPos;
    //Show the background overlay and topbox
    document.getElementById('screenoverlay').style.visibility = 'visible';
    document.getElementById('topbox').style.visibility = 'visible';
    xmlhttp.onreadystatechange=stateChangedTop;
    xmlhttp.open("POST",URL,true);
    xmlhttp.send(null);
}
// sends response text on to 'box'
function stateChangedTop() {
    if (xmlhttp.readyState==4){
        document.getElementById("box").innerHTML=xmlhttp.responseText;
    }
}
// closes box, topbox and screenoverlay
function closeTop(){
    //Hide the overlay and tobox...
    document.getElementById('screenoverlay').style.visibility='hidden';
    document.getElementById('topbox').style.visibility='hidden';
}
/* ----- end of showTop Functions ----- */

/* ----- showArg Functions - displays things in a second pop-up boxes on top of the main view ----- */
// delay for showTArg1()
function showArg(URL, index){
    var func="showArg1('"+URL+"','"+index+"')";
    setTimeout(func, 200);
}

```

```

    document.getElementById("argbox").innerHTML = '<span></span>';
}
// creates 'argtopbox' and 'screenverlay' and runs stateChangedArg() with
response
function showArg1(URL, index){
    URL=URL+"&index="+index;
    // Make object
    xmlHttp=GetXmlHttpRequest();
    // If the object is not created:
    if (xmlHttp==null){
        alert ("Your browser does not support AJAX!");
        return;
    }
    //Position the topBox.
    boxWidth = 700;
    boxHeight = 200;
    screenWidth=document.all?document.body.clientWidth>window.innerWidth;
    screenHeight=document.all?document.body.clientHeight>window.innerHeight;
    xPos = (screenWidth - boxWidth) * 0.5;
    yPos = (screenHeight - boxHeight) * 0.2;
    document.getElementById('argtopbox').style.left=xPos;
    document.getElementById('argtopbox').style.top=yPos;
    //Show the background overlay and topbox.
    document.getElementById('screenoverlay').style.visibility = 'visible';
    document.getElementById('argtopbox').style.visibility = 'visible';
    xmlHttp.onreadystatechange=stateChangedArg;
    xmlHttp.open("POST",URL,true);
    xmlHttp.send(null);
}
// sends response text on to 'argbox'
function stateChangedArg() {
    if (xmlHttp.readyState==4){
        document.getElementById("argbox").innerHTML=xmlHttp.responseText;
    }else {
        document.getElementById("argbox").innerHTML = '<span></span>';
    }
}
// closes argtopbox
function closeArgTop(){
    //Hide the overlay and tobox
    document.getElementById('argtopbox').style.visibility='hidden';
}
/* ----- end of showArg Functions ----- */

/* ----- Do Something Functions - allows the DSS to do different things like add
and del ----- */
// runs stateChangedDo() with response from xml.php
function doSomething(URL, index) {
    // Make object
    xmlHttpDo=GetXmlHttpRequest();
    // If the object is not created:
    if (xmlHttpDo==null){
        alert ("Your browser does not support AJAX!");
        return;
    }
    URL=URL+"&index="+index;
    URL=URL+"&sid="+Math.random();
    xmlHttpDo.onreadystatechange=stateChangedDo;
    xmlHttpDo.open("POST",URL,true);
    xmlHttpDo.send(null);
}
// sends response text on to 'doSomethingText'
function stateChangedDo() {
    if (xmlHttpDo.readyState==4){
        document.getElementById("doSomethingText").innerHTML=xmlHttpDo.responseText;
    }
}
/* ----- end of Do something Functions ----- */

/* ----- addText Function - allows the DSS to do add text ----- */
//sends text onto xml.php

```

```

function addText(URL, index, textbit) {
    // Make object
    xmlHttpAddText=GetXmlHttpRequestObject();
    // If the object is not created:
    if (xmlHttpAddText==null){
        alert ("Your browser does not support AJAX!");
        return;
    }
    URL=URL+"&index="+index;
    URL=URL+"&text="+textbit;
    URL=URL+"&sid="+Math.random();
    // Posts the text to xml.php
    xmlHttpAddText.open("POST",URL,true);
    xmlHttpAddText.send(null);
}
/* ----- end of addText Function ----- */

/* ----- showSug Functions - functions for bringing up word suggestions ----- */
// delay for showSug1()
function showSug(URL, pattern){
    var func="showSug1('"+URL+"','"+pattern+"')";
    setTimeout(func, 10);
    document.getElementById("sugbox").innerHTML = '<span></span>';
}
// creates sugtopbox and screenoverlay and runs stateChangedSug() with get=sug
function showSug1(URL, pattern){
    URL=URL+"&pattern="+pattern;
    // Make object
    xmlHttp=GetXmlHttpRequestObject();
    // If the object is not created:
    if (xmlHttp==null) {
        alert ("Your browser does not support AJAX!");
        return;
    }
    //Position the topBox. In this example I am just centering it on the screen
    boxWidth = 700;
    boxHeight = 200;
    screenWidth=document.all?document.body.clientWidth>window.innerWidth;
    screenHeight=document.all?document.body.clientHeight>window.innerHeight;
    xPos = (screenWidth - boxWidth) * 0.5;
    yPos = (screenHeight - boxHeight) * 0.2;
    document.getElementById('sugtopbox').style.left=xPos;
    document.getElementById('sugtopbox').style.top=yPos;
    //Show the background overlay and topbox...
    document.getElementById('screenoverlay').style.visibility = 'visible';
    document.getElementById('sugtopbox').style.visibility = 'visible';
    // Send the url if the state of the input box is changed.
    xmlHttp.onreadystatechange=stateChangedSug;
    xmlHttp.open("POST",URL,true);
    xmlHttp.send(null);
    //document.getElementById("box").innerHTML=xmlHttp.responseText;
}
// sends response text to 'sugbox'
function stateChangedSug() {
    if (xmlHttp.readyState==4){
        document.getElementById("sugbox").innerHTML=xmlHttp.responseText;
    }else {
        document.getElementById("sugbox").innerHTML = '<span></span>';
    }
}
// closes sugtopbox
function closeSugTop(){
    // Hide the overlay and topbox
    document.getElementById('sugtopbox').style.visibility='hidden';
}
/* ----- end of showSug Functions ----- */

/* ----- GetXmlHttpRequestObject Function ----- */
// creates an XML http object
function GetXmlHttpRequestObject(){
    var xmlHttp=null;

```



```

try {
    // Firefox, Opera 8.0+, Safari
    xmlhttp=new XMLHttpRequest();
}
catch (e) {
    // Internet Explorer
    try{
        xmlhttp=new ActiveXObject("Msxml2.XMLHTTP");
    }
    catch (e){
        xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
    }
}
return xmlhttp;
}

/* ----- end of GetXmlHttpRequest Functions ----- */

/* ----- disableEnterKey Functions - functions disabling the enter key ----- */
// so that pressing 'enter' doesn't reload the page
function disableEnterKey(e){
    var key;
    if(window.event)
        key = window.event.keyCode;      //IE
    else
        key = e.which;      //firefox
    if(key == 9)
        return false;
    else
        return true;
}
// so that pressing 'enter' doesn't reload the page
function enter_pressed(e){
    var keycode;
    if (window.event) keycode = window.event.keyCode;
    else if (e) keycode = e.which;
    else return false;
    return (keycode == 13);
}

/* ----- end of disableEnterKey Functions ----- */

```

```

<?php
/**
 * view.php
 *
 * Processing script for the js function view()
 *
 * @package    DSS prototype
 * @author     Henriette Roued-Cunliffe <henriette@roued.com>
 * @license    for OS license
 * @link       http://roued.com/e-doc
 * @since      File available since 2011
 */
// FILE: config.php
include_once('config.php');
// FUNCTION: DOMDocument()
$xml = new DOMDocument();
// FILE: view.xsl
$xml_name = 'view.xsl';
// FILE: db-tei.php
$xml->load($DBtoTEI);
$xml = new DOMDocument();
$xml->load($xml_name);
$proc = new XsltProcessor();
$xml = $proc->importStylesheet($xml);
$proc->setParameter(null, ' ', ' ');
$newdom = $proc->transformToDoc($xml);
$view = $newdom->saveXML();
// OUTPUT: main DSS view
echo $view;

```

?>

```

<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<!--
* view.xsl
*
* XSLT stylesheet for the js function view()
*
* @package    DSS prototype
* @author     Henriette Roued-Cunliffe <henriette@roued.com>
* @license    for OS license
* @link       http://roued.com/e-doc
* @since      File available since 2011
-->

<!-- Remove all extra spaces in the document -->
<xsl:strip-space elements="*" />
<xsl:template match="/">
  <xsl:apply-templates select="//item" />
</xsl:template>

<xsl:template match="//item">
  <div class="main-view">
    <xsl:choose>
      <xsl:when test="//1">
        <xsl:apply-templates select="//1" />
      </xsl:when>
    </xsl:choose>
  </div>
</xsl:template>

<!-- Transform lines -->
<xsl:template match="//1">
  <div class="line">
    <span class="linenr">Line <xsl:value-of select="@n" />: </span>
    <xsl:apply-templates select="w" />
  </div>
</xsl:template>

<!-- Transform words -->
<xsl:template match="w">
  <span class="w">
    <a class="word">
      <xsl:attribute name="onclick">
        <!-- FUNCTION: showTop(get=word) -->
        <xsl:text>showTop( 'xml.php?do=get&get=word', '</xsl:text>
        <xsl:value-of select="@n" />
        <xsl:text>')</xsl:text>
      </xsl:attribute>
      <xsl:text>*</xsl:text>
    </a>
    <xsl:if test="c">
      <xsl:apply-templates select="c" />
    </xsl:if>
  </span>
</xsl:template>

<!-- Transform characters -->
<xsl:template match="c">
  <span class="c">
    <xsl:attribute name="onclick">
      <!-- FUNCTION: showTop(get=char) -->
      <xsl:text>showTop( 'xml.php?do=get&get=char', '</xsl:text>
      <xsl:value-of select="@n" />
      <xsl:text>')</xsl:text>
    </xsl:attribute>
    <xsl:choose>
      <xsl:when test="string(.)">
        <xsl:param name="countg" select="count(g)" />
        <xsl:param name="countu" select="count(g[@ana='#unresolved'])" />

```

```

<xsl:param name="countr" select="count(g[@ana='#resolved'])"/>
<xsl:choose>
  <xsl:when test="$countr='1' and $countu='0'">
    <xsl:apply-templates>
      <xsl:with-param name="type" select="1"/>
    </xsl:apply-templates>
  </xsl:when>
  <xsl:when test="$countr='0' and $countu='1'">
    <xsl:apply-templates>
      <xsl:with-param name="type" select="2"/>
    </xsl:apply-templates>
  </xsl:when>
  <xsl:when test="$countr='1' and $countu!='0'">
    <xsl:apply-templates>
      <xsl:with-param name="type" select="3"/>
    </xsl:apply-templates>
  </xsl:when>
  <xsl:when test="$countr='0' and $countu!='0'">
    <span class="red"><xsl:text>?</xsl:text></span>
  </xsl:when>
</xsl:choose>
</xsl:when>
<xsl:otherwise>
  <span class="blue"> <xsl:text>*</xsl:text> </span>
</xsl:otherwise>
</xsl:choose>
</span>
</xsl:template>

<!-- Transform letter elements -->
<xsl:template match="g">
  <xsl:param name="type"/>
  <xsl:choose>
    <xsl:when test="@ana='#resolved' and $type='1'">
      <xsl:value-of select="."/>
    </xsl:when>
    <xsl:when test="@ana='#unresolved' and $type='2'">
      <span class="yellow"><xsl:value-of select="."/></span>
    </xsl:when>
    <xsl:when test="@ana='#resolved' and $type='3'">
      <span class="green"><xsl:value-of select="."/></span>
    </xsl:when>
  </xsl:choose>
</xsl:template>

</xsl:stylesheet>

```

```

<?php
/**
 * db-tei.php
 *
 * Web Service DB to TEI - outputs the structure of the document from the database
 * as a TEI formatted XML document
 *
 * @package    DSS prototype
 * @author     Henriette Roued-Cunliffe <henriette@roued.com>
 * @license    for OS license
 * @link       http://roued.com/e-doc
 * @since      File available since 2011
 */

// FILE: Zend/Rest/Server.php - on local server
include_once ('Zend/Rest/Server.php');

// FUNCTION: DBtoTEI()
function DBtoTEI() {
  // SET: connection to MySQL Database
  $con = mysql_connect("127.0.0.1","root","17122002");
  if (!$con)
  {
    die('Could not connect: ' . mysql_error());
  }
}

```

```

    }
    mysql_select_db("DSS_670", $con);
    // SET: method title
    $method_title = 'DSS DB to TEI method';
    // SET: authority text
    $authority1 = 'Made available by Centre for the Study of Ancient Documents,
University of Oxford (http://www.csad.ox.ac.uk/) and Henriette Roued-Cunliffe as
a part of her D.Phil at the University of Oxford (http://www.roued.com/)';
    // SET: todays date
    $today = date(d) . '-' . date(m) . '-' . date(Y);
    // SET: source
    $source = "DSS";
    // SET: languages
    $langs = array(
        'eng' => 'English',
        'lat' => 'Latin',
        'grc' => 'Greek',
    );
    $lang_list = array('eng', 'lat');
    // SET: error code
    $error_code = 0;
    // FUNCTION: DOMDocument()
    $doc = new DOMDocument();
    $doc->formatOutput = true;
    // SET: header elements
    $tei = $doc->createElement("TEI");
    $doc->appendChild($tei);
    // teiHeader
    $teiheader = $doc->createElement("teiHeader");
    $tei->appendChild($teiheader);
    // fileDesc
    $fileDesc = $doc->createElement("fileDesc");
    $teiheader->appendChild($fileDesc);
    // title
    $titleStmt = $doc->createElement("titleStmt");
    $fileDesc->appendChild($titleStmt);
    $title = $doc->createElement("title");
    $title->appendChild($doc->createTextNode($method_title));
    $titleStmt->appendChild($title);
    // publicationStmt
    $publicationStmt = $doc->createElement("publicationStmt");
    $fileDesc->appendChild($publicationStmt);
    $authority = $doc->createElement("authority");
    $authority->appendChild($doc->createTextNode($authority1));
    $publicationStmt->appendChild($authority);
    $date = $doc->createElement("date");
    $date->appendChild($doc->createTextNode($today));
    $publicationStmt->appendChild($date);
    // sourceDesc
    $sourceDesc = $doc->createElement("sourceDesc");
    $fileDesc->appendChild($sourceDesc);
    $p1 = $doc->createElement("p");
    $sourceDesc->appendChild($p1);
    $p1->appendChild($doc->createTextNode('From DSS Database'));
    // encodingDesc
    $encodingDesc = $doc->createElement("encodingDesc");
    $teiheader->appendChild($encodingDesc);
    $ab5 = $doc->createElement("ab");
    $ab5->appendChild($doc->createTextNode($encoding));
    $encodingDesc->appendChild($ab5);
    // profileDesc
    $profileDesc = $doc->createElement("profileDesc");
    $teiheader->appendChild($profileDesc);
    $textClass = $doc->createElement("textClass");
    $profileDesc->appendChild($textClass);
    $catRef = $doc->createElement("catRef");
    $textClass->appendChild($catRef);
    $catRef->setAttribute("target", 'List of '.$list_of.' from '.$source);
    $classCode = $doc->createElement("classCode");
    $textClass->appendChild($classCode);
    $classCode->setAttribute("scheme", 'resolve');
    $code = $doc->createElement("code");
    $classCode->appendChild($code);

```

```

$code->setAttribute("xml:id", 'resolved');
$code->appendChild($doc->createTextNode('Resolved'));
$code->setAttribute("xml:id", 'unresolved');
$code->appendChild($doc->createTextNode('Unresolved'));
$langUsage = $doc->createElement("langUsage");
$profileDesc->appendChild($langUsage);
for($m=0;$m<count($lang_list);$m++){
    $language = $doc->createElement("language");
    $language->appendChild($doc->createTextNode($langs[$lang_list[$m]]));
    $langUsage->appendChild($language);
    $language->setAttribute("ident", $lang_list[$m]);
}
// Text
$text = $doc->createElement("text");
$tei->appendChild($text);
$body = $doc->createElement("body");
$text->appendChild($body);
$list = $doc->createElement("list");
$body->appendChild($list);
$item = $doc->createElement("item");
$list->appendChild($item);
// SET: lines
// QUERY: SELECT from DSS_lines
$result_lines = mysql_query("SELECT * FROM `DSS_lines` WHERE 1");
while($row_lines = mysql_fetch_array($result_lines)){
    $lines[$row_lines['ID']]['ID']=$row_lines['ID'];
    $lines[$row_lines['ID']]['name']=$row_lines['name'];
    $lines[$row_lines['ID']]['sectionID']=$row_lines['sectionID'];
    $lines[$row_lines['ID']]['afterID']=$row_lines['afterID'];
    $lines[$row_lines['ID']]['beforeID']=$row_lines['beforeID'];
}
foreach ($lines as $line1){
    if($line1['afterID']==0){
        $line = $doc->createElement("l");
        $item->appendChild($line);
        $line->setAttribute("n", $line1['ID']);
        $prev_line_ID = $line1['ID'];
    };
    if($line1['afterID']==$prev_line_ID){
        $line = $doc->createElement("l");
        $item->appendChild($line);
        $line->setAttribute("n", $line1['ID']);
        $prev_line_ID = $line1['ID'];
    };
    // SET: words
    unset ($words);
    // QUERY: SELECT from DSS_words
    $query_words= "SELECT * FROM `DSS_words` WHERE `lineID` = " . $line1['ID'];
    $result_words = mysql_query($query_words);
    while($row_words = mysql_fetch_array($result_words)){
        $words[$row_words['ID']]['ID']=$row_words['ID'];
        $words[$row_words['ID']]['lineID']=$row_words['lineID'];
        $words[$row_words['ID']]['afterID']=$row_words['afterID'];
        $words[$row_words['ID']]['beforeID']=$row_words['beforeID'];
    }
    if($words){
        foreach ($words as $word1){
            if($word1['afterID']==0){
                $word = $doc->createElement("w");
                $line->appendChild($word);
                $word->setAttribute("n", $word1['ID']);
                $prev_word_ID = $word1['ID'];
            }; // end of if($word1['afterID']==0){
            if($word1['afterID']==$prev_word_ID){
                $word = $doc->createElement("w");
                $line->appendChild($word);
                $word->setAttribute("n", $word1['ID']);
                $prev_word_ID = $word1['ID'];
            }; // end of if($word1['afterID']==$prev_word_ID){
            // SET: chars
            unset ($count_chars);
            // QUERY: SELECT from DSS_characters

```

```

$query_chars= "SELECT * FROM `DSS_characters` WHERE `wordID` = " .
$word1['ID'];
$result_chars = mysql_query($query_chars);
$row_chars = mysql_fetch_array($result_chars);
$count_chars=1;
while($row_chars = mysql_fetch_array($result_chars)) {
    $count_chars++;
}; // end of while($row_chars = mysql_fetch_array($result_chars)) {
// QUERY: SELECT from DSS_characters
$query_chars1= "SELECT * FROM `DSS_characters` WHERE `wordID` = " .
$word1['ID'] . " AND `afterID` = 0";
$result_chars1 = mysql_query($query_chars1);
while($row_chars1 = mysql_fetch_array($result_chars1)){
    $char = $doc->createElement("c");
    $word->appendChild($char);
    $char->setAttribute("n", $row_chars1['ID']);
    $prev_char_ID = $row_chars1['ID'];
}; // end of while($row_chars1 = mysql_fetch_array($result_chars1)){
unset($letters);
// QUERY: SELECT from DSS_letters
$query_letters= "SELECT * FROM `DSS_letters` WHERE `charID` = " .
$prev_char_ID;
$result_letters = mysql_query($query_letters);
while($row_letters = mysql_fetch_array($result_letters)){
    $letters[$row_letters['ID']]['ID']=$row_letters['ID'];
    $letters[$row_letters['ID']]['charID']=$row_letters['charID'];
    $letters[$row_letters['ID']]['letter']=$row_letters['letter'];
    $letters[$row_letters['ID']]['resolve']=$row_letters['resolve'];
} // end of while($row_letters = mysql_fetch_array($result_letters)){
if($letters){
    foreach ($letters as $letter1){
        if($letter1['resolve']==1){
            $g = $doc->createElement("g");
            $char->appendChild($g);
            $g->setAttribute("ana", "#resolved");
            $g->appendChild($doc->createTextNode($letter1['letter']));
        }else{
            $g = $doc->createElement("g");
            $char->appendChild($g);
            $g->setAttribute("ana", "#unresolved");
            $g->appendChild($doc->createTextNode($letter1['letter']));
        }; // end if($letter1['resolve']==1){
    }; // end foreach ($letters as $letter1){
}; // end if($letters){
for($bb=0;$bb<=$count_chars;$bb++){
    // QUERY: SELECT from DSS_characters
    $query_chars2= "SELECT * FROM `DSS_characters` WHERE `wordID` = " .
$word1['ID'] . " AND `afterID` = " . $prev_char_ID;
    $result_chars2 = mysql_query($query_chars2);
    while($row_chars2 = mysql_fetch_array($result_chars2)){
        $char = $doc->createElement("c");
        $word->appendChild($char);
        $char->setAttribute("n", $row_chars2['ID']);
        $prev_char_ID = $row_chars2['ID'];
        unset($letters);
        // QUERY: SELECT from DSS_letters
        $query_letters= "SELECT * FROM `DSS_letters` WHERE `charID` = " .
$prev_char_ID;
        $result_letters = mysql_query($query_letters);
        while($row_letters = mysql_fetch_array($result_letters)){
            $letters[$row_letters['ID']]['ID']=$row_letters['ID'];
            $letters[$row_letters['ID']]['charID']=$row_letters['charID'];
            $letters[$row_letters['ID']]['letter']=$row_letters['letter'];

            $letters[$row_letters['ID']]['resolve']=$row_letters['resolve'];
        } // end of while($row_letters =
mysql_fetch_array($result_letters)){
    if($letters){
        foreach ($letters as $letter1){
            if($letter1['resolve']==1){
                $g = $doc->createElement("g");
                $char->appendChild($g);
                $g->setAttribute("ana", "#resolved");

```

```

        $g->appendChild($doc-
>createTextNode($letter1['letter']));
    }else{
        $g = $doc->createElement("g");
        $char->appendChild($g);
        $g->setAttribute("ana", "#unresolved");
        $g->appendChild($doc-
>createTextNode($letter1['letter']));
    }; // end if($letter1['resolve']==1){
    }; // end foreach ($letters as $letter1){
    }; // end if($letters){
    }; // end of while($row_chars2 = mysql_fetch_array($result_chars2)){
    }; // end for($bb=0;$bb<=$count_chars;$bb++){
    }; // end foreach ($words as $word1){
    }; // end if($words){
    }; // end foreach ($lines as $line1){
    //Return status failed, if any error found.
    if ($error_code == 1) {
        $ab7 = $doc->createElement("ab");
        $ab7->appendChild($doc->createTextNode('failed'));
        $encodingDesc->appendChild($ab7);
        $ab7->setAttribute("type", "status");
    }
    // Return status passed, if no error found
    else if ($error_code == 0) {
        $ab7 = $doc->createElement("ab");
        $ab7->appendChild($doc->createTextNode('passed'));
        $encodingDesc->appendChild($ab7);
        $ab7->setAttribute("type", "status");
    }
    // OUTPUT: XML string
    return simplexml_load_string($doc->saveXML());
} //end function DBtoTEI() {
// RUN: Web Service function
$server = new Zend_Rest_Server();
$server->addFunction('DBtoTEI');
$server->handle();?>

```

```
<?php
```

```

/**
 * xml.php
 *
 * Processing script for the js functions showEng(),
 *
 * @package    DSS prototype
 * @author     Henriette Roued-Cunliffe <henriette@roued.com>
 * @license    for OS license
 * @link       http://roued.com/e-doc
 * @since      File available since 2011
 */

// FILE: config.php
include_once('config.php');
// Get some variables
$do= $_REQUEST['do'];
$add= $_REQUEST['add'];
$get= $_REQUEST['get'];
$del= $_REQUEST['del'];
$sug= $_REQUEST['sug'];
$text= $_REQUEST['text'];
$index= $_REQUEST['index'];
$pattern= $_REQUEST['pattern'];
$type= $_REQUEST['type'];
$xml = new DOMDocument();
// FILE: db-tei.php
$xml->load($DBtoTEI);
// CASE: type of actions
switch($do){

// CASE: get actions

```

```

case "get":
    switch($get){

        // CASE: SELECT records from DSS_engines
        case "eng":
            // FUNCTION: showEng(get=eng)
            $showeng="xml.php?do=get&get=eng";
            $showeng_code = 'showEng("'.$showeng.'")';
            // QUERY: SELECT from DSS_engines
            $query_eng = "SELECT * FROM `DSS_engines` ORDER BY
`DSS_engines`.`suggestionType` ASC;";
            $result_eng = mysql_query($query_eng);
            // OUTPUT: search engine view
            echo '<h1>Search engines</h1>';
            echo '<ul>';
            while($row_eng = mysql_fetch_array($result_eng)){
                echo '<li>';
                // IF: engine is used
                if($row_eng['used']==1){
                    // FUNCTION: doSomething(del=eng)
                    $do_unresolve="xml.php?do=del&del=eng";
                    $do_unresolve_code =
'doSomething("'.$do_unresolve.'",'.$row_eng['ID'].')';
                    // OUTPUT: checked radio-button
                    echo "<form><label>";
                    echo $row_eng['title'];
                    echo "</label><input type='radio' name='used' value='resolved'
checked='checked' onclick='\".$do_unresolve_code.view();\".$showeng_code.\"'/> <br
/></form>";
                    // IF: engine is not used
                }else{
                    // FUNCTION: doSomething (add=eng)
                    $do_resolve="xml.php?do=add&add=eng";
                    $do_resolve_code =
'doSomething("'.$do_resolve.'",'.$row_eng['ID'].')';
                    // OUTPUT: un-checked radio-button
                    echo "<form><label>";
                    echo $row_eng['title'];
                    echo "</label><input type='radio' name='unused' value='resolved'
onclick='\".$do_resolve_code.view();\".$showeng_code.\"'/> <br /></form>";
                }; // end of if($row_eng['used']==1){
                echo '</li>';
            } // end of while($row_eng = mysql_fetch_array($result_eng)){
            echo '</ul>';
            break;

            // CASE: SELECT records from DSS_letters
            case "char":
                // FUNCTION: showTop (get=char)
                $showtop="xml.php?do=get&get=char";
                $showtop_code = 'showTop("'.$showtop.'",'.$index.')';
                // QUERY: SELECT from DSS_letters
                $query_char = "SELECT * FROM `DSS_letters` WHERE `charID` = " . $index .
";";
                $result_char = mysql_query($query_char);
                while($row_char = mysql_fetch_array($result_char)){
                    $ischar[] = $row_char['ID'];
                    $letters[$row_char['ID']]['ID']=$row_char['ID'];
                    $letters[$row_char['ID']]['charID']=$row_char['charID'];
                    $letters[$row_char['ID']]['letter']=$row_char['letter'];
                    $letters[$row_char['ID']]['resolve']=$row_char['resolve'];
                } // end of while($row_char = mysql_fetch_array($result_char)){
                // OUTPUT: letter pop-up view
                echo '<h1>Letters</h1>';
                if($letters){
                    foreach ($letters as $letter1){
                        // FUNCTION: doSomething(del=letter)
                        $do_delleter="xml.php?do=del&del=letter";
                        $do_delleter_code =
'doSomething("'.$do_delleter.'",'.$letter1['ID'].')';
                        // IF: letter is resolved
                        if($letter1['resolve']==1){
                            // OUTPUT letter text and delete button

```



```

        echo "Letter is <em>".$letter1['letter'].</em> <button
type='button' onclick='".$do_delletter_code."view():".$showtop_code."'>-
</button><br/>";
        // FUNCTION: doSomething(del=resolve)
        $do_unresolve="xml.php?do=del&del=resolve";
        $do_unresolve_code =
'doSomething('".$do_unresolve."','".$letter1['ID'].')';
        // OUTPUT: checked radio-button
        echo "<form><input type='radio' name='resolve' value='resolved'
checked='checked' onclick='".$do_unresolve_code."view():".$showtop_code."'>/>
resolved <br /></form>";
        // IF: letter is not resolved
    }else{
        // OUTPUT letter text and delete button
        echo "Letter could be <em>".$letter1['letter'].</em> <button
type='button' onclick='".$do_delletter_code."view():".$showtop_code."'>-
</button><br/>";
        // FUNCTION: doSomething(add=resolve)
        $do_resolve="xml.php?do=add&add=resolve&charID=".$index;
        $do_resolve_code =
'doSomething('".$do_resolve."','".$letter1['ID'].')';
        // OUTPUT: un-checked radio-button
        echo "<form><input type='radio' name='resolve' value='resolved'
onclick='".$do_resolve_code.'">resolved <br />
</form>";
    }; // end if($letter1['resolve']==1){
        // FUNCTION: addText(add=lettertext)
        $do_letter="xml.php?do=add&add=lettertext";
        $do_letter_code = 'addText('".$do_letter.'"','".$letter1['ID'].',this.value)';
        // OUTPUT: letter edit form
        echo "<form><label><strong>Edit letter: </strong></label><input
name='text' type='text' id='text' maxlength='1' size='1'
onKeyPress='if(enter_pressed(event)){".$do_letter_code."view():".$showtop_code."}'
'></input><br/><br/>";
        // FUNCTION: showArg(get=arg)
        $showarg="xml.php?do=get&get=arg&type=letter";
        $showarg_code = 'showArg('".$showarg."','".$letter1['ID'].')';
        // OUTPUT: 'Show Arguments' button
        echo "<button type='button' onclick='".$showarg_code."' >Show
Arguments</button></form>";
        echo '<hr/>';
    }; // end foreach ($letters as $letter1){
}; // end if($letters){
// FUNCTION: doSomething(add=letter)
$do_addletter="xml.php?do=add&add=letter";
$do_addletter_code = 'doSomething('".$do_addletter."','".$index.')';
// OUTPUT: '+add a letter' button
echo "<form><button type='button'
onclick='".$do_addletter_code."view():".$showtop_code."' >+ Add a
letter</button></form><br/><br/>";
    echo "Only resolve one letter please!";
    break;
// CASE: SELECT records from DSS_arguments
case "arg":
    // FUNCTION: showArg(get=arg)
    $showarg="xml.php?do=get&get=arg&type=letter";
    $showarg_code = 'showArg('".$showarg."','".$index.')';
    // QUERY SELECT from DSS_arguments
    $query_arg = "SELECT * FROM `DSS_arguments` WHERE `type` LIKE '" . $type .
    "' AND `typeID` = " . $index . " ";
    $result_arg = mysql_query($query_arg);
    while($row_arg = mysql_fetch_array($result_arg)){
        $args[$row_arg['ID']]['ID']=$row_arg['ID'];
        $args[$row_arg['ID']]['text']=$row_arg['text'];
        $args[$row_arg['ID']]['boolean']=$row_arg['boolean'];
        $args[$row_arg['ID']]['type']=$row_arg['type'];
        $args[$row_arg['ID']]['typeID']=$row_arg['typeID'];
    }
    // OUTPUT arguments pop-up view
    echo '<h1>Arguments</h1>';
    // IF: there are any arguments
    if($args){

```

```

foreach ($args as $arg1){
    // FUNCTION: doSomething(del=arg)
    $do_del="xml.php?do=del&del=arg";
    $do_del_code = 'doSomething("'. $do_del.'",'. $arg1['ID'].')';
    // IF: argument is for
    if($arg1['boolean']==1){
        // FUNCTION:doSomething(add=against)
        $do_against="xml.php?do=add&add=against";
        $do_against_code = 'doSomething("'.
$do_against.'",'. $arg1['ID'].')';
        // OUTPUT checked radio-button
        echo "<form><input type='radio' name='for' value='for'
checked='checked' onclick='". $do_against_code."". $showarg_code."/> argument for
<br /></form>";
        // OUTPUT '+' and argument text and delete button
        echo '+ ' . $arg1['text'] . "<button type='button'
onclick='". $do_del_code."". $showarg_code.">-</button><br/>";
    }else{
        // FUNCTION: doSomething(add=for)
        $do_for="xml.php?do=add&add=for";
        $do_for_code = 'doSomething("'. $do_for.'",'. $arg1['ID'].')';
        //OUTPUT: un-checked radio-button
        echo "<form><input type='radio' name='for' value='for'
onclick='". $do_for_code."". $showarg_code."/> argument for <br /></form>";
        // OUTPUT '-' and argument text and delete button
        echo '- ' . $arg1['text'] . "<button type='button'
onclick='". $do_del_code."". $showarg_code.">-</button><br/>";
    }; // end if($arg1['boolean']==1){
    // FUNCTION: addText(add=argtext)
    $do_text="xml.php?do=add&add=argtext";
    $do_text_code = 'addText("'.
$do_text.'",'. $arg1['ID'].',this.value)';
    // OUTPUT: text edit form
    echo "<form onsubmit='return false;'><label><strong>Edit text:
</strong></label><input value='". $arg1['text']. "' name='text' type='text'
id='text' maxlength='500'
onKeyPress='if(enter_pressed(event)){". $do_text_code."". $showarg_code."}'></input
><br/></form><br/>";

    // --- Links to same documents --- //
    // QUERY: SELECT from DSS_linkToDoc
    unset($linkdocs);
    $query_linkdoc = "SELECT * FROM `DSS_linkToDoc` WHERE `argumentID` ="
. $arg1['ID'] . " ";
    $result_linkdoc = mysql_query($query_linkdoc);
    while($row_linkdoc = mysql_fetch_array($result_linkdoc)){
        $linkdocs[$row_linkdoc['ID']]['ID']=$row_linkdoc['ID'];

        $linkdocs[$row_linkdoc['ID']]['argumentID']=$row_linkdoc['argumentID'];
        $linkdocs[$row_linkdoc['ID']]['linkType']=$row_linkdoc['linkType'];
        $linkdocs[$row_linkdoc['ID']]['linkID']=$row_linkdoc['linkID'];
    }
    // IF: there are any document links
    if($linkdocs){
        foreach ($linkdocs as $linkdoc1){
            // FUNCTION: doSomething(del=doclink)
            $do_deldoclink="xml.php?do=del&del=doclink";
            $do_deldoclink_code = 'doSomething("'.
$do_deldoclink.'",'. $linkdoc1['ID'].')';
            // FILE: db-tei.php
            $xmllsimple = simplexml_load_file($DBtoTEI);
            switch($linkdoc1['linkType']){
                // CASE: link to letters
                case 'letter':
                    $pathletter = '//c[@n="'. $linkdoc1['linkID']. '"]';
                    $letters=$xmllsimple->xpath($pathletter);
                    unset($letter2);
                    foreach($letters as $letter){
                        $letter2 = $letter;
                        foreach ($letter->children() as $child1){
                            $letter2 = $child1;
                        }; // end of foreach ($letter->children() as $child1){
                    }; // end of foreach($letters as $letter){

```

```

        // OUTPUT: link to letter text
        echo "This argument is linked to the letter: <em>".$letter2
. "</em> <button type='button'
onclick='\".$do_deldoclink_code.\"\".$showarg_code.\"'-</button><br/>";
        break;
        // CASE: link to words
        case 'word':
            // FILE: get_word.xsl
            $xsl_name = 'get_word.xsl';
            $xml->load($DBtoTEI);
            $xsl = new DOMDocument();
            $xsl->load($xsl_name);
            $proc = new XsltProcessor();
            $xsl = $proc->importStylesheet($xsl);
            $proc->setParameter(null, 'index', $linkdoc1['linkID']);
            $newdom = $proc->transformToDoc($xml);
            $edition = $newdom->saveXML();
            $word2 = $edition;
            // OUTPUT: link to word text
            echo "This argument is linked to the word: <em>".$word2 .
"</em> <button type='button' onclick='\".$do_deldoclink_code.\"\".$showarg_code.\"'->-
</button><br/>";
        break;
        // CASE: default
        default:
            // OUTPUT: default encouragement to link
            echo "Link this argument to a word or letter <button
type='button' onclick='\".$do_deldoclink_code.\"\".$showarg_code.\"'->-
</button><br/>";
        break;
    }; // end of switch($linkdoc1['linkType']){
unset($params);
switch($linkdoc1['linkType']){
    // CASE: set parameters if linkType is letter
    case 'letter':
        $params['letter'] = $linkdoc1['linkID'];
        break;
    // CASE: set parameters if linkType is word
    case 'word':
        $params['word'] = $linkdoc1['linkID'];
        break;
}; // end switch($linkdoc1['linkType']){
// SET parameters
$params['id']=$linkdoc1['ID'];
$params['index']=$index;
// FILE: viewlink.xsl
$xsl_name = 'viewlink.xsl';
$xsl = new DOMDocument();
$xsl->load($xsl_name);
$proc = new XsltProcessor();
$xsl = $proc->importStylesheet($xsl);
$proc->setParameter('', $params);
$newdom = $proc->transformToDoc($xml);
$view = $newdom->saveXML();
// OUTPUT: mini views
echo $view;
echo '<br/><br/>';
} // end foreach ($linkdocs as $linkdoc1){
} // end if($linkdocs){
// FUNCTION: doSomething(add=doclink)
$do_adddoc="xml.php?do=add&add=doclink";
$do_adddoc_code = 'doSomething('\". $do_adddoc.\"\", '$arg1['ID'].')';
// OUTPUT: '+Add Document Link' button
echo "<form><button type='button'
onclick='\".$do_adddoc_code.\"\".$showarg_code.\"' >+ Add Document
Link</button></form><br/>";

// --- Links to external Webpages --- //
unset($linkurls);
// QUERY: SELECT from DSS_linkToURL
$query_linkurl = "SELECT * FROM `DSS_linkToURL` WHERE `argumentID` ="
. $arg1['ID'] . " ";
$result_linkurl = mysql_query($query_linkurl);

```

```

while($row_linkurl = mysql_fetch_array($result_linkurl)){
    $linkurls[$row_linkurl['ID']]['ID']=$row_linkurl['ID'];

    $linkurls[$row_linkurl['ID']]['argumentID']=$row_linkurl['argumentID'];
    $linkurls[$row_linkurl['ID']]['URL']=$row_linkurl['URL'];
    $linkurls[$row_linkurl['ID']]['URLtitle']=$row_linkurl['URLtitle'];
}

if($linkurls){
    foreach ($linkurls as $linkurl1){
        // FUNCTION: addText(add=urltext)
        $do_url="xml.php?do=add&add=urltext";
        $do_url_code = 'addText("'.
$do_url.'",'. $linkurl1['ID'].',this.value);';
        // FUNCTION: addText(add=urltitle)
        $do_urltitle="xml.php?do=add&add=urltitle";
        $do_urltitle_code = 'addText("'.
$do_urltitle.'",'. $linkurl1['ID'].',this.value);';
        // FUNCTION: doSomething(del=url)
        $do_delurl="xml.php?do=del&del=url";
        $do_delurl_code = 'doSomething("'.
$do_delurl.'",'. $linkurl1['ID'].')';
        // OUTPUT: url link editing
        echo "<strong>URL: </strong><a target='_blank'
href='". $linkurl1['URL']. "' title='". $linkurl1['URLtitle']. "'> .
$linkurl1['URLtitle'] . "</a> <button type='button'
onclick='". $do_delurl_code . "'></button><br/>";
        echo "<form><label><strong>Edit URL: </strong></label><input
onKeyPress='return disableEnterKey(event)' value='". $linkurl1['URL']. "'
name='text' tabindex='0' type='text' id='text' maxlength='200'
onChange='". $do_url_code . "'></input><br/><label><strong>Edit
Title: </strong></label><input onKeyPress='return disableEnterKey(event)'
value='". $linkurl1['URLtitle']. "' name='text' tabindex='0' type='text' id='text'
maxlength='200'
onChange='". $do_urltitle_code . "'></input><br/></form><br/>";

    } // end foreach ($linkurls as $linkurl1){
} // end if($linkurls){
// FUNCTION: doSomething(add=url)
$do_addlink="xml.php?do=add&add=url";
$do_addlink_code = 'doSomething("'. $do_addlink.'",'. $arg1['ID'].')';
// OUTPUT: '+Add URL' button
echo "<form><button type='button'
onclick='". $do_addlink_code . "'>+ Add
URL</button></form><br/>";

// --- Links to external bibliographic references --- //
unset($linkbibl);
// QUERY: SELECT from DSS_linkToBibl
$query_linkbibl = "SELECT * FROM `DSS_linkToBibl` WHERE `argumentID`
=" . $arg1['ID'] . ";";
$result_linkbibl = mysql_query($query_linkbibl);

while($row_linkbibl = mysql_fetch_array($result_linkbibl)){
    $linkbibls[$row_linkbibl['ID']]['ID']=$row_linkbibl['ID'];

    $linkbibls[$row_linkbibl['ID']]['argumentID']=$row_linkbibl['argumentID'];
    $linkbibls[$row_linkbibl['ID']]['biblURL']=$row_linkbibl['biblURL'];
    $linkbibls[$row_linkbibl['ID']]['biblRef']=$row_linkbibl['biblRef'];
}
if($linkbibls){
    foreach ($linkbibls as $linkbibl1){
        // FUNCTION: addText(add=bibltext)
        $do_bibl="xml.php?do=add&add=bibltext";
        $do_bibl_code = 'addText("'.
$do_bibl.'",'. $linkbibl1['ID'].',this.value);';
        // FUNCTION: addText(add=bibltitle)
        $do_bibltitle="xml.php?do=add&add=bibltitle";
        $do_bibltitle_code = 'addText("'.
$do_bibltitle.'",'. $linkbibl1['ID'].',this.value);';
        // FUNCTION: doSomething(del=bibl)
        $do_delbibl="xml.php?do=del&del=bibl";

```

```

        $do_delbibl_code = 'doSomething(''.
$do_delbibl.'', '$linkbibl1['ID'].')';';
        // OUTPUT: bibliographic reference link editing
        echo "<strong>Bibliographic URL: </strong><a target='_blank'
href='\".$linkbibl1['biblURL'].\"' title='\".$linkbibl1['biblRef'].\"'>\" .
$linkbibl1['biblRef'] . \"</a> <button type='button'
onclick='\".$do_delbibl_code.\".\".$showarg_code.\"'></button><br/>";
        echo "<form><label><strong>Edit Bibliographic URL:
</strong></label><input onKeyPress='return disableEnterKey(event)'
value='\".$linkbibl1['biblURL'].\"' name='text' tabindex='0' type='text' id='text'
maxlength='200' onchange='\".$do_bibl_code.\".\".$showarg_code.\"'></input><br/>
<label><strong>Edit Bibliographic Reference: </strong></label><input
onKeyPress='return disableEnterKey(event)' value='\".$linkbibl1['biblRef'].\"'
name='text' tabindex='0' type='text' id='text' maxlength='200'
onchange='\".$do_bibltitle_code.\".\".$showarg_code.\"'></input><br/></form><br/>";

        } // end foreach ($linkurls as $linkurl1){
    } // end if($linkurls){
    // FUNCTION: doSomething(add=bibl)
    $do_addbibl="xml.php?do=add&add=bibl";
    $do_addbibl_code = 'doSomething(''. $do_addbibl.'', '$arg1['ID'].')';';
    //OUTPUT: '+ Add Bibliographic Reference' button
    echo "<form><button type='button'
onclick='\".$do_addbibl_code.\".\".$showarg_code.\"' >+ Add Bibliographic
Reference</button></form><br/>";
    // OUTPUT: horizontal line
    echo '<hr/>';
    }; // end foreach ($args as $arg1){
}; // end if($args){
// FUNCTION: doSomething(add=arg)
$do_addarg="xml.php?do=add&add=arg";
$do_addarg_code = 'doSomething(''. $do_addarg.'', '$index.~')';';
// OUTPUT: '+ Add an argument' button
echo "<form><button type='button'
onclick='\".$do_addarg_code.\".\".$showarg_code.\"' >+ Add an
argument</button></form><br/><br/>";
break;

// CASE: get words and suggest words
case "word":
    // FILE: get_word.xml
    $xsl_name = 'get_word.xml';
    // FILE: db-tei.php
    $xml->load($DBtoTEI);
    $xsl = new DOMDocument();
    $xsl->load($xsl_name);
    $proc = new XsltProcessor();
    $xsl = $proc->importStylesheet($xsl);
    // SET: parameters
    $proc->setParameter(null, 'index', $index);
    $newdom = $proc->transformToDoc($xml);
    $edition = $newdom->saveXML();
    $word2 = $edition;
    // OUTPUT the word
    echo '<h1>Word</h1>';
    echo "The word is <em>\".$word2 . \"</em><br/>";
    // FILE: get_sugword.xml
    $xsl_name_sug = 'get_sugword.xml';
    // FILE: db-tei.php
    $xml->load($DBtoTEI);
    $xsl = new DOMDocument();
    $xsl->load($xsl_name_sug);
    $proc = new XsltProcessor();
    $xsl = $proc->importStylesheet($xsl);
    // SET: parameters
    $proc->setParameter(null, 'index', $index);
    $newdom = $proc->transformToDoc($xml);
    foreach($newdom->childNodes as $node){
        $sug_word .= $newdom->saveXML($node);
        // FUNCTION: showSug(sug=word)
        $showsug="xml.php?do=sug&sug=word";
        $showsug_code = 'showSug(''. $showsug.'', '$sug_word.~')';';
        // FUNCTION: showSug (sug=lexicon)

```

```

        $showlex="xml.php?do=sug& sug=lexicon";
        $showlex_code = 'showSug("'.$showlex.'", "'.$sug_word.'");';
        // OUTPUT: 'search parallels' button
        echo "<form><button type='button' onclick='\"".$showsug_code.\"' >Search
parallels for: \".$sug_word.\"</button></form><br/><br/>";
        // OUTPUT: 'search lexicon' button
        echo "<form><button type='button' onclick='\"".$showlex_code.\"' >Search
lexicon for: \".$sug_word.\"</button></form><br/><br/>";
    }
    break;
}
break;

// CASE: add actions
case "add":
    switch($add){
        // CASE: UPDATE 'used = 1' in DSS_engines
        case "eng":
            // QUERY: UPDATE 'used = 1' in DSS_engines
            $add_eng_query = "UPDATE `DSS_\".$document.\"`.`DSS_engines` SET `used` =
'1' WHERE `DSS_engines`.`ID` = \" . $index . \"";
            mysql_query($add_eng_query);
            break;
            // CASE: INSERT a record in DSS_letters
        case "letter":
            // QUERY: INSERT a record in DSS_letters
            $add_letter_query = "INSERT INTO `DSS_\".$document.\"`.`DSS_letters` (`ID`,
`charID`, `letter`, `resolve`) VALUES (NULL, \"'.$index.'\", '', '0')";
            mysql_query($add_letter_query);
            break;
            // CASE: UPDATE 'resolve = 1' in DSS_letters
        case "resolve":
            // SET: $charID
            $charID= $_REQUEST['charID'];
            // QUERY: SELECT from DSS_letters
            $query_resolve = "SELECT * FROM `DSS_letters` WHERE `resolve` = 1 AND
`charID` = \" . $charID . \"";
            $result_resolve = mysql_query($query_resolve);
            while($row_resolve = mysql_fetch_array($result_resolve))
            {
                $is_resolve = $row_resolve['ID'];
            }
            // IF: no letter is already resolved
            if($is_resolve==''){
                // QUERY: UPDATE 'resolve = 1' in DSS_letters
                $add_resolve_query = "UPDATE `DSS_letters` SET `resolve` = 1 WHERE
`ID` = \" . $index . \"";
                mysql_query($add_resolve_query);
            };
            break;
            // CASE: UPDATE 'letter' in DSS_letters
        case "lettertext":
            // QUERY: UPDATE 'letter' in DSS_letters
            $add_lettertext_query = "UPDATE `DSS_\".$document.\"`.`DSS_letters` SET
`letter` = \"'\" . $text . \"'\" WHERE `DSS_letters`.`ID` = \" . $index . \"";
            mysql_query($add_lettertext_query);
            break;
            // CASE: INSERT a record into DSS_arguments
        case "arg":
            // QUERY: INSERT a record into DSS_arguments
            $add_arg_query = "INSERT INTO `DSS_\".$document.\"`.`DSS_arguments` (`ID`,
`text`, `boolean`, `type`, `typeID`) VALUES (NULL, '\".$index.\"'\", '1', 'letter',
'\".$index.\"')";
            mysql_query($add_arg_query);
            break;
            // CASE: UPDATE field 'text' in DSS_arguments
        case "argtext":
            // QUERY: UPDATE field 'text' in DSS_arguments
            $add_argtext_query = "UPDATE `DSS_\".$document.\"`.`DSS_arguments` SET
`text` = \"'\" . $text . \"'\" WHERE `DSS_arguments`.`ID` = \" . $index . \"";
            mysql_query($add_argtext_query);
            break;
            // CASE: UPDATE 'boolean = 0' in DSS_arguments

```

```

case "against":
    // QUERY: UPDATE 'boolean = 0' in DSS_arguments
    $add_against_query = "UPDATE `DSS_`. $document."`.`DSS_arguments` SET
`boolean` = '0' WHERE `DSS_arguments`.`ID` = " . $index . " ";";
    mysql_query($add_against_query);
    break;
// CASE: UPDATE 'boolean = 1' in DSS_arguments
case "for":
    // QUERY: UPDATE 'boolean = 1' in DSS_arguments
    $add_for_query = "UPDATE `DSS_`. $document."`.`DSS_arguments` SET
`boolean` = '1' WHERE `DSS_arguments`.`ID` = " . $index . " ";";
    mysql_query($add_for_query);
    break;
// CASE: INSERT a record in DSS_linkToURL
case "url":
    // QUERY: INSERT a record in DSS_linkToURL
    $add_url_query = "INSERT INTO `DSS_`. $document."`.`DSS_linkToURL` (`ID`,
`argumentID`, `URL`, `URLtitle`) VALUES (NULL, '$index.', '', '');"
    mysql_query($add_url_query);
    break;
// CASE: UPDATE field 'URL' in DSS_linkToURL
case "urltext":
    // QUERY: UPDATE field 'URL' in DSS_linkToURL
    $add_urltext_query = "UPDATE `DSS_`. $document."`.`DSS_linkToURL` SET
`URL` = '$text.'" WHERE `DSS_linkToURL`.`ID` = " . $index . " ";";
    mysql_query($add_urltext_query);
    break;
// CASE: UPDATE field 'URLtitle' in DSS_linkToURL
case "urltitle":
    // QUERY: UPDATE field 'URLtitle' in DSS_linkToURL
    $add_urltitle_query = "UPDATE `DSS_`. $document."`.`DSS_linkToURL` SET
`URLtitle` = '$text.'" WHERE `DSS_linkToURL`.`ID` = " . $index . " ";";
    mysql_query($add_urltitle_query);
    break;
// CASE: INSERT a record in DSS_linkToBibl
case "bibl":
    // QUERY: INSERT a record in DSS_linkToBibl
    $add_bibl_query = "INSERT INTO `DSS_`. $document."`.`DSS_linkToBibl`
(`ID`, `argumentID`, `biblURL`, `biblRef`) VALUES (NULL, '$index.', '', '');"
    mysql_query($add_bibl_query);
    break;
// CASE: UPDATE field 'biblURL' in DSS_linkToBibl
case "bibltext":
    // QUERY: UPDATE field 'biblURL' in DSS_linkToBibl
    $add_bibltext_query = "UPDATE `DSS_`. $document."`.`DSS_linkToBibl` SET
`biblURL` = '$text.'" WHERE `DSS_linkToBibl`.`ID` = " . $index . " ";";
    mysql_query($add_bibltext_query);
    break;
// CASE: UPDATE field 'biblRef' in DSS_linkToBibl
case "bibltitle":
    // QUERY: UPDATE field 'biblRef' in DSS_linkToBibl
    $add_bibltitle_query = "UPDATE `DSS_`. $document."`.`DSS_linkToBibl` SET
`biblRef` = '$text.'" WHERE `DSS_linkToBibl`.`ID` = " . $index . " ";";
    mysql_query($add_bibltitle_query);
    break;
// CASE: INSERT a record in DSS_linkToDoc
case "doclink":
    // QUERY: INSERT a record in DSS_linkToDoc
    $add_doclink_query = "INSERT INTO `DSS_`. $document."`.`DSS_linkToDoc`
(`ID`, `argumentID`, `linkType`, `linkID`) VALUES (NULL, '$index.', '', '');"
    mysql_query($add_doclink_query);
    break;
// CASE: UPDATE 'linkType = word' in DSS_linkToDoc
case "linkword":
    // QUERY: UPDATE 'linkType = word' in DSS_linkToDoc
    $add_linkword_query = "UPDATE `DSS_`. $document."`.`DSS_linkToDoc` SET
`linkType` = 'word', `linkID` = '$index.'" WHERE `DSS_linkToDoc`.`ID` = " .
$type . " ";";
    mysql_query($add_linkword_query);
    break;
// CASE: UPDATE 'linkType = letter' in DSS_linkToDoc
case "linkletter":
    // QUERY: UPDATE 'linkType = letter' in DSS_linkToDoc

```

```

        $add_linkletter_query = "UPDATE `DSS_`.$document.`.`DSS_linkToDoc` SET
`linkType` = 'letter', `linkID` = '". $index.'" WHERE `DSS_linkToDoc`.`ID` = " .
$type . " ";";
        mysql_query($add_linkletter_query);
        break;
    }
break;

// CASE: del actions
case "del":
    switch($del){
        // CASE: UPDATE 'used = 0' in DSS_engines
        case "eng":
            // QUERY: UPDATE 'used = 0' in DSS_engines
            $del_eng_query = "UPDATE `DSS_`.$document.`.`DSS_engines` SET `used` = 0
WHERE `ID` = " . $index . " ";";
            mysql_query($del_eng_query);
            break;
        // CASE: DELETE a record in DSS_letters
        case "letter":
            // QUERY: DELETE a record in DSS_letters
            $del_char_query = "DELETE FROM `DSS_`.$document.`.`DSS_letters` WHERE
`DSS_letters`.`ID` = " . $index . " ";";
            mysql_query($del_char_query);
            break;
        // CASE: UPDATE 'resolve = 0' in DSS_letters
        case "resolve":
            // QUERY: UPDATE 'resolve = 0' in DSS_letters
            $del_resolve_query = "UPDATE `DSS_`.$document.`.`DSS_letters` SET
`resolve` = 0 WHERE `ID` = " . $index . " ";";
            mysql_query($del_resolve_query);
            break;
        // CASE: DELETE a record from DSS_linkToURL
        case "arg":
            // QUERY: DELETE a record from DSS_linkToURL
            $del_arg_query = "DELETE FROM `DSS_`.$document.`.`DSS_arguments` WHERE
`DSS_arguments`.`ID` = " . $index . " ";";
            mysql_query($del_arg_query);
            break;
        // CASE: DELETE a record from DSS_linkToURL
        case "url":
            // QUERY: DELETE a record from DSS_linkToURL
            $del_url_query = "DELETE FROM `DSS_`.$document.`.`DSS_linkToURL` WHERE
`DSS_linkToURL`.`ID` = " . $index . " ";";
            mysql_query($del_url_query);
            break;
        // CASE: DELETE a record from DSS_linkToBibl
        case "bibl":
            // QUERY: DELETE a record from DSS_linkToBibl
            $del_bibl_query = "DELETE FROM `DSS_`.$document.`.`DSS_linkToBibl` WHERE
`DSS_linkToBibl`.`ID` = " . $index . " ";";
            mysql_query($del_bibl_query);
            break;
        // CASE: DELETE a record from DSS_linkToDoc
        case "doclink":
            // QUERY: DELETE a record from DSS_linkToDoc
            $del_doclink_query = "DELETE FROM `DSS_`.$document.`.`DSS_linkToDoc`
WHERE `DSS_linkToDoc`.`ID` = " . $index . " ";";
            mysql_query($del_doclink_query);
            break;
    }
break;

// CASE: suggest actions
case "sug":
    switch($sug){
        // CASE: getting suggested words from concordances
        case "word":
            // SET: pattern to lower-case
            $low_pattern = strtolower($pattern);
            // FILE: sug_word.xsl
            $xsl_name = 'sug_word.xsl';

```



```

// QUERY: SELECT concordances from DSS_engines
$query_appello = "SELECT * FROM `DSS_engines` WHERE `used` = 1 AND
`suggestionType` LIKE 'concordance'";
$result_appello = mysql_query($query_appello);
while($row_appello = mysql_fetch_array($result_appello)){
    // SET: XML Web Service to search
    $webser_url = $row_appello['webservice'];
    $sug_word = $webser_url . $low_pattern;
    $xml->load($sug_word);
    $xsl = new DOMDocument();
    $xsl->load($xsl_name);
    $proc = new XsltProcessor();
    $xsl = $proc->importStylesheet($xsl);
    // SET: parameter
    $proc->setParameter(null, 'documentws', $row_appello['document']);
    $newdom = $proc->transformToDoc($xml);
    $edition .= $newdom->saveXML();
}
// OUTPUT: list of suggested words from selected concordance engines
echo $edition;
break;
// CASE: getting suggested words from lexicons
case "lexicon":
    // SET: pattern to lower-case
    $low_pattern = strtolower($pattern);
    // FILE: sug_lexicon.xsl
    $xsl_name = 'sug_lexicon.xsl';
    // QUERY: SELECT lexicons from DSS_engines
    $query_appello = "SELECT * FROM `DSS_engines` WHERE `used` = 1 AND
`suggestionType` LIKE 'lexicon'";
    $result_appello = mysql_query($query_appello);
    while($row_appello = mysql_fetch_array($result_appello)){
        // SET: XML Web Service to search
        $webser_url = $row_appello['webservice'];
        $sug_word = $webser_url . $low_pattern;
        $xml->load($sug_word);
        $xsl = new DOMDocument();
        $xsl->load($xsl_name);
        $proc = new XsltProcessor();
        $xsl = $proc->importStylesheet($xsl);
        // SET: parameter
        $proc->setParameter(null, 'documentws', $row_appello['document']);
        $newdom = $proc->transformToDoc($xml);
        $edition .= $newdom->saveXML();
    }
    // OUTPUT: list of suggested words from selected lexicon engines
    echo $edition;
break;
}
break;
}
?>

```

```

<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<!--
* viewlink.xsl
*
* XSLT stylesheet for the js function showArg()
*
* @package    DSS prototype
* @author     Henriette Roued-Cunliffe <henriette@roued.com>
* @license    for OS license
* @link       http://roued.com/e-doc
* @since      File available since 2011
-->
<xsl:param name="letter"/>
<xsl:param name="word"/>
<xsl:param name="id"/>
<xsl:param name="index"/>
<xsl:strip-space elements="*" />

```

```

<xsl:template match="/">
  <xsl:apply-templates select="//item"/>
</xsl:template>

<xsl:template match="//item">
  <div class="link-view">
    <xsl:choose>
      <xsl:when test="//1">
        <xsl:apply-templates select="//1"/>
      </xsl:when>
    </xsl:choose>
  </div>
</xsl:template>

<xsl:template match="//1">
  <div class="link-line"><span class="link-linenr">Line <xsl:value-of
select="@n"/>: </span>
  <xsl:apply-templates select="w"/>
</div>
</xsl:template>

<xsl:template match="w">
  <xsl:choose>
    <xsl:when test="@n=$word">
      <span class="link-w-highlight">
        <xsl:if test="c">
          <xsl:apply-templates select="c"/>
        </xsl:if>
      </span>
    </xsl:when>
    <xsl:otherwise>
      <span class="link-w">
        <a class="link-word">
          <xsl:attribute name="onclick">view();
            <!-- FUNCTION: doSomething(add=linkword) -->
            <xsl:text>doSomething(
'xml.php?do=add&add=linkword&type=
</xsl:text>
<xsl:value-of select="$id"/>
<xsl:text>', '</xsl:text>
<xsl:value-of select="@n"/>
<!-- FUNCTION: showArg(get=add=arg) -->
<xsl:text>');
showArg('xml.php?do=get&get=arg&type=letter', '</xsl:text>
<xsl:value-of select="$index"/>
<xsl:text>');</xsl:text>
          </xsl:attribute>
          <xsl:text>*</xsl:text>
        </a>
        <xsl:if test="c">
          <xsl:apply-templates select="c"/>
        </xsl:if>
      </span>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>

<xsl:template match="c">
  <xsl:choose>
    <xsl:when test="@n=$letter">
      <span class="link-c-highlight">
        <xsl:choose>
          <xsl:when test="string(.)">
            <xsl:param name="countg" select="count(g)"/>
            <xsl:param name="countu"
select="count(g[@ana='#unresolved'])"/>
            <xsl:param name="countr"
select="count(g[@ana='#resolved'])"/>
            <xsl:choose>
              <xsl:when test="$countr='1' and $countu='0'">
                <xsl:apply-templates>
                  <xsl:with-param name="type" select="1"/>
                </xsl:apply-templates>
              </xsl:when>
            </xsl:choose>
          </xsl:when>
        </xsl:choose>
      </span>
    </xsl:when>
  </xsl:choose>

```

```

        <xsl:when test="$countr='0' and $countu='1'">
            <xsl:apply-templates>
                <xsl:with-param name="type" select="2"/>
            </xsl:apply-templates>
        </xsl:when>
        <xsl:when test="$countr='1' and $countu!='0'">
            <xsl:apply-templates>
                <xsl:with-param name="type" select="3"/>
            </xsl:apply-templates>
        </xsl:when>
        <xsl:when test="$countr='0' and $countu!='0'">
            <xsl:text>?</xsl:text>
        </xsl:when>
    </xsl:choose>
</xsl:when>
<xsl:otherwise>
    <xsl:text>&#46;</xsl:text>
</xsl:otherwise>
</xsl:choose>
</span>
</xsl:when>
<xsl:otherwise>
<span class="link-c">
    <xsl:attribute name="onclick">
        <!-- FUNCTION: doSomething(add=linkletter) -->
        <xsl:text>doSomething(
'xml.php?do=add&add=linkletter&type=
    <xsl:value-of select="$id"/>
    <xsl:text>', '</xsl:text>
    <xsl:value-of select="@n"/>
    <!-- FUNCTION: showArg(get=add=arg) -->
    <xsl:text>');
showArg('xml.php?do=get&get=arg&type=letter', '</xsl:text>
    <xsl:value-of select="$index"/>
    <xsl:text>');</xsl:text>
    </xsl:attribute>
    <xsl:choose>
        <xsl:when test="string(.)">
            <xsl:param name="countg" select="count(g)"/>
            <xsl:param name="countu"
select="count(g[@ana='#unresolved'])"/>
            <xsl:param name="countr"
select="count(g[@ana='#resolved'])"/>
            <xsl:choose>
                <xsl:when test="$countr='1' and $countu='0'">
                    <xsl:apply-templates>
                        <xsl:with-param name="type" select="1"/>
                    </xsl:apply-templates>
                </xsl:when>
                <xsl:when test="$countr='0' and $countu='1'">
                    <xsl:apply-templates>
                        <xsl:with-param name="type" select="2"/>
                    </xsl:apply-templates>
                </xsl:when>
                <xsl:when test="$countr='1' and $countu!='0'">
                    <xsl:apply-templates>
                        <xsl:with-param name="type" select="3"/>
                    </xsl:apply-templates>
                </xsl:when>
                <xsl:when test="$countr='0' and $countu!='0'">
                    <xsl:text>?</xsl:text>
                </xsl:when>
            </xsl:choose>
        </xsl:when>
        <xsl:otherwise>
            <xsl:text>&#46;</xsl:text>
        </xsl:otherwise>
    </xsl:choose>
</span>
</xsl:otherwise>
</xsl:choose>
</xsl:template>

```

```

<xsl:template match="g">
  <xsl:param name="type"/>
  <xsl:choose>
    <xsl:when test="@ana='#resolved' and $type='1'">
      <xsl:value-of select="."/>
    </xsl:when>
    <xsl:when test="@ana='#unresolved' and $type='2'">
      <xsl:value-of select="."/><xsl:text>&#x0323;</xsl:text>
    </xsl:when>
    <xsl:when test="@ana='#resolved' and $type='3'">
      <xsl:value-of select="."/><xsl:text>&#x0323;</xsl:text>
    </xsl:when>
  </xsl:choose>
</xsl:template>
</xsl:stylesheet>

```

```

<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<!--
* get_word.xsl
*
* XSLT stylesheet for the js function showArg()
*
* @package    DSS prototype
* @author     Henriette Roued-Cunliffe <henriette@roued.com>
* @license    for OS license
* @link       http://roued.com/e-doc
* @since      File available since 2011
-->
<xsl:param name="index"/>
<xsl:strip-space elements="*" />
<xsl:template match="/">
<xsl:apply-templates select="//w[@n = $index]" />
</xsl:template>

<xsl:template match="//w[@n = $index]">
<xsl:apply-templates select="c" />

</xsl:template>

<xsl:template match="c">
  <xsl:choose>
    <xsl:when test="string(.)">
      <xsl:param name="countg" select="count(g)" />
      <xsl:param name="countu" select="count(g[@ana='#unresolved'])" />
      <xsl:param name="countr" select="count(g[@ana='#resolved'])" />
      <xsl:choose>
        <xsl:when test="$countr='1' and $countu='0'">
          <xsl:apply-templates>
            <xsl:with-param name="type" select="1" />
          </xsl:apply-templates>
        </xsl:when>
        <xsl:when test="$countr='0' and $countu='1'">
          <xsl:apply-templates>
            <xsl:with-param name="type" select="2" />
          </xsl:apply-templates>
        </xsl:when>
        <xsl:when test="$countr='1' and $countu!='0'">
          <xsl:apply-templates>
            <xsl:with-param name="type" select="3" />
          </xsl:apply-templates>
        </xsl:when>
        <xsl:when test="$countr='0' and $countu!='0'">
          <xsl:text>?</xsl:text>
        </xsl:when>
      </xsl:choose>
    </xsl:when>
    <xsl:otherwise>
      <xsl:text>&#46;</xsl:text>
    </xsl:otherwise>
  </xsl:choose>

```

```

</xsl:template>

<xsl:template match="g">
  <xsl:param name="type"/>
  <xsl:choose>
    <xsl:when test="@ana='#resolved' and $type='1'">
      <xsl:value-of select="."/>
    </xsl:when>
    <xsl:when test="@ana='#unresolved' and $type='2'">
      <xsl:value-of select="."/><xsl:text>&#x0323;</xsl:text>
    </xsl:when>
    <xsl:when test="@ana='#resolved' and $type='3'">
      <xsl:value-of select="."/><xsl:text>&#x0323;</xsl:text>
    </xsl:when>
  </xsl:choose>
</xsl:template>

</xsl:stylesheet>

```

```

<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<!--
* get_sugword.xsl
*
* XSLT stylesheet for the js function showTop(get=word)
*
* @package    DSS prototype
* @author     Henriette Roued-Cunliffe <henriette@roued.com>
* @license    for OS license
* @link       http://roued.com/e-doc
* @since      File available since 2011
-->
<xsl:param name="index"/>
<xsl:strip-space elements="*" />
<xsl:template match="/">
  <xsl:apply-templates select="//w[@n = $index]" />
</xsl:template>

<xsl:template match="//w[@n = $index]">
  <xsl:apply-templates select="c" />
</xsl:template>

<xsl:template match="c">
  <xsl:choose>
    <xsl:when test="string(.)">
      <xsl:param name="countg" select="count(g)" />
      <xsl:param name="countu" select="count(g[@ana='#unresolved'])" />
      <xsl:param name="countr" select="count(g[@ana='#resolved'])" />
      <xsl:choose>
        <xsl:when test="$countr='1' and $countu='0'">
          <xsl:apply-templates>
            <xsl:with-param name="type" select="1" />
          </xsl:apply-templates>
        </xsl:when>
        <xsl:when test="$countr='0' and $countu='1'">
          <xsl:text>*</xsl:text>
        </xsl:when>
        <xsl:when test="$countr='1' and $countu!='0'">
          <xsl:apply-templates>
            <xsl:with-param name="type" select="3" />
          </xsl:apply-templates>
        </xsl:when>
        <xsl:when test="$countr='0' and $countu!='0'">
          <xsl:text>[</xsl:text><xsl:apply-templates>
            <xsl:with-param name="type" select="4" />
          </xsl:apply-templates><xsl:text>]</xsl:text>
        </xsl:when>
      </xsl:choose>
    </xsl:when>
    <xsl:otherwise>
      <xsl:text>*</xsl:text>
    </xsl:otherwise>
  </xsl:choose>

```

```

        </xsl:otherwise>
    </xsl:choose>
</xsl:template>

<xsl:template match="g">
    <xsl:param name="type"/>
    <xsl:choose>
        <xsl:when test="@ana='#resolved' and $type='1'">
            <xsl:text></xsl:text><xsl:value-of select="."/><xsl:text></xsl:text>
        </xsl:when>
        <xsl:when test="$type='4'">
            <xsl:value-of select="."/>
        </xsl:when>
        <xsl:when test="@ana='#resolved' and $type='3'">
            <xsl:value-of select="."/>
        </xsl:when>
    </xsl:choose>
</xsl:template>

</xsl:stylesheet>

```

```

<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<!--
* sug_word.xsl
*
* XSLT stylesheet for the js function showSug
*
* @package    DSS prototype
* @author     Henriette Roued-Cunliffe <henriette@roued.com>
* @license    for OS license
* @link       http://roued.com/e-doc
* @since      File available since 2011
-->
<xsl:param name="documentws"/>
<xsl:template match="/">
    <xsl:choose>
        <xsl:when test="//list[@type='words']/item">
            <xsl:text> Results from: </xsl:text><xsl:value-of
select="//catRef/@target"/><br/><br/>
            <ul>
                <xsl:apply-templates select="//list[@type='words']/item">
                    <xsl:sort select="ident/@type" order="ascending"/>
                    <xsl:sort select="@n" order="ascending"/>
                </xsl:apply-templates>
            </ul>
        </xsl:when>
        <xsl:otherwise>
            <xsl:text>There is nothing in the </xsl:text><xsl:value-of
select="//catRef/@target"/> <xsl:text> that matches this
pattern!</xsl:text><br/><br/>
        </xsl:otherwise>
    </xsl:choose>
</xsl:template>

<xsl:template match="//list[@type='words']/item">
    <li>
        <b><xsl:value-of select="ident/@type"/></b>: <xsl:value-of select="ident"/>
        <xsl:if test="list[@type='tablets']">
            <xsl:text> </xsl:text>
            <xsl:apply-templates select="list[@type='tablets']/item"/>
        </xsl:if>
        <xsl:if test="list[@type='lemma_subtype']">
            <ul><xsl:apply-templates
select="list[@type='lemma_subtype']/item"/></ul>
        </xsl:if>
        <xsl:if test="list[@type='date']">
            <ul><xsl:apply-templates select="list[@type='date']/item"/></ul>
        </xsl:if>
    </li>
</xsl:template>

```

```

<xsl:template match="list[@type='lemma_subtype']/item">
  <li>
    <b>sub-lemma</b>: <xsl:value-of select="ident"/>
    <xsl:if test="list[@type='tablets']">
      <xsl:text> </xsl:text>
      <xsl:apply-templates select="list[@type='tablets']/item"/>
    </xsl:if>
  </li>
</xsl:template>

<xsl:template match="list[@type='date']/item">
  <li><b>date</b>: <xsl:value-of select="ident"/></li>
</xsl:template>

<xsl:template match="list[@type='tablets']/item">
  <a target="_blank">
    <xsl:attribute name="href">
      <xsl:value-of select="$documentws"/><xsl:value-of select="ident"/>
    </xsl:attribute>
    <xsl:value-of select="ident"/>
  </a>
  <xsl:text>, </xsl:text>
</xsl:template>

</xsl:stylesheet>

```

```

<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<!--
* sug_lexicon.xsl
*
* XSLT stylesheet for the js function showSug
*
* @package      DSS prototype
* @author       Henriette Roued-Cunliffe <henriette@roued.com>
* @license      for OS license
* @link         http://roued.com/e-doc
* @since        File available since 2011
-->
<xsl:param name="documentws"/>
<xsl:template match="/">
  <xsl:value-of select="@WORD"/>
  <xsl:apply-templates select="//RES"/>
</xsl:template>

<xsl:template match="//RES">
  <p>
    <b><xsl:value-of select="@AUTHOR"/> <xsl:text> </xsl:text></b>
    <xsl:value-of select="@DICT"/>
    <xsl:text> </xsl:text>
    <a target="_blank">
      <xsl:attribute name="href">
        <xsl:value-of select="@LINK"/>
      </xsl:attribute>
      <xsl:value-of select="@LEM"/>
    </a>
  </p>
</xsl:template>

</xsl:stylesheet>

```

APPENDIX E

This appendix describes in more detail how the DSS prototype was developed and how the different files, views, scripts and database tables fit together. First of all this should give the reader of this thesis a better understanding of the DSS prototype development. Secondly, it will act as a guide to anyone who might wish to use this prototype as a concept to build a fully functioning DSS as proposed in the conclusion (Chapter 5). This appendix is divided into 4 diagrams. Each diagram has a start point and an end point. The start point is either a view or a pop-up box or as in diagram 1 the loading of the `index.php` page. The end point can also be a view such as the `DSS Main View`, or a pop-up box such as the `Letters Pop-up Box`.

DIAGRAM 1

This diagram begins with the file `index.php`. In which the images `670f-a.jpg` and `legend.jpg` are displayed. The files `DSS_style.css` (contains all styling for the prototype) and `config.php` (configuration file where variables are set) are both included in the index file too. The index file is the scaffolding of the prototype. AJAX scripts send information back to hooks in the index file, which are then displayed without the need to reload the page. The first scripts are activated by the `body onload` script when the `index.php` file is loaded.

The script `view()` is a good example of this. `view()` is a delay script for `view1()`. While `view1()` is trying to load, `view()` displays the `ajax-loader.gif` image so that the user knows that something is going to happen soon.

`view1()` in turn runs the file; in this case `view.php` and retrieves the output from this file. This output is then sent on to `stateChagedView()` which in turn sends it onto the hook in `index.php` in this case `<div id="main-view"></div>`. In this way once `view()` is

activated it will send a request to `view.php`. The output returned will be displayed on the page in-between the `<div>` tags.

`view.php` creates this output by retrieving the document structure from the `db-tei.php` Web Service which it then styles with the `view.xsl` XSLT style sheet. `db-tei.php` returns the structure of the 670 tablet as TEI formatted XML. This structure has been extracted from the four structural database tables: `DSS_lines`, `DSS_words`, `DSS_characters` and `DSS_letters`. At the end of this chain of files and scripts is the DSS main view where the structure of tablet 670 is displayed.

LINE 1:

*M

a

r

t

i

u

s

*V

i

c

t

o

r

i

*f

r

a

t

r

i

LINE 2:

*k

a

r

i

s

s

i

m

o

*s

a

l

u

t

e

m

LINE 3:

*s

c

i

a

s

*m

e

*r

e

c

t

e

*e

s

s

e

*q

u

o

d

*t

e

LINE 4:

*i

n

u

i

c

e

m

*f

a

c

e

r

e

*c

u

p

i

o

*p

r

o

c

u

r

a

t

o

r

e

m

LINE 5:

*t

e

*f

a

c

i

o

*f

r

a

t

e

r

**

*

Below the chain for the DSS main view is the chain for the Search Engine View. This works in the same way as the above example where `showEng()` is activated first and acts as a delay for `showEng1()`, which in turn runs `xml.php` with the command `get=eng`. This gathers information about the search engines available to the prototype from the `DSS_engines` table. Once returned this is passed on to `stateChangedEng()` which sends it on to the hook `<div id="engbox">` creating the search engine view which sits in the left sidebar.

SEARCH ENGINES

- Vindolanda Tablets Online II 
- Inscriptions of Roman Tripolitania 
- Inscriptions of Aphrodisias 
- Perseus Latin Dictionaries 

From the search engine view it is possible to activate some other scripts. Each search engine can be turned on and off with radio-buttons. If turned off, it activates the script: `doSomething(del=eng)`. If turned on, it activates the `doSomething(add=eng)` script.

Each time any of these scripts are run the `showEng()` script is activated as well to show the updated search engine view. `doSomething()` is a generic script used for a variety of things in the prototype depending on the command that comes with it. In this case the commands are `del=eng` and `add=eng`. `del=eng` updates the 'used' field in `DSS_engines` to 0. and `add=eng` updates the same field to 1. If 'used'=1 for a particular search engine this search engine will be used later in the suggestion search. If 'used'=0 it will not be used. `doSomething()` does not return an output and therefore needs no delay.

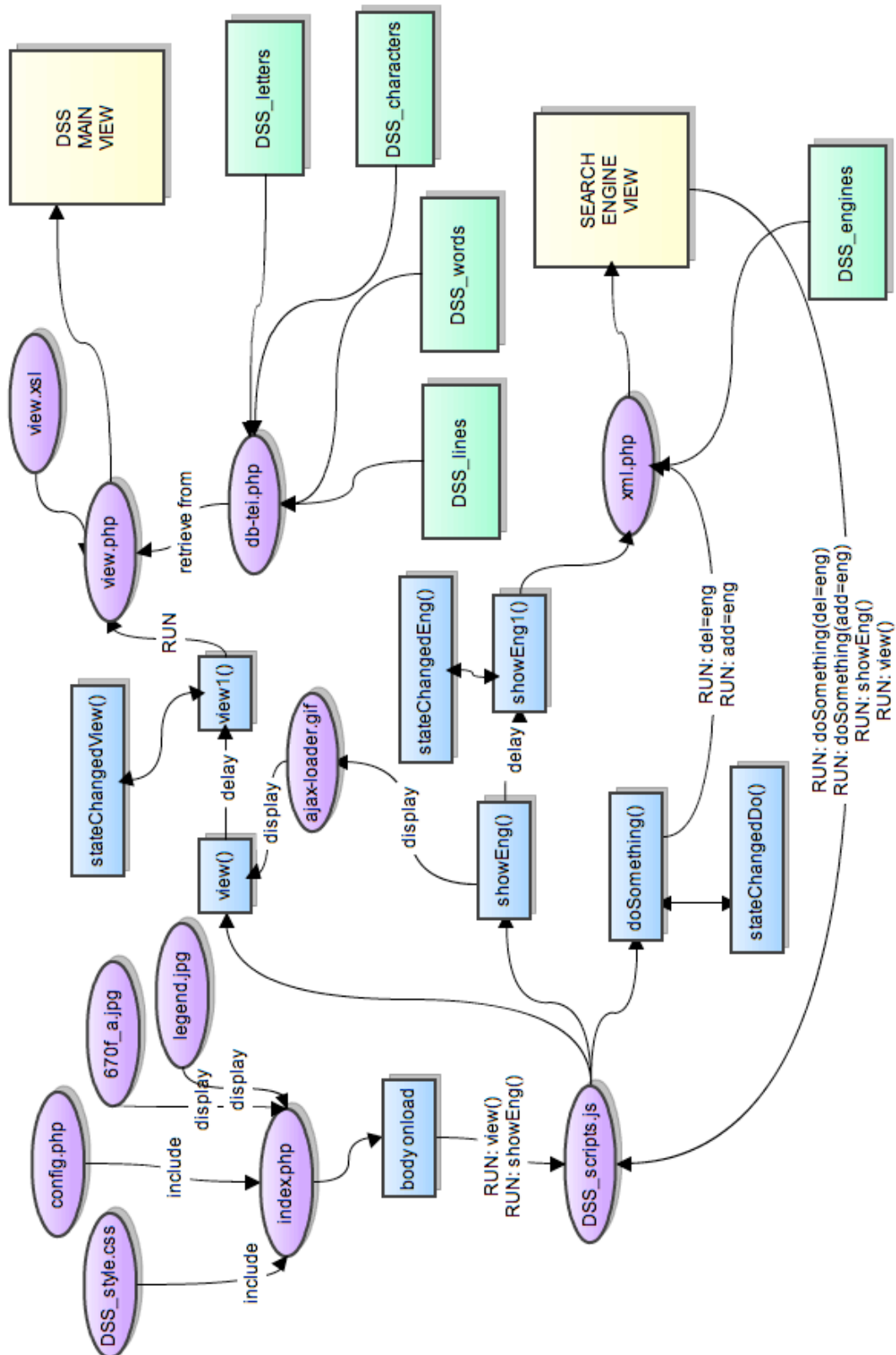


DIAGRAM 2

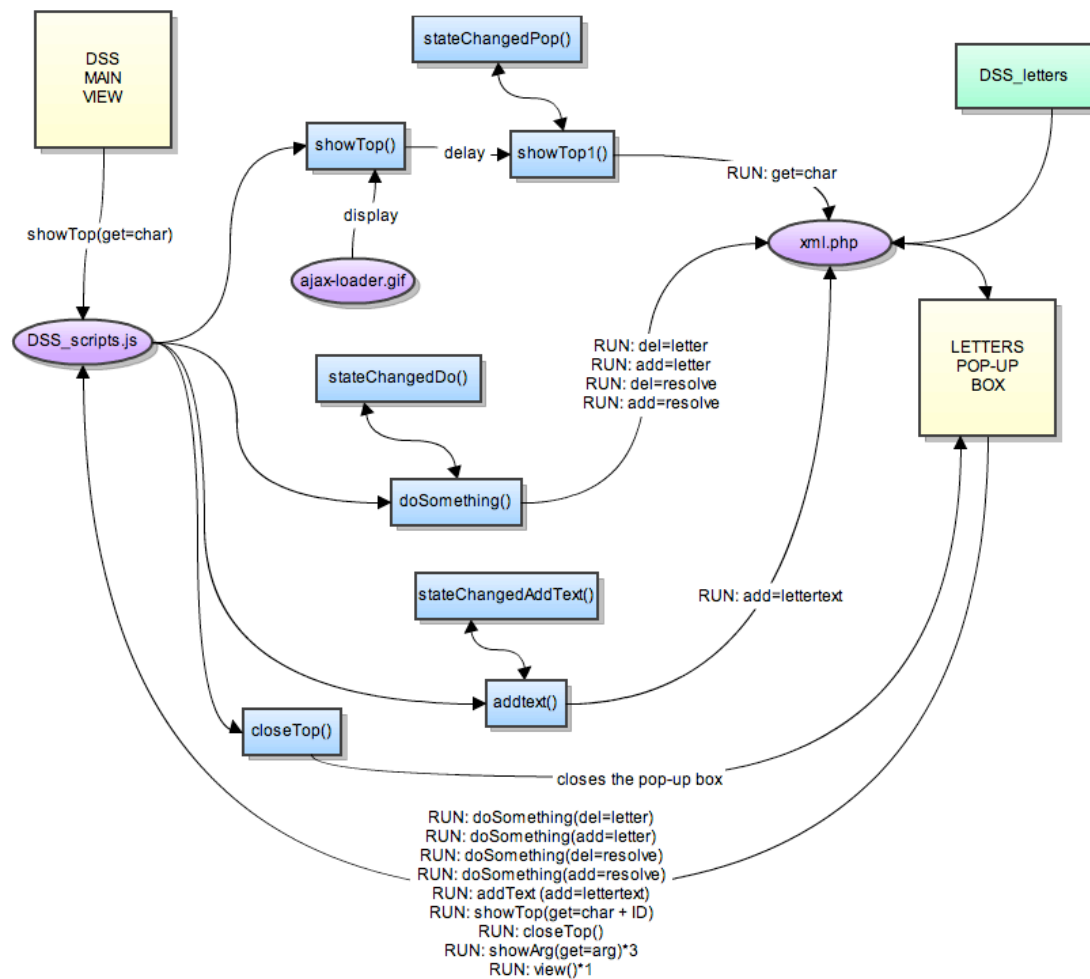
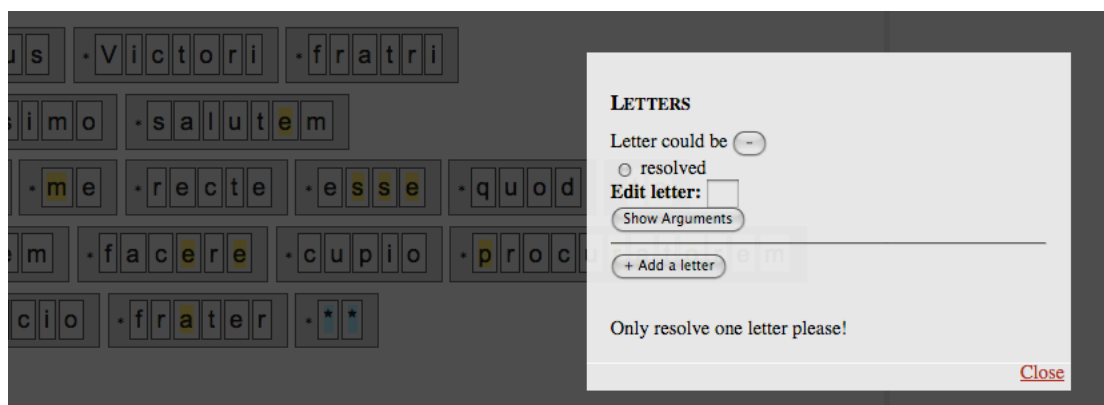


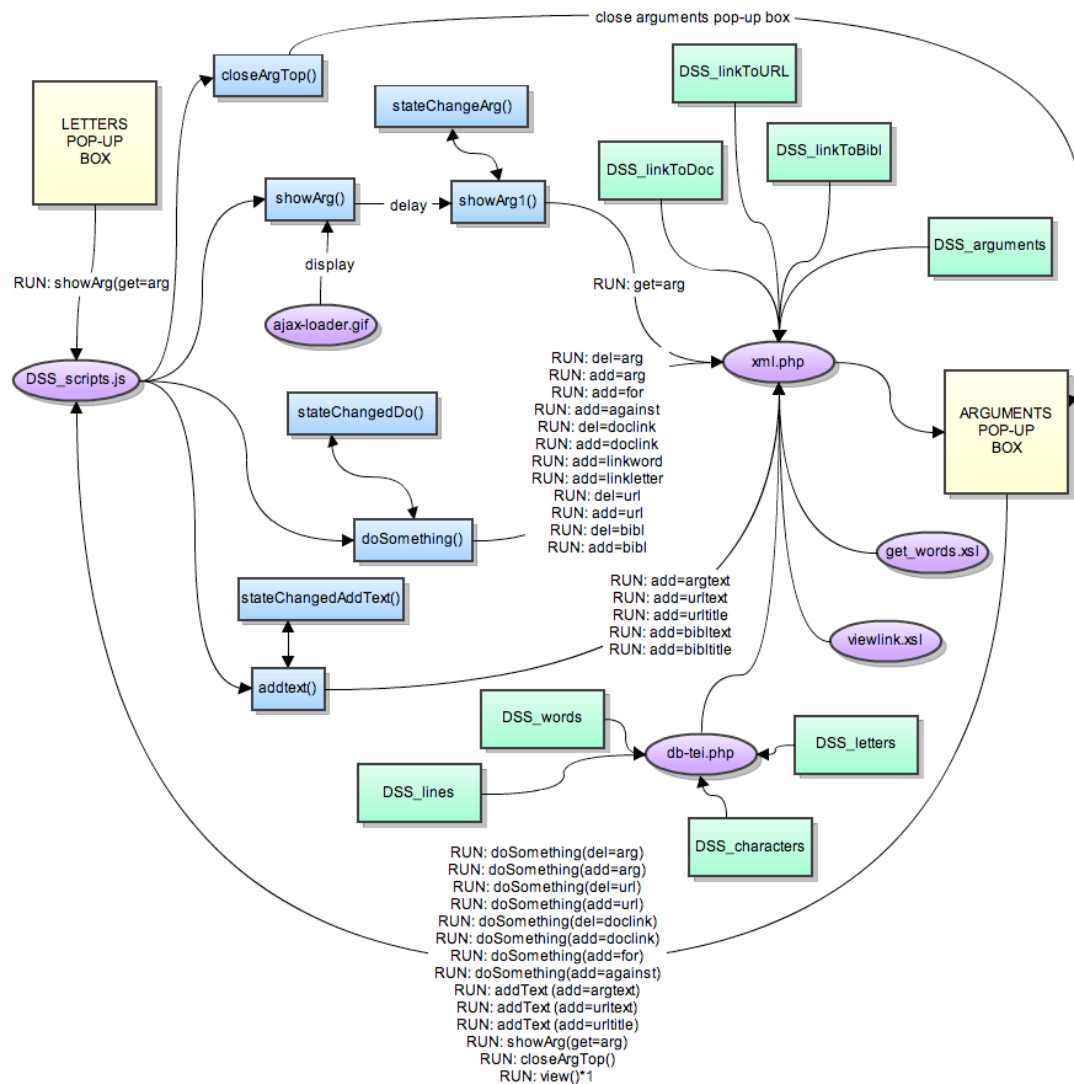
Diagram 2 begins with a command from the `DSS main view` which was the result of Diagram 1. The command is `showTop(get=char)`. In other words, it brings up the pop-up box for a certain character.



The `showTop()` script is a delay for the `showTop1()` script. This runs the `get=char` command in `xml.php` which gets the letters for a character ID from the `dss.letters` table. `stateChangedTop()` sends this on to the `<div id='topbox'>` hook in `index.php` displaying the `Letters Pop-up Box`. From this pop-up box, a number of `doSomething()` scripts are called plus an `addText()` script. The `doSomething()` commands are `del=letter` (deletes a letter), `add=letter` (adds a letter to the character), `add=resolve` (resolves a letter) and `del=resolve` (removes the resolve from a letter).

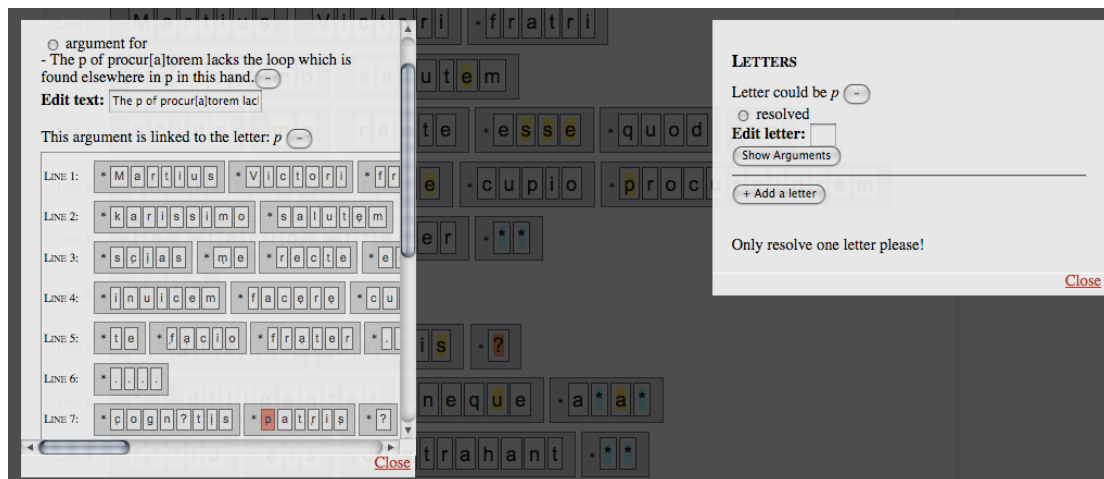
`addText()` scripts are used to add and edit text in the prototype. In this case the command `add=lettertext` enables the editing of a certain letter. `closeTop()` is a script that closes the pop-up box.

DIAGRAM 3



This diagram carries on from diagram 2 – the Letter Pop-up Box which activates the `showArg (get=arg)` script when the Show Arguments button is clicked.

`showArg()` is a delay for `showArg1()` which runs the `get=arg` command in `xml.php`. This command retrieves the arguments for each letter from the table `DSS_arguments`. `stateChangeArg()` then places this output in the hook `<div id='argbox'>` displaying the Arguments Pop-up Box.



From the Argument Pop-up Box arguments can be added (`add=arg`), deleted (`del=arg`), and can be for (`add=fav`) and against (`add=against`) a letter. The argument text (`add=argtext`) can be edited too.

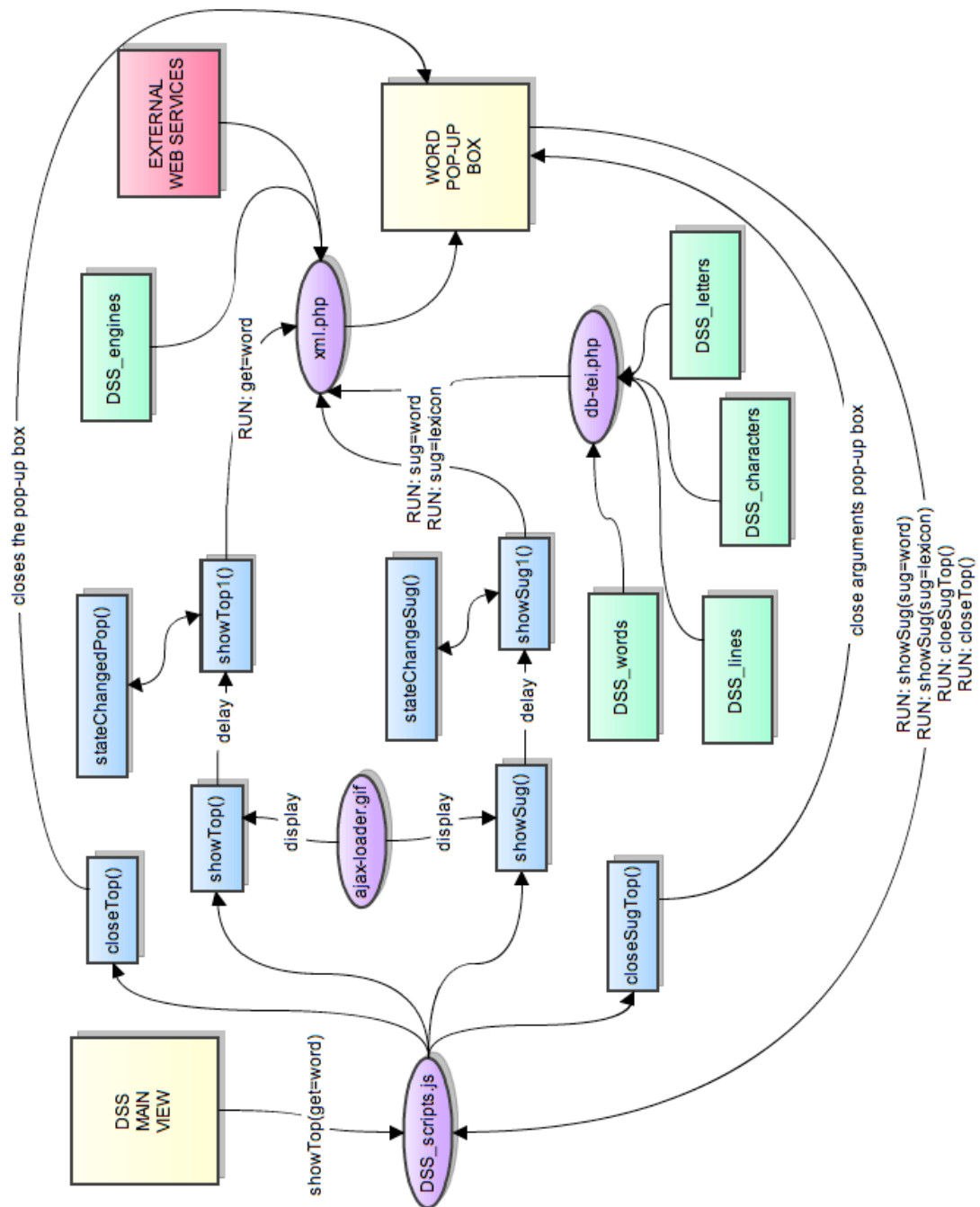
The arguments pop-up box interface also enables the activation of the different types of linking that an argument can have. The first is linking to other words or characters in the documents. This is controlled with the `doSomething()` commands: `add=doclink` (add a link to the document), `del=doclink` (delete a link to the document), `add=linkword` (link a word to the argument) and `add=linkletter` (link a letter to the argument).

It is also possible to link to a URL (`del=url`, `add=url`, `add=urltext` and `add=urltitle`) and to link to a bibliographic reference (`del=bibl`, `add=bibl`, `add=bibltext`, `add=bibltitle`). The `doSomething()` and `addtext()` scripts work as before by sending the commands on to `xml.php`. Here they mainly update the database tables: `DSS_arguments`, `DSS_linkToDoc`, `DSS_linkToURL`, and `DSS_linkToBibl`. The only differences is the command `add=doclink` which includes the structure of the document from `db.tei.php` transformed with the `viewlink.xsl` stylesheet to produce the mini version of the main view.

From this mini view words and letters can be clicked to link them to the argument. When linking to a word the `db-tei.php` Web Service is called again this time transformed

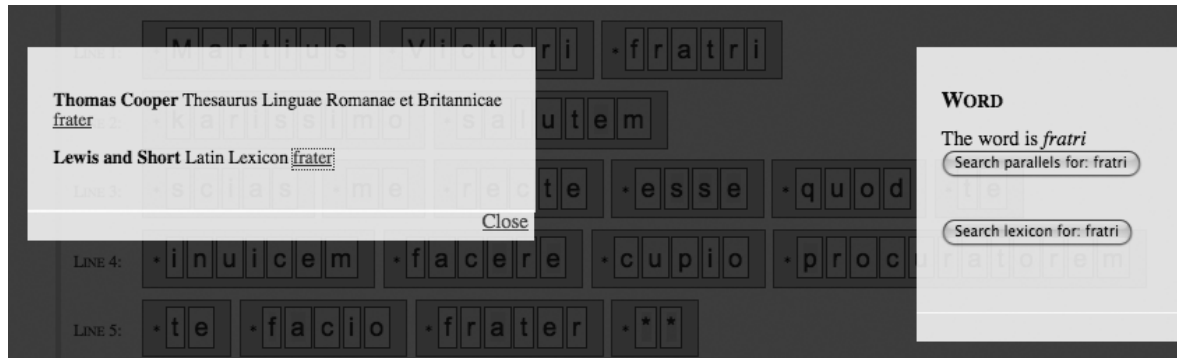
with `get-words.xml` to display the word that has been linked to. The `closeArgTop()` script closes the pop-up box.

DIAGRAM 4



This final diagram begins in the same way as diagram 2 with the `DSS_mainview`. Like before the `showTop()` script is run but this time with the command `get=word`. This command displays the word that the user clicked on with the aid of the `db-tei.php` Web

Service. The word is displayed in the Word Pop-up Box along with buttons to search either the concordance search engines (`showSug(sug=word)`) or the lexicon search engines (`showSug(sug=lexican)`).



The `showSug()` script is a delay for the `showSug1()` script. This sends the commands to `xml.php` which checks in the tables `DSS_engines` for search engines to search depending on whether they are in use (see diagram 1) or not and whether they are a concordance or a lexical engine. The table also contains the URL for the external Web Services which `xml.php` in turn searches for the word pattern. The result of this search is displayed in the Suggestion Pop-up Box.

Finally, the Word Pop-up Box is closed with the `closeTop()` script and the suggestion pop-up box is closed with the `closeSugTop()` script.

APPENDIX F

Code for APPELLO file index.php –developed by Henriette Roued-Cunliffe. See original file in folder APPELLO on CD.

```
<?php

/**
 * index.php
 *
 * Processing script for the APPELLO word search Web Service,
 *
 * @package    APPELLO
 * @author     Henriette Roued-Cunliffe <henriette@roued.com>
 * @license    for OS license
 * @link       http://roued.com/e-doc
 * @since      File available since 2010
 */

// FILE: Server.php - this file must be on server to run the Web Service - can be
// downloaded and installed from http://framework.zend.com/
include_once ('Zend/Rest/Server.php');

// ---- GET TABLETS ---- //
/*
 * Function which just returns a list of tablets in this resource
 * NO PARAMETERS
 */
// FUNCTION: get_tablets()
function get_tablets() {
    $source = "Vindolanda Writing Tablets";
    $today_d = date(d);
    $today_m = date(m);
    $today_y = date(Y);
    $today = $today_d . '-' . $today_m . '-' . $today_y;
    $bibl_array = array(
        'Tab.Vindol.I' => 'A.K.Bowman, J.D.Thomas, Vindolanda: the Latin writing-
        tablets, Britannia Monograph 4. London, 1983 ',
        'Tab.Vindol.II' => 'A.K.Bowman, J.D.Thomas, The Vindolanda Writing-Tablets
        (Tabulae Vindolandenses II). London, 1994',
        'Tab.Vindol.III' => 'A.K.Bowman, J.D.Thomas, The Vindolanda Writing Tablets
        (Tabulae Vindolandenses III). London: British Museum Publications, 2003');

    //error_code=0 for Success / error_code=1 for Failure
    $error_code = 0;
    // SET: new DOMDocument
    $doc = new DOMDocument();
    $doc->formatOutput = true;
    // OUTPUT: building a TEI document
    $tei = $doc->createElement("TEI");
    $doc->appendChild($tei);
    // OUTPUT: teiHeader
    $teiheader = $doc->createElement("teiHeader");
    $tei->appendChild($teiheader);
    // OUTPUT: fileDesc
    $fileDesc = $doc->createElement("fileDesc");
    $teiheader->appendChild($fileDesc);
    // OUTPUT: title
    $titleStmt = $doc->createElement("titleStmt");
    $fileDesc->appendChild($titleStmt);
    $title = $doc->createElement("title");
    $title->appendChild($doc->createTextNode('APPELLO method: Get Tablets'));
    $titleStmt->appendChild($title);
    // OUTPUT: publicationStmt
    $publicationStmt = $doc->createElement("publicationStmt");
    $fileDesc->appendChild($publicationStmt);
    $authority = $doc->createElement("authority");
```

```

$authority->appendChild($doc->createTextNode('Made available by Centre for the
Study of Ancient Documents, University of Oxford (http://www.csad.ox.ac.uk/) and
Henriette Roued-Cunliffe as a part of her D.Phil at the University of Oxford
(http://www.roued.com/e-doc/')));
$publicationStmt->appendChild($authority);
$date = $doc->createElement("date");
$date->appendChild($doc->createTextNode($today));
$publicationStmt->appendChild($date);
// OUTPUT: sourceDesc
$sourceDesc = $doc->createElement("sourceDesc");
$fileDesc->appendChild($sourceDesc);
$bibl4 = $doc->createElement("bibl");
$bibl4->appendChild($doc->createTextNode($bibl_array['Tab.Vindol.II']));
$sourceDesc->appendChild($bibl4);
$bibl4->setAttribute("n", 'Tab.Vindol.II');
$bibl5 = $doc->createElement("bibl");
$bibl5->appendChild($doc->createTextNode($bibl_array['Tab.Vindol.III']));
$sourceDesc->appendChild($bibl5);
$bibl5->setAttribute("n", 'Tab.Vindol.III');
// OUTPUT: encodingDesc
$encodingDesc = $doc->createElement("encodingDesc");
$teiheader->appendChild($encodingDesc);
$ab5 = $doc->createElement("ab");
$ab5->appendChild($doc->createTextNode('This file was produced by APPELLO
version 0.2'));
$encodingDesc->appendChild($ab5);
// OUTPUT: profileDesc
$profileDesc = $doc->createElement("profileDesc");
$teiheader->appendChild($profileDesc);
$textClass = $doc->createElement("textClass");
$profileDesc->appendChild($textClass);
$catRef = $doc->createElement("catRef");
$textClass->appendChild($catRef);
$catRef->setAttribute("target", 'List of Documents in '.$source);
// OUTPUT: langUsage
$langUsage = $doc->createElement("langUsage");
$profileDesc->appendChild($langUsage);
$language1 = $doc->createElement("language");
$language1->appendChild($doc->createTextNode('English'));
$langUsage->appendChild($language1);
$language1->setAttribute("ident", 'eng');
$language2 = $doc->createElement("language");
$language2->appendChild($doc->createTextNode('Latin'));
$langUsage->appendChild($language2);
$language2->setAttribute("ident", 'lat');
// OUTPUT: Text
$text = $doc->createElement("text");
$tei->appendChild($text);
$body = $doc->createElement("body");
$text->appendChild($body);
$list = $doc->createElement("list");
$body->appendChild($list);
$list->setAttribute("type", "tablets");

// SET: directory
$dir="tablets";
// IF: the directory can be scanned
if($scans = scandir($dir, 0)){
    // SET: the amount of files in the directory
    $count = count($scans);
    // FOR: each file
    for($i=2;$i<$count;$i++){
        // SET: file address
        $tablet=$dir . '/' . $scans[$i];
        // IF: the file is loaded
        if($xml_tablet = simplexml_load_file($tablet)){
            //OUTPUT: new APPELLO XML
            $xml_tablet->registerXPathNamespace('tei', 'http://www.tei-
c.org/ns/1.0');
            $tabID1=$xml_tablet->xpath('@n');
            $tabID=$tabID1[0];
            $item = $doc->createElement("item");
            $list->appendChild($item);

```

```

$item->setAttribute("n", $tabID);
$inventory1=$xml_tablet->xpath('//tei:rs[@type="stilusInv"]');
$inventory=$inventory1[0];
$category1=$xml_tablet->xpath('//tei:catRef/@target');
$category=$category1[0];
$category1=$xml_tablet->xpath('//tei:catRef/@target');
$category=$category1[0];
$title_st1=$xml_tablet->xpath('//tei:title/node()');
$title_st2=$xml_tablet->xpath('//tei:title/*');
$title_st=$title_st1[0] . $title_st2[0];
$ref_st='?method=get_tablet&tabletID=' . $tabID;
$bibls=$xml_tablet->xpath('//tei:bibl[@type="original"]/@n');
$bibl_id=$bibls[0];
$bibl_st=$bibl_array['' . $bibl_id . ''];
// IF: there are figures
if($xml_tablet->xpath('//tei:figure')){
    $ab = $doc->createElement("ab");
    $fig1=$xml_tablet->xpath('//tei:figure/tei:graphic/@url');
    for($k=0;$k<count($fig1);$k++){
        $figure=$fig1[$k];
        $fig = $doc->createElement("figure");
        $ab->appendChild($fig);
        $head = $doc->createElement("head");
        $head->appendChild($doc->createTextNode($figure));
        $fig->appendChild($head);
    } // END: for($k=0;$k<count($fig1);$k++){
    $item->appendChild($ab);
    $ab->setAttribute("type", "images");
} // END: f($xml_tablet->xpath('//tei:figure')){
// OUTPUT: dimensions
$widths=$xml_tablet->xpath('//tei:width');
$width=$widths[0];
$widthunits=$xml_tablet->xpath('//tei:width/@unit');
$widthunit=$widthunits[0];
$heights=$xml_tablet->xpath('//tei:height');
$height=$heights[0];
$heightunits=$xml_tablet->xpath('//tei:height/@unit');
$heightunit=$heightunits[0];
// OUTPUT: archaeological context
$archLocations=$xml_tablet->xpath('//tei:rs[@type="archLocation"]');
$archLocation=$archLocations[0];
$archPeriods=$xml_tablet->xpath('//tei:rs[@type="archPeriod"]');
$archPeriod=$archPeriods[0];
$plates=$xml_tablet->xpath('//tei:rs[@type="plate"]');
$plate=$plates[0];
$cxtDates=$xml_tablet->xpath('//encodingDesc/p/date');
$cxtDate=$cxtDates[0];
$cxtNames=$xml_tablet->xpath('//encodingDesc/p/persName');
$cxtName=$cxtNames[0];
$cxtEncoded=$xml_tablet->xpath('//encodingDesc');
if(isset($cxtEncoded[0])){
    $isCxtEncoded='yes';
}else{
    $isCxtEncoded='';
} // END: if(isset($cxtEncoded[0])){
}else{
    $error_code=1;
} // END: if($xml_tablet = simplexml_load_file($tablet)){
// OUTPUT: metadata for header
$idnt1 = $doc->createElement("ident");
$idnt1->appendChild($doc->createTextNode($inventory));
$item->appendChild($idnt1);
$idnt1->setAttribute("type", "inventory");
$idnt2 = $doc->createElement("ident");
$idnt2->appendChild($doc->createTextNode($category));
$item->appendChild($idnt2);
$idnt2->setAttribute("type", "category");
$title = $doc->createElement("title");
$title->appendChild($doc->createTextNode($title_st));
$item->appendChild($title);
$ref = $doc->createElement("ref");
$ref->appendChild($doc->createTextNode($ref_st));
$item->appendChild($ref);

```

```

$bibl = $doc->createElement("bibl");
$bibl->appendChild($doc->createTextNode($bibl_st));
$item->appendChild($bibl);
$bibl->setAttribute("n", $bibl_id);
if($width!=''){
    $ab1 = $doc->createElement("ab");
    $item->appendChild($ab1);
    $ab1->setAttribute("type", "measurements");
    $width1 = $doc->createElement("width");
    $width1->appendChild($doc->createTextNode($width));
    $ab1->appendChild($width1);
    $width1->setAttribute("unit", $widthunit);
    $height1 = $doc->createElement("height");
    $height1->appendChild($doc->createTextNode($height));
    $ab1->appendChild($height1);
    $height1->setAttribute("unit", $heightunit);
}
if($plate!=''){
    $ab2 = $doc->createElement("ab");
    $ab2->appendChild($doc->createTextNode($plate));
    $item->appendChild($ab2);
    $ab2->setAttribute("type", "plates");
}
if($archLocation!=''){
    $ab3 = $doc->createElement("ab");
    $ab3->appendChild($doc->createTextNode($archLocation));
    $item->appendChild($ab3);
    $ab3->setAttribute("type", "archLocation");
}
if($archPeriod!=''){
    $ab4 = $doc->createElement("ab");
    $ab4->appendChild($doc->createTextNode($archPeriod));
    $item->appendChild($ab4);
    $ab4->setAttribute("type", "archPeriod");
}
}
}
}else{
    $error_code = 1;
} // END: if($scans = scandir($dir, 0)){

//Return status failed, if any error found.
if ($error_code == 1) {
    $ab7 = $doc->createElement("ab");
    $ab7->appendChild($doc->createTextNode('failed'));
    $encodingDesc->appendChild($ab7);
    $ab7->setAttribute("type", "status");
}
// Return status passed, if no error found
else if ($error_code == 0) {
    $ab7 = $doc->createElement("ab");
    $ab7->appendChild($doc->createTextNode('passed'));
    $encodingDesc->appendChild($ab7);
    $ab7->setAttribute("type", "status");
}
return simplexml_load_string($doc->saveXML());
}

// ---- GET TABLET ---- //
/*
* Simple function which just returns the tablet specified in the tablet ID in
it's original XML
* PARAMETERS
* $tabletID: Identified the tablet, which will be returned.
* The possible tabletID can be identified from the function get_tablets.
*/
// FUNCTION: get_tablet()
function get_tablet($tabletID) {
    // SET: DOMDocument
    $doc = new DOMDocument();
    // SET: directory
    $dir="tablets";
    // SET: tablet file name
    $tablet=$dir . '/' . $tabletID . '.xml';

```

```

// FILE: load tablet XML file
$doc->load($tablet);
// OUTPUT: XML file
return simplexml_load_string($doc->saveXML());
}

// ---- GET WORD ---- //
/*
 * Function which retrieves all the words in the tablets
 * OPTIONAL PARAMETER
 * $pattern: If a pattern is sent with the function only the words which fit the
pattern will be returned.
 * Pattern wildcard is *
 * Alternative characters can be presented like this: {abc}
 */
// FUNCTION: get_word()
function get_word($pattern = FALSE) {
    // SET: meta for the APPELLO XML file
    $method_title = 'APPELLO method: Get Words';
    $authority1 = 'Made available by Centre for the Study of Ancient Documents,
University of Oxford (http://www.csad.ox.ac.uk/) and Henriette Roued-Cunliffe as
a part of her D.Phil at the University of Oxford (http://www.roued.com/e-doc/)';
    $today = date(d) . '-' . date(m) . '-' . date(Y);
    $bibl_array = array(
        'Tab.Vindol.I' => 'A.K.Bowman, J.D.Thomas, Vindolanda: the Latin writing-
tablets, Britannia Monograph 4. London, 1983 ',
        'Tab.Vindol.II' => 'A.K.Bowman, J.D.Thomas, The Vindolanda Writing-Tablets
(Tabulae Vindolandenses II). London, 1994',
        'Tab.Vindol.III' => 'A.K.Bowman, J.D.Thomas, The Vindolanda Writing Tablets
(Tabulae Vindolandenses III). London: British Museum Publications, 2003 ');
    $bibl_ref = array('Tab.Vindol.II', 'Tab.Vindol.III');
    $encoding = 'This file was produced by APPELLO version 0.2';
    $list_of = 'words';
    $source = "Vindolanda Writing Tablets";
    $langs = array(
        'eng' => 'English',
        'lat' => 'Latin',
        'grc' => 'Greek',
    );
    $lang_list = array('eng', 'lat');
    // IF: the pattern parameter was sent
    if($pattern){
        $search = array(
            '{',
            '}',
            '*',
        );
        $replace = array(
            '[',
            ']{1}',
            '([a-zA-Z]{1})',
        );
        $pattern = str_replace($search, $replace, $pattern);
        $pattern = '/' . $pattern . '/';
    }
    //error_code=0 for Success / error_code=1 for Failure
    $error_code = 0;
    // SET: DOMDocument
    $doc = new DOMDocument();
    $doc->formatOutput = true;
    // OUTPUT: TEI element
    $tei = $doc->createElement("TEI");
    $doc->appendChild($tei);
    // OUTPUT: teiHeader
    $teiheader = $doc->createElement("teiHeader");
    $tei->appendChild($teiheader);
    // OUTPUT: fileDesc
    $fileDesc = $doc->createElement("fileDesc");
    $teiheader->appendChild($fileDesc);
    // OUTPUT: title
    $titleStmt = $doc->createElement("titleStmt");
    $fileDesc->appendChild($titleStmt);
    $title = $doc->createElement("title");

```

```

$title->appendChild($doc->createTextNode($method_title));
$titleStmt->appendChild($title);
// OUTPUT: publicationStmt
$publicationStmt = $doc->createElement("publicationStmt");
$fileDesc->appendChild($publicationStmt);
$authority = $doc->createElement("authority");
$authority->appendChild($doc->createTextNode($authority1));
$publicationStmt->appendChild($authority);
$date = $doc->createElement("date");
$date->appendChild($doc->createTextNode($today));
$publicationStmt->appendChild($date);
// OUTPUT: sourceDesc
$sourceDesc = $doc->createElement("sourceDesc");
$fileDesc->appendChild($sourceDesc);
for($m=0;$m<count($bibl_ref);$m++){
    $bibl = $doc->createElement("bibl");
    $bibl->appendChild($doc->createTextNode($bibl_array[$bibl_ref[$m]]));
    $sourceDesc->appendChild($bibl);
    $bibl->setAttribute("n", $bibl_ref[$m]);
}
// OUTPUT: encodingDesc
$encodingDesc = $doc->createElement("encodingDesc");
$teiheader->appendChild($encodingDesc);
$ab5 = $doc->createElement("ab");
$ab5->appendChild($doc->createTextNode($encoding));
$encodingDesc->appendChild($ab5);
// OUTPUT: profileDesc
$profileDesc = $doc->createElement("profileDesc");
$teiheader->appendChild($profileDesc);
$textClass = $doc->createElement("textClass");
$profileDesc->appendChild($textClass);
$catRef = $doc->createElement("catRef");
$textClass->appendChild($catRef);
$catRef->setAttribute("target", 'List of '.$list_of.' from '.$source);
// OUTPUT: langUsage
$langUsage = $doc->createElement("langUsage");
$profileDesc->appendChild($langUsage);
for($m=0;$m<count($lang_list);$m++){
    $language = $doc->createElement("language");
    $language->appendChild($doc->createTextNode($langs[$lang_list[$m]]));
    $langUsage->appendChild($language);
    $language->setAttribute("ident", $lang_list[$m]);
} // END: for($m=0;$m<count($lang_list);$m++){
// OUTPUT: Text
$text = $doc->createElement("text");
$tei->appendChild($text);
$body = $doc->createElement("body");
$text->appendChild($body);
    $list = $doc->createElement("list");
$body->appendChild($list);
$list->setAttribute("type", $list_of);
// SET: tablet directory
$dir="tablets";
// IF: there are documents in the directory
if($scans = scandir($dir, 0)){
    // SET: tablet count
    $count = count($scans);
    // FOR: each tablet
    for($i=2;$i<$count;$i++){
        // SET: file path
        $tablet=$dir . '/' . $scans[$i];
        // IF: the file is loaded
        if($xml_tablet = simplexml_load_file($tablet)){
            $xml_tablet->registerXPathNamespace('tei', 'http://www.tei-
c.org/ns/1.0');
            // SET: tablet number
            $idents=$xml_tablet->xpath('//tei:rs[@n="1"]');
            foreach($idents[0]->attributes() as $a => $b) {
                $$a=$b;
                $tabletNum=$key;
            } // END: of foreach($idents[0]->attributes() as $a => $b)
            // SET: word
            $words=$xml_tablet->xpath('//tei:w[@n]');

```

```

// FOR: each words tag
foreach($words as $word){
    unset($lemma);
    unset($n);
    unset($subtype);
    unset($rend);
    unset($rend1);
    $match = 'no';
    //For each attribute in the tag
    foreach($word[0]->attributes() as $a => $b) {
        // Put together values
        $$a=$b;
    } // END: of foreach($word[0]->attributes() as $a => $b)
    $lem = "" . $lemma[0] . "";
    $n1 = "" . $n[0] . "";
    $type1 = "" . $subtype[0] . "";
    $tnr = "" . $tabletNum[0] . "";
    $rend1 = "" . $rend[0] . "";
    // Putting together the array
    $l[$lem][0] = $lem;
    $l[$lem][1][$type1][0] = $type1;
    $l[$lem][1][$type1][1][$tnr][0] = $tnr;
    $l[$lem][1][$type1][1][$tnr][1] = $n1;
    $l[$lem][1][$type1][1][$tnr][2][] = $rend1;
    $l[$lem][5] = 'word';
} // END: of foreach($words as $word)
} // END: of if($xml_tablet = simplexml_load_file($tablet))
} // END: of for($i=3;$i<$count;$i++)
// FOR: each lemma
foreach ($l as $l1){
    // IF: the pattern fits and pattern is turned on.
    if($pattern){
        // IF: pattern matches lemma
        if(preg_match($pattern, $l1[0])){
            $match = 'yes';
            $matchType1 = 'yes';
        }else{
            $match = 'no';
            foreach ($l1[1] as $l2){
                if(preg_match($pattern, $l2[0])){
                    $match = 'yes';
                    $matchType1 = 'no';
                } // END: of if(preg_match($pattern, $l2[0]))
            } // END: of foreach ($l1[1] as $l2)
        } // END: of if(preg_match($pattern, $l1[0]))
    }else{
        $match = 'yes';
    } // END: of if($pattern)
    // IF: match is yes
    if($match == "yes"){
        $item = $doc->createElement("item");
        $list->appendChild($item);
        $item->setAttribute("n", $l1[0]);
        $ident1 = $doc->createElement("ident");
        $ident1->appendChild($doc->createTextNode($l1[0]));
        $item->appendChild($ident1);
        $ident1->setAttribute("type", 'lemma');
        foreach ($l1[1] as $l2){
            //IF: type match is yes
            if($matchType1=='yes'){
                $matchType='yes';
            }else{
                // IF: pattern is found in type
                if($pattern){
                    if(preg_match($pattern, $l2[0])){
                        $matchType = 'yes';
                    }else{
                        $matchType = 'no';
                    } // END: of if(preg_match($pattern, $l2[0]))
                }else{
                    $matchType = 'yes';
                } // END: of if($pattern)
            } // END: of if($matchType1=='yes')
        }
    }
}

```



```

// IF: empty
if($12[0]!=''){
  // IF: pattern matches type
  if($matchType == 'yes'){
    // OUTPUT: list
    $types_element = $doc->createElement("list");
    $item->appendChild($types_element);
    $types_element->setAttribute("type", "lemma_subtype");
    $type_element = $doc->createElement("item");
    $types_element->appendChild($type_element);
    $type_element->setAttribute("n", $12[0]);
    $typename_element = $doc->createElement("ident");
    $typename_element->appendChild($doc-
>createTextNode($12[0]));
    $type_element->appendChild($typename_element);
    $typename_element->setAttribute("type", "lemma_subtype");
    // OUTPUT: Tablet list
    $tablets_element = $doc->createElement("list");
    $type_element->appendChild($tablets_element);
    $tablets_element->setAttribute("type", "tablets");
    // FOR: each tablet
    foreach ($12[1] as $13){
      // OUTPUT: tablet items
      $tablet_element = $doc->createElement("item");
      $tablets_element->appendChild($tablet_element);
      $tablet_element->setAttribute("n", $13[0]);
      $tabletsnumber_element = $doc->createElement("ident");
      $tabletsnumber_element->appendChild($doc-
>createTextNode($13[0]));
      $tablet_element->appendChild($tabletsnumber_element);
      $tabletsnumber_element->setAttribute("type",
"tabletID");
      $number_element = $doc->createElement("num");
      $number_element->appendChild($doc-
>createTextNode($13[1]));
      $tablet_element->appendChild($number_element);
      $number_element->setAttribute("type", "amount");
      // OUTPUT: rendered word
      if($13[2][0]!=''){
        for($h=0;$h<$13[1];$h++){
          $render_el = $doc->createElement("w");
          $render_el->appendChild($doc-
>createTextNode($13[2][$h]));
          $tablet_element->appendChild($render_el);
          $render_el->setAttribute("n", $h+1);
        } // END: for($h=0;$h<$13[1];$h++){
      } // END: if($13[2][0]!=''){
    } // END: foreach ($12[1] as $13)
  } // END: if($matchType == 'yes')
}else{
  // OUTPUT: tablet list
  $tablets_element = $doc->createElement("list");
  $item->appendChild($tablets_element);
  $tablets_element->setAttribute("type", "tablets");
  // FOR: each tablet
  foreach ($12[1] as $13){
    // OUTPUT: tablet items
    $tablet_element = $doc->createElement("item");
    $tablets_element->appendChild($tablet_element);
    $tablet_element->setAttribute("n", $13[0]);
    $tabletsnumber_element = $doc->createElement("ident");
    $tabletsnumber_element->appendChild($doc-
>createTextNode($13[0]));
    $tablet_element->appendChild($tabletsnumber_element);
    $tabletsnumber_element->setAttribute("type", "tabletID");
    $number_element = $doc->createElement("num");
    $number_element->appendChild($doc-
>createTextNode($13[1]));
    $tablet_element->appendChild($number_element);
    $number_element->setAttribute("type", "amount");

```

```

        // OUTPUT: rendered word
        if($l3[2][0]!=''){
            for($h=0;$h<$l3[1];$h++){
                $render_el = $doc->createElement("w");
                $render_el->appendChild($doc-
>createTextNode($l3[2][$h]));
                $tablet_element->appendChild($render_el);
                $render_el->setAttribute("n", $h+1);
            } // END: for($h=0;$h<$l3[1];$h++){
        } // END: if($l3[2][0]!=''){
    } // END: of foreach ($l2[1] as $l3)
} // END: of if($l2[0]!='')
} // END: of foreach ($l1[1] as $l2)
} // END: of if($match == "yes")
} // END: of foreach ($l as $l1)
} // END: of if($scans = scandir($dir, 0))
//Return status failed, if any error found.
if ($error_code == 1) {
    $ab7 = $doc->createElement("ab");
    $ab7->appendChild($doc->createTextNode('failed'));
    $encodingDesc->appendChild($ab7);
    $ab7->setAttribute("type", "status");
}
// Return status passed, if no error found
else if ($error_code == 0) {
    $ab7 = $doc->createElement("ab");
    $ab7->appendChild($doc->createTextNode('passed'));
    $encodingDesc->appendChild($ab7);
    $ab7->setAttribute("type", "status");
}
return simplexml_load_string($doc->saveXML());
}

// ---- GET RS ---- //
/*
* Function which retrieves all the calender instances in the tablets
* OPTIONAL PARAMETER
* $pattern: If a pattern is sent with the function only the calender instance,
which fit the pattern will be returned.
* Pattern wildcard is *
* Alternative characters can be presented like this: {abc}
*/
// FUNCTION: get_rs
function get_rs($rs_cat, $pattern = FALSE) {
    // SET: rs category
    $cat = $rs_cat;
    $rs_array = array(
        'dates' => 'julian_calendar',
        'military' => 'militaryOfficial',
        'consul' => 'consul',
    );
    // SET: meta for header
    $method_title = 'APPELLO method: Get ' . $cat;
    $authority1 = 'Made available by Centre for the Study of Ancient Documents,
University of Oxford (http://www.csad.ox.ac.uk/) and Henriette Roued-Cunliffe
a part of her D.Phil at the University of Oxford (http://www.roued.com/e-doc/)';
    $today = date(d) . '-' . date(m) . '-' . date(Y);
    $bibl_array = array(
        'Tab.Vindol.I' => 'A.K.Bowman, J.D.Thomas, Vindolanda: the Latin writing-
tablets, Britannia Monograph 4. London, 1983 ',
        'Tab.Vindol.II' => 'A.K.Bowman, J.D.Thomas, The Vindolanda Writing-Tablets
(Tabulae Vindolandenses II). London, 1994',
        'Tab.Vindol.III' => 'A.K.Bowman, J.D.Thomas, The Vindolanda Writing Tablets
(Tabulae Vindolandenses III). London: British Museum Publications, 2003 ');
    $bibl_ref = array('Tab.Vindol.II', 'Tab.Vindol.III');
    $encoding = 'This file was produced by APPELLO version 0.2';
    $list_of = $cat;
    $source = "Vindolanda Writing Tablets";
    $langs = array(
        'eng' => 'English',
        'lat' => 'Latin',
        'grc' => 'Greek',
    );
}

```

```

$lang_list = array('eng', 'lat');
// SET: tablet number
$tabID_path = '//tei:rs[@n="1"]';
// SET: attribute for tablet ID
$tabID_attr = 'key';
// SET: top label
$stopID_array = array(
    'dates' => 'month',
    'military' => 'militaryOfficial',
    'consul' => 'consul',
);
// SET: bottom label
$bottomID_array = array(
    'dates' => 'date',
);
// SET: xPath to tag
$tag_path = '//tei:rs[@type="'. $rs_array[$rs_cat]. '"]';
// SET: local tablets directory
$dir="tablets";
// SET: Attributes for tags
$attr_top = 'nymRef';
$attr_bottom = 'rend';
$attr_nr = 'n';
$attr_key = 'key';
// IF: pattern
if($pattern){
    $search = array(
        '{',
        '}',
        '*',
    );
    $replace = array(
        '([',
        ']{1})',
        '([a-zA-Z]{1})',
    );
    $pattern = str_replace($search, $replace, $pattern);
    $pattern = '/' . $pattern . '/';
}
//SET: error_code=0 for Success / error_code=1 for Failure
$error_code = 0;
// SET: DOMDocument
$doc = new DOMDocument();
$doc->formatOutput = true;
// OUTPUT: TEI document
$tei = $doc->createElement("TEI");
$doc->appendChild($tei);
// OUTPUT: teiHeader
$teiheader = $doc->createElement("teiHeader");
$tei->appendChild($teiheader);
// OUTPUT: fileDesc
$fileDesc = $doc->createElement("fileDesc");
$teiheader->appendChild($fileDesc);
// OUTPUT: title
$titleStmt = $doc->createElement("titleStmt");
$fileDesc->appendChild($titleStmt);
$title = $doc->createElement("title");
$title->appendChild($doc->createTextNode($method_title));
$titleStmt->appendChild($title);
// OUTPUT: publicationStmt
$publicationStmt = $doc->createElement("publicationStmt");
$fileDesc->appendChild($publicationStmt);
$authority = $doc->createElement("authority");
$authority->appendChild($doc->createTextNode($authority1));
$publicationStmt->appendChild($authority);
$date = $doc->createElement("date");
$date->appendChild($doc->createTextNode($today));
$publicationStmt->appendChild($date);
// OUTPUT: sourceDesc
$sourceDesc = $doc->createElement("sourceDesc");
$fileDesc->appendChild($sourceDesc);
for($m=0;$m<count($bibl_ref);$m++){
    $bibl = $doc->createElement("bibl");

```

```

    $bibl->appendChild($doc->createTextNode($bibl_array[$bibl_ref[$m]]));
    $sourceDesc->appendChild($bibl);
    $bibl->setAttribute("n", $bibl_ref[$m]);
} // END: for($m=0;$m<count($bibl_ref);$m++){
// OUTPUT: encodingDesc
$encodingDesc = $doc->createElement("encodingDesc");
$teiheader->appendChild($encodingDesc);
$ab5 = $doc->createElement("ab");
$ab5->appendChild($doc->createTextNode($encoding));
$encodingDesc->appendChild($ab5);
// OUTPUT: profileDesc
$profileDesc = $doc->createElement("profileDesc");
$teiheader->appendChild($profileDesc);
$textClass = $doc->createElement("textClass");
$profileDesc->appendChild($textClass);
$catRef = $doc->createElement("catRef");
$textClass->appendChild($catRef);
$catRef->setAttribute("target", 'List of '.$list_of.' from '.$source);
// OUTPUT: langUsage
$langUsage = $doc->createElement("langUsage");
$profileDesc->appendChild($langUsage);
for($m=0;$m<count($lang_list);$m++){
    $language = $doc->createElement("language");
    $language->appendChild($doc->createTextNode($langs[$lang_list[$m]]));
    $langUsage->appendChild($language);
    $language->setAttribute("ident", $lang_list[$m]);
} // END: for($m=0;$m<count($lang_list);$m++){
// OUTPUT: Text
$text = $doc->createElement("text");
$tei->appendChild($text);
// OUTPUT: Body
$body = $doc->createElement("body");
$text->appendChild($body);
// OUTPUT: List
$list = $doc->createElement("list");
$body->appendChild($list);
// OUTPUT: Type of list
$list->setAttribute("type", $list_of);
// IF: Scan tablets directory
if($scans = scandir($dir, 0)){
    // FOR: tablets
    for($i=2;$i<count($scans);$i++){
        $tablet=$dir . '/' . $scans[$i];
        // IF: tablet loads
        if($xml_tablet = simplexml_load_file($tablet)){
            $xml_tablet->registerXPathNamespace('tei', 'http://www.tei-
c.org/ns/1.0');
            // SET: tablet nr
            $idents=$xml_tablet->xpath($tabID_path);
            foreach($idents[0]->attributes() as $a => $b) {
                $$a=$b;
                $tabletNum=$$tabID_attr;
            } // END: of foreach($idents[0]->attributes() as $a => $b)
            // OUTPUT: tags
            $tags=$xml_tablet->xpath($tag_path);
            // FOR: each tag
            foreach($tags as $tag){
                // UNSET: attributes
                unset($$attr_top);
                unset($$attr_bottom);
                unset($$attr_nr);
                unset($$attr_key);
                // SET: pattern matches to 'no' for default
                $match = 'no';
                $matchType = 'no';
                // FOR: each attribute in the tag
                foreach($tag[0]->attributes() as $a => $b) {
                    $$a=$b;
                } // END: of foreach($tag[0]->attributes() as $a => $b)
                $top1 = $$attr_top;
                $bottom1 = $$attr_bottom;
                $n1 = $$attr_nr;
                $id1 = $$attr_key;

```

```

$bottom = "" . $bottom1[0] . "";
$nr = "" . $n1[0] . "";
$id = "" . $id1[0] . "";
if($stop1[0]!=''){
$stop = "" . $stop1[0] . "";
}else{
$stop = "" . $id1[0] . "";
}
$tnr = "" . $tabletNum[0] . "";
$l[$stop][0] = $stop;
$l[$stop][2] = $id;;
$l[$stop][1][$bottom][0] = $bottom;
$l[$stop][1][$bottom][2] = $id;
$l[$stop][1][$bottom][1][$tnr][0] = $tnr;
$l[$stop][1][$bottom][1][$tnr][1] = $nr;
$l[$stop][5] = $rs_cat;
} // END: of foreach($tags as $tag)
} // END: of if($xml_tablet = simplexml_load_file($tablet))
} // END: of for($i=2;$i<$count;$i++)
// FOR: each item in the array
foreach ($l as $l1){
// IF: the pattern fits and pattern is turned on.
if($pattern){
// IF: pattern matches month
if(preg_match($pattern, $l1[2])){
$match = 'yes';
$matchType1 = 'yes';
}else{
$match = 'no';
foreach ($l1[1] as $l2){
if(preg_match($pattern, $l2[2])){
$match = 'yes';
$matchType1 = 'no';
} // END: of if(preg_match($pattern, $l2[0]))
} // END: of foreach ($l1[1] as $l2)
} // END: of if(preg_match($pattern, $l1[0]))
}else{
$match = 'yes';
} // END: of if($pattern)
if($match == "yes"){
$item = $doc->createElement("item");
$list->appendChild($item);
$item->setAttribute("n", $l1[0]);
$ident1 = $doc->createElement("ident");
$ident1->appendChild($doc->createTextNode($l1[0]));
$item->appendChild($ident1);
$ident1->setAttribute("type", $stopID_array[$rs_cat]);
foreach ($l1[1] as $l2){
if($matchType1=='yes'){
$matchType='yes';
}else{
if($pattern){
if(preg_match($pattern, $l2[2])){
$matchType = 'yes';
}else{
$matchType = 'no';
} // END: of if(preg_match($pattern, $l2[0]))
}else{
$matchType = 'yes';
} // END: of if($pattern)
} // END: of if($matchType1=='yes')
if($l2[0]!=''){
if($matchType == 'yes'){
$types_element = $doc->createElement("list");
$item->appendChild($types_element);
$types_element->setAttribute("type",
$bottomID_array[$rs_cat]);
$type_element = $doc->createElement("item");
$types_element->appendChild($type_element);
$type_element->setAttribute("n", $l2[0]);
$type_element->setAttribute("xml:id", $l2[2]);
$typename_element = $doc->createElement("ident");
$typename_element->appendChild($doc->createTextNode($l2[0]));

```

```

$type_element->appendChild($typename_element);
$typename_element->setAttribute("type",
$bottomID_array[$rs_cat]);
$tablets_element = $doc->createElement("list");
$type_element->appendChild($tablets_element);
$tablets_element->setAttribute("type", "tablets");
foreach ($l2[1] as $l3){
    // OUTPUT: Tablet number
    $tablet_element = $doc->createElement("item");
    $tablets_element->appendChild($tablet_element);
    $tablet_element->setAttribute("n", $l3[0]);
    // OUTPUT: TabletNumber
    $tabletsnumber_element = $doc->createElement("ident");
    $tabletsnumber_element->appendChild($doc-
>createTextNode($l3[0]));
    $tablet_element->appendChild($tabletsnumber_element);
    $tabletsnumber_element->setAttribute("type", "tabletID");
    // OUTPUT: Number
    $number_element = $doc->createElement("num");
    $number_element->appendChild($doc->createTextNode($l3[1]));
    $tablet_element->appendChild($number_element);
    $number_element->setAttribute("type", "amount");
} // END: of foreach ($l2[1] as $l3)
} // END: of if($matchType == 'yes')
}else{
    $item->setAttribute("xml:id", $l1[2]);
    // OUTPUT: Tablets
    $tablets_element = $doc->createElement("list");
    $item->appendChild($tablets_element);
    $tablets_element->setAttribute("type", "tablets");
    foreach ($l2[1] as $l3){
        // OUTPUT: Tablet number
        $tablet_element = $doc->createElement("item");
        $tablets_element->appendChild($tablet_element);
        $tablet_element->setAttribute("n", $l3[0]);
        // OUTPUT: TabletNumber
        $tabletsnumber_element = $doc->createElement("ident");
        $tabletsnumber_element->appendChild($doc-
>createTextNode($l3[0]));
        $tablet_element->appendChild($tabletsnumber_element);
        $tabletsnumber_element->setAttribute("type", "tabletID");
        // OUTPUT: Number
        $number_element = $doc->createElement("num");
        $number_element->appendChild($doc->createTextNode($l3[1]));
        $tablet_element->appendChild($number_element);
        $number_element->setAttribute("type", "amount");
    } // END: of foreach ($l2[1] as $l3)
    } // END: of if($l2[0]!='')
    } // END: of foreach ($l1[1] as $l2)
    } // END: of if($match == "yes")
    } // END: of foreach ($l as $l1)
} // END: of if($scans = scandir($dir, 0))
//Return status failed, if any error found.
if ($error_code == 1) {
    $ab7 = $doc->createElement("ab");
    $ab7->appendChild($doc->createTextNode('failed'));
    $encodingDesc->appendChild($ab7);
    $ab7->setAttribute("type", "status");
}
// Return status passed, if no error found
else if ($error_code == 0) {
    $ab7 = $doc->createElement("ab");
    $ab7->appendChild($doc->createTextNode('passed'));
    $encodingDesc->appendChild($ab7);
    $ab7->setAttribute("type", "status");
}
return simplexml_load_string($doc->saveXML());
} // END: function

// ---- GET PERSON ---- //
/*
* Function which retrieves all the persons in the tablets
* OPTIONAL PARAMETER

```

```

* $pattern: If a pattern is sent with the function only the persons which fit the
pattern will be returned.
* Pattern wildcard is *
* Alternative characters can be presented like this: {abc}
*/
// FUNCTION: get_person
function get_person($pattern=FALSE) {
    // SET: category to person
    $cat = 'person';
    // SET: metadata for header
    $method_title = 'APPELLO method: Get ' . $cat;
    $authority1 = 'Made available by Centre for the Study of Ancient Documents,
University of Oxford (http://www.csad.ox.ac.uk/) and Henriette Roued-Cunliffe as
a part of her D.Phil at the University of Oxford (http://www.roued.com/e-doc/)';
    $today = date(d) . '-' . date(m) . '-' . date(Y);
    $bibl_array = array(
        'Tab.Vindol.I' => 'A.K.Bowman, J.D.Thomas, Vindolanda: the Latin writing-
tablets, Britannia Monograph 4. London, 1983 ',
        'Tab.Vindol.II' => 'A.K.Bowman, J.D.Thomas, The Vindolanda Writing-Tablets
(Tabulae Vindolandenses II). London, 1994',
        'Tab.Vindol.III' => 'A.K.Bowman, J.D.Thomas, The Vindolanda Writing Tablets
(Tabulae Vindolandenses III). London: British Museum Publications, 2003 ');
    $bibl_ref = array('Tab.Vindol.II', 'Tab.Vindol.III');
    $encoding = 'This file was produced by APPELLO version 0.2';
    $list_of = $cat;
    $source ="Vindolanda Writing Tablets";
    $langs = array(
        'eng' => 'English',
        'lat' => 'Latin',
        'grc' => 'Greek',
    );
    // SET: Array of languages in this document
    $lang_list = array('eng', 'lat');
    // SET: xPath for tablet number
    $tabID_path = '//tei:rs[@n="1"]';
    // SET: attribute for tablet ID
    $tabID_attr = 'key';
    // SET: top label
    $topID_array = array(
        'person' => 'person',
    );
    // SET: bottom label
    $bottomID_array = array(
    );
    // SET: xPath to tag
    $tag_path = '//tei:persName[@nymRef]';
    // SET: local tablets directory
    $dir="tablets";
    // SET: Attributes for tags
    $attr_top = 'nymRef';
    $attr_nr = 'n';
    $attr_key = 'key';
    $attr_extra = 'rend';
    $extra_array = array(
        'person' => 'profession',
    );
    $attr_orig = 'rendition';
    // IF: there is a pattern - clean it
    if($pattern){
        $search = array(
            '{',
            '}',
            '*',
        );
        $replace = array(
            '([',
            ']{1})',
            '([a-zA-Z]{1})',
        );
        $pattern = str_replace($search, $replace, $pattern);
        $pattern = '/' . $pattern . '/';
    }
    //SET: error_code=0 for Success / error_code=1 for Failure

```

```

$error_code = 0;
// SET: DOMDocument
$doc = new DOMDocument();
$doc->formatOutput = true;
// OUTPUT: TEI document
$tei = $doc->createElement("TEI");
$doc->appendChild($tei);
// OUTPUT: teiHeader
$teiheader = $doc->createElement("teiHeader");
$tei->appendChild($teiheader);
// OUTPUT: fileDesc
$fileDesc = $doc->createElement("fileDesc");
$teiheader->appendChild($fileDesc);
// OUTPUT: title
$titleStmt = $doc->createElement("titleStmt");
$fileDesc->appendChild($titleStmt);
$title = $doc->createElement("title");
$title->appendChild($doc->createTextNode($method_title));
$titleStmt->appendChild($title);
// OUTPUT: publicationStmt
$publicationStmt = $doc->createElement("publicationStmt");
$fileDesc->appendChild($publicationStmt);
$authority = $doc->createElement("authority");
$authority->appendChild($doc->createTextNode($authority1));
$publicationStmt->appendChild($authority);
$date = $doc->createElement("date");
$date->appendChild($doc->createTextNode($today));
$publicationStmt->appendChild($date);
// OUTPUT: sourceDesc
$sourceDesc = $doc->createElement("sourceDesc");
$fileDesc->appendChild($sourceDesc);
for($m=0;$m<count($bibl_ref);$m++){
    $bibl = $doc->createElement("bibl");
    $bibl->appendChild($doc->createTextNode($bibl_array[$bibl_ref[$m]]));
    $sourceDesc->appendChild($bibl);
    $bibl->setAttribute("n", $bibl_ref[$m]);
}
// OUTPUT: encodingDesc
$encodingDesc = $doc->createElement("encodingDesc");
$teiheader->appendChild($encodingDesc);
$ab5 = $doc->createElement("ab");
$ab5->appendChild($doc->createTextNode($encoding));
$encodingDesc->appendChild($ab5);
// OUTPUT: profileDesc
$profileDesc = $doc->createElement("profileDesc");
$teiheader->appendChild($profileDesc);
$textClass = $doc->createElement("textClass");
$profileDesc->appendChild($textClass);
$catRef = $doc->createElement("catRef");
$textClass->appendChild($catRef);
$catRef->setAttribute("target", 'List of '.$list_of.' from '.$source);
// OUTPUT: langUsage
$langUsage = $doc->createElement("langUsage");
$profileDesc->appendChild($langUsage);
for($m=0;$m<count($lang_list);$m++){
    $language = $doc->createElement("language");
    $language->appendChild($doc->createTextNode($langs[$lang_list[$m]]));
    $langUsage->appendChild($language);
    $language->setAttribute("ident", $lang_list[$m]);
}
// OUTPUT: Text
$text = $doc->createElement("text");
$tei->appendChild($text);
// OUTPUT: Body
$body = $doc->createElement("body");
$text->appendChild($body);
// OUTPUT: List
$list = $doc->createElement("list");
$body->appendChild($list);
// OUTPUT: Type of list
$list->setAttribute("type", $list_of);
// IF: Scan tablets directory
if($scans = scandir($dir, 0)){

```



```

// FOR: each tablet
for($i=2;$i<count($scans);$i++){
  // SET: tablet connection
  $tablet=$dir . '/' . $scans[$i];
  // IF: connected
  if($xml_tablet = simplexxml_load_file($tablet)){
    $xml_tablet->registerXPathNamespaces('tei', 'http://www.tei-
c.org/ns/1.0');
    $idents=$xml_tablet->xpath($tabID_path);
    // FOR: each tablet number
    foreach($idents[0]->attributes() as $a => $b) {
      $$a=$b;
      $tabletNum=$$tabID_attr;
    } // END: of foreach($idents[0]->attributes() as $a => $b)
    // SET: tags
    $tags=$xml_tablet->xpath($tag_path);
    // FOR: each tag
    foreach($tags as $tag){
      // UNSET: attributes
      unset($$attr_top);
      unset($$attr_bottom);
      unset($$attr_nr);
      unset($$attr_key);
      unset($$attr_extra);
      // SET: pattern matches to 'no' for default
      $match = 'no';
      $matchType = 'no';
      // FOR: each attribute in the tag
      foreach($tag[0]->attributes() as $a => $b) {
        // Make the values
        $$a=$b;
      } // END: of foreach($tag[0]->attributes() as $a => $b)
      // SET: attribute settings
      $top1 = $$attr_top;
      $bottom1 = $$attr_bottom;
      $n1 = $$attr_nr;
      $id1 = $$attr_key;
      $extral = $$attr_extra;
      $orig1 = $$attr_orig;
      $bottom = "" . $bottom1[0] . "";
      $nr = "" . $n1[0] . "";
      $id = "" . $id1[0] . "";
      // IF: top attribute
      if($top1[0]!=''){
        $top = "" . $top1[0] . "";
      }else{
        $top = "" . $id1[0] . "";
      } // END: if($top1[0]!=''){
      $tnr = "" . $tabletNum[0] . "";
      // IF: attribute extra
      if($extral[0]!=''){
        $extra = "" . $extral[0] . "";
      }else{
        $extra='';
      } // END: if($extral[0]!=''){
      // IF: original attribute
      if($orig1[0]!=''){
        $orig = "" . $orig1[0] . "";
      }else{
        $orig='';
      } // END: if($orig1[0]!=''){
      // SET: attribute array
      $l[$top][0] = $top;
      $l[$top][2] = strtolower($id);
      $l[$top][3] = strtolower($extra);
      $l[$top][1][$bottom][0] = $bottom;
      $l[$top][1][$bottom][2] = $id;
      $l[$top][1][$bottom][1][$tnr][0] = $tnr;
      $l[$top][1][$bottom][1][$tnr][1] = $nr;
      $l[$top][1][$bottom][1][$tnr][2][] = $orig;
      $l[$top][5] = $cat;
    } // END: of foreach($tags as $tag)
  } // END: of if($xml_tablet = simplexxml_load_file($tablet))
}

```

```

} // END: of for($i=3;$i<$count;$i++)
// FOR: each person
foreach ($l as $l1){
    // IF: the pattern fits and pattern is turned on.
    if($pattern){
        // IF: pattern matches person
        if(preg_match($pattern, $l1[2])){
            $match = 'yes';
            $matchType1 = 'yes';
        }else{
            $match = 'no';
            // FOR: each pattern in person
            foreach ($l1[1] as $l2){
                if(preg_match($pattern, $l2[2])){
                    $match = 'yes';
                    $matchType1 = 'no';
                } // END: of if(preg_match($pattern, $l2[0]))
            } // END: of foreach ($l1[1] as $l2)
        } // END: of if(preg_match($pattern, $l1[0]))
    }else{
        $match = 'yes';
    } // END: of if($pattern)
    // IF: person matches
    if($match == "yes"){
        $item = $doc->createElement("item");
        $list->appendChild($item);
        $item->setAttribute("n", $l1[0]);
        $ident1 = $doc->createElement("ident");
        $ident1->appendChild($doc->createTextNode($l1[0]));
        $item->appendChild($ident1);
        $ident1->setAttribute("type", $stopID_array[$cat]);

        // IF: extra Values
        if($l1[3]){
            $ident2 = $doc->createElement("ident");
            $ident2->appendChild($doc->createTextNode($l1[3]));
            $item->appendChild($ident2);
            $ident2->setAttribute("type", $extra_array[$cat]);
        }
        // FOR: each type match
        foreach ($l1[1] as $l2){
            if($matchType1=='yes'){
                $matchType='yes';
            }else{
                if($pattern){
                    // IF: pattern matches type
                    if(preg_match($pattern, $l2[2])){
                        $matchType = 'yes';
                    }else{
                        $matchType = 'no';
                    } // END: of if(preg_match($pattern, $l2[0]))
                }else{
                    $matchType = 'yes';
                } // END: of if($pattern)
            } // END: of if($matchType1=='yes')
            // IF: there is a type
            if($l2[0]!=''){
                // IF: type is a match
                if($matchType == 'yes'){
                    // OUTPUT: list
                    $types_element = $doc->createElement("list");
                    $item->appendChild($types_element);
                    $types_element->setAttribute("type", $bottomID_array[$cat]);
                    $type_element = $doc->createElement("item");
                    $types_element->appendChild($type_element);
                    $type_element->setAttribute("n", $l2[0]);
                    $type_element->setAttribute("xml:id", $l2[2]);
                    $typename_element = $doc->createElement("ident");
                    $typename_element->appendChild($doc->createTextNode($l2[0]));
                    $type_element->appendChild($typename_element);
                    $typename_element->setAttribute("type",
$bottomID_array[$cat]);
                    // OUTPUT: Tablets

```

```

$tablets_element = $doc->createElement("list");
$type_element->appendChild($tablets_element);
$tablets_element->setAttribute("type", "tablets");
foreach ($l2[1] as $l3){
    // OUTPUT: Tablet number
    $tablet_element = $doc->createElement("item");
    $tablets_element->appendChild($tablet_element);
    $tablet_element->setAttribute("n", $l3[0]);
    // OUTPUT: TabletNumber
    $tabletsnumber_element = $doc->createElement("ident");
    $tabletsnumber_element->appendChild($doc-
>createTextNode($l3[0]));
    $tablet_element->appendChild($tabletsnumber_element);
    $tabletsnumber_element->setAttribute("type", "tabletID");
    // OUTPUT: Number
    $number_element = $doc->createElement("num");
    $number_element->appendChild($doc->createTextNode($l3[1]));
    $tablet_element->appendChild($number_element);
    $number_element->setAttribute("type", "amount");
    if($l3[2][0]!=''){
        for($h=0;$h<$l3[1];$h++){
            $render_el = $doc->createElement("w");
            $render_el->appendChild($doc-
>createTextNode($l3[2][$h]));
            $tablet_element->appendChild($render_el);
            $render_el->setAttribute("n", $h+1);
        } // END: for($h=0;$h<$l3[1];$h++){
    } // END: if($l3[2][0]!=''){
} // END: of foreach ($l2[1] as $l3)
} // END: of if($matchType == 'yes')
// IF: there is no type
}else{
    $item->setAttribute("xml:id", $l1[2]);
    // OUTPUT: Tablets
    $tablets_element = $doc->createElement("list");
    $item->appendChild($tablets_element);
    $tablets_element->setAttribute("type", "tablets");
    foreach ($l2[1] as $l3){
        // OUTPUT: Tablet number
        $tablet_element = $doc->createElement("item");
        $tablets_element->appendChild($tablet_element);
        $tablet_element->setAttribute("n", $l3[0]);
        // OUTPUT: TabletNumber
        $tabletsnumber_element = $doc->createElement("ident");
        $tabletsnumber_element->appendChild($doc-
>createTextNode($l3[0]));
        $tablet_element->appendChild($tabletsnumber_element);
        $tabletsnumber_element->setAttribute("type", "tabletID");
        // OUTPUT: Number
        $number_element = $doc->createElement("num");
        $number_element->appendChild($doc->createTextNode($l3[1]));
        $tablet_element->appendChild($number_element);
        $number_element->setAttribute("type", "amount");
        if($l3[2][0]!=''){
            for($h=0;$h<$l3[1];$h++){
                $render_el = $doc->createElement("w");
                $render_el->appendChild($doc-
>createTextNode($l3[2][$h]));
                $tablet_element->appendChild($render_el);
                $render_el->setAttribute("n", $h+1);
            } // END: for($h=0;$h<$l3[1];$h++){
        } // END: if($l3[2][0]!=''){
    } // END: of foreach ($l2[1] as $l3)
} // END: of if($l2[0]!='')
} // END: of foreach ($l1[1] as $l2)
} // END: of if($match == "yes")
} // END: of foreach ($l as $l1)
} // END: of if($scans = scandir($dir, 0))
//Return status failed, if any error found.
if ($error_code == 1) {
    $ab7 = $doc->createElement("ab");
    $ab7->appendChild($doc->createTextNode('failed'));
    $encodingDesc->appendChild($ab7);
}

```

```

    $ab7->setAttribute("type", "status");
} // END: if ($error_code == 1) {
// Return status passed, if no error found
else if ($error_code == 0) {
    $ab7 = $doc->createElement("ab");
    $ab7->appendChild($doc->createTextNode('passed'));
    $encodingDesc->appendChild($ab7);
    $ab7->setAttribute("type", "status");
} // END: else if ($error_code == 0) {
return simplexml_load_string($doc->saveXML());
} // END: function get_person($pattern=FALSE) {

// ---- GET GEOGRAPHICAL NAME ---- //
/*
* Function which retrieves all the geographical names in the tablets
* OPTIONAL PARAMETER
* $pattern: If a pattern is sent with the function only the geographical names
which fit the pattern will be returned.
* Pattern wildcard is *
* Alternative characters can be presented like this: {abc}
*/
// FUNCTION: get_geog()
function get_geog($pattern=FALSE) {
    // SET: category
    $cat = 'placename';
    // SET: title
    $method_title = 'APPELLO method: Get ' . $cat;
    // SET: authority
    $authority1 = 'Made available by Centre for the Study of Ancient Documents,
University of Oxford (http://www.csad.ox.ac.uk/) and Henriette Roued-Cunliffe as
a part of her D.Phil at the University of Oxford (http://www.roued.com/e-doc/);
    // SET: todays date
    $today = date(d) . '-' . date(m) . '-' . date(Y);
    // SET: Array of possible sources with bibliographic references
    $bibl_array = array(
        'Tab.Vindol.I' => 'A.K.Bowman, J.D.Thomas, Vindolanda: the Latin writing-
tablets, Britannia Monograph 4. London, 1983 ',
        'Tab.Vindol.II' => 'A.K.Bowman, J.D.Thomas, The Vindolanda Writing-Tablets
(Tabulae Vindolandenses II). London, 1994',
        'Tab.Vindol.III' => 'A.K.Bowman, J.D.Thomas, The Vindolanda Writing Tablets
(Tabulae Vindolandenses III). London: British Museum Publications, 2003 ');
    // SET: Array of sources in this document
    $bibl_ref = array('Tab.Vindol.II', 'Tab.Vindol.III');
    // SET: encoding
    $encoding = 'This file was produced by APPELLO version 0.2';
    // SET: category list
    $list_of = $cat;
    // SET: source
    $source = "Vindolanda Writing Tablets";
    // SET: Array of possible languages
    $langs = array(
        'eng' => 'English',
        'lat' => 'Latin',
        'grc' => 'Greek',
    );
    // SET: Array of languages in this document
    $lang_list = array('eng', 'lat');
    // SET: xPath for tablet number
    $tabID_path = '//tei:rs[@n="1"]';
    // SET: attribute for tablet ID
    $tabID_attr = 'key';
    // SET: top label
    $topID_array = array(
        'placename' => 'placename',
    );
    // SET: bottom label
    $bottomID_array = array(
    );
    // SET: xPath to tag
    $tag_path = '//tei:placeName[@key]';
    // SET: local tablets directory
    $dir="tablets";

```

```

// SET: Attributes for tags
$attr_top = 'nymRef';
$attr_nr = 'n';
$attr_key = 'key';
$extra_array = array(
);
$attr_orig = 'rend';
// IF: there is a pattern - clean it
if($pattern){
    $search = array(
        '{',
        '}',
        '*',
    );
    $replace = array(
        '([',
        ']{1})',
        '([a-zA-Z]{1})',
    );
    $pattern = str_replace($search, $replace, $pattern);
    $pattern = '/' . $pattern . '/';
} // END: if($pattern){
// SET: error_code=0 for Success / error_code=1 for Failure
$error_code = 0;
// SET: document
$doc = new DOMDocument();
$doc->formatOutput = true;
// SET: Response Element - building a TEI document
$tei = $doc->createElement("TEI");
$doc->appendChild($tei);
// SET: teiHeader
$teiheader = $doc->createElement("teiHeader");
$tei->appendChild($teiheader);
// SET: fileDesc
$fileDesc = $doc->createElement("fileDesc");
$teiheader->appendChild($fileDesc);
// SET: title
$titleStmt = $doc->createElement("titleStmt");
$fileDesc->appendChild($titleStmt);
$title = $doc->createElement("title");
$title->appendChild($doc->createTextNode($method_title));
$titleStmt->appendChild($title);
// SET: publicationStmt
$publicationStmt = $doc->createElement("publicationStmt");
$fileDesc->appendChild($publicationStmt);
$authority = $doc->createElement("authority");
$authority->appendChild($doc->createTextNode($authority1));
$publicationStmt->appendChild($authority);
$date = $doc->createElement("date");
$date->appendChild($doc->createTextNode($today));
$publicationStmt->appendChild($date);
// SET: sourceDesc
$sourceDesc = $doc->createElement("sourceDesc");
$fileDesc->appendChild($sourceDesc);
for($m=0;$m<count($bibl_ref);$m++){
    $bibl = $doc->createElement("bibl");
    $bibl->appendChild($doc->createTextNode($bibl_array[$bibl_ref[$m]]));
    $sourceDesc->appendChild($bibl);
    $bibl->setAttribute("n", $bibl_ref[$m]);
} // END: for($m=0;$m<count($bibl_ref);$m++){
// SET: encodingDesc
$encodingDesc = $doc->createElement("encodingDesc");
$teiheader->appendChild($encodingDesc);
$ab5 = $doc->createElement("ab");
$ab5->appendChild($doc->createTextNode($encoding));
$encodingDesc->appendChild($ab5);
// SET: profileDesc
$profileDesc = $doc->createElement("profileDesc");
$teiheader->appendChild($profileDesc);
$textClass = $doc->createElement("textClass");
$profileDesc->appendChild($textClass);
$catRef = $doc->createElement("catRef");
$textClass->appendChild($catRef);

```

```

$catRef->setAttribute("target", 'List of '.$list_of.' from '.$source);
// SET: langUsage
$langUsage = $doc->createElement("langUsage");
$profileDesc->appendChild($langUsage);
for($m=0;$m<count($lang_list);$m++){
    $language = $doc->createElement("language");
    $language->appendChild($doc->createTextNode($langs[$lang_list[$m]]));
    $langUsage->appendChild($language);
    $language->setAttribute("ident", $lang_list[$m]);
}
// OUTPUT: Text
$text = $doc->createElement("text");
$tei->appendChild($text);
// OUTPUT: Body
$body = $doc->createElement("body");
$text->appendChild($body);
// OUTPUT: List
$list = $doc->createElement("list");
$body->appendChild($list);
// OUTPUT: Type of list
$list->setAttribute("type", $list_of);
// IF: it is possible to scan tablets directory
if($scans = scandir($dir, 0)){
    // FOR: Get data
    for($i=2;$i<count($scans);$i++){
        // SET: Connection to each tablet
        $tablet=$dir . '/' . $scans[$i];
        // IF: connected
        if($xml_tablet = simplexml_load_file($tablet)){
            // OUTPUT: tei namespace
            $xml_tablet->registerXPathNamespace('tei', 'http://www.tei-
c.org/ns/1.0');
            // SET: tablet nr
            $idents=$xml_tablet->xpath($tabID_path);
            foreach($idents[0]->attributes() as $a => $b) {
                $$a=$b;
                $tabletNum=$$tabID_attr;
            } // END: of foreach($idents[0]->attributes() as $a => $b)
            // SET: Tags
            $tags=$xml_tablet->xpath($tag_path);
            // FOR: each tag
            foreach($tags as $tag){
                // UNSET: attributes
                unset($$attr_top);
                unset($$attr_bottom);
                unset($$attr_nr);
                unset($$attr_key);
                unset($$attr_extra);
                // SET: pattern matches to 'no' for default
                $match = 'no';
                $matchType = 'no';
                // FOR: each attribute in the tag
                foreach($tag[0]->attributes() as $a => $b) {
                    // SET: values
                    $$a=$b;
                } // END: of foreach($tag[0]->attributes() as $a => $b)
                // SET: attributes
                $top1 = $$attr_top;
                $bottom1 = $$attr_bottom;
                $n1 = $$attr_nr;
                $id1 = $$attr_key;
                $extral = $$attr_extra;
                $orig1 = $$attr_orig;
                $bottom = "" . $bottom1[0] . "";
                $nr = "" . $n1[0] . "";
                $id = "" . $id1[0] . "";
                // IF: top attribute
                if($top1[0]!=''){
                    $top = "" . $top1[0] . "";
                }else{
                    $top = "" . $id1[0] . "";
                } // END: if($top1[0]!=''){
                $tnr = "" . $tabletNum[0] . "";
            }
        }
    }
}

```

```

// IF: attribute extra
if($extral[0]!=''){
    $extra = "" . $extral[0] . "";
}else{
    $extra='';
} // END: if($extral[0]!=''){
// IF: original attribute
if($origl[0]!=''){
    $orig = "" . $origl[0] . "";
}else{
    $orig='';
} // END: if($origl[0]!=''){
// SET: attribute array
$l[$stop][0] = $stop;
$l[$stop][2] = strtolower($id);
$l[$stop][3] = strtolower($extra);
$l[$stop][1][$bottom][0] = $bottom;
$l[$stop][1][$bottom][2] = $id;
$l[$stop][1][$bottom][1][$tnr][0] = $tnr;
$l[$stop][1][$bottom][1][$tnr][1] = $nr;
$l[$stop][1][$bottom][1][$tnr][2][] = $orig;
$l[$stop][5] = $cat;
} // END: of foreach($tags as $tag)
} // END: of if($xml_tablet = simplexml_load_file($tablet))
} // END: of for($i=3;$i<$count;$i++)
// FOR: each array part
foreach ($l as $l1){
    // IF: the pattern fits and pattern is turned on.
    if($pattern){
        // IF: pattern matches
        if(preg_match($pattern, $l1[2])){
            $match = 'yes';
            $matchType1 = 'yes';
        }else{
            $match = 'no';
            // IF: pattern is bottom tag
            foreach ($l1[1] as $l2){
                if(preg_match($pattern, $l2[2])){
                    $match = 'yes';
                    $matchType1 = 'no';
                } // END: of if(preg_match($pattern, $l2[0]))
            } // END: of foreach ($l1[1] as $l2)
        } // END: of if(preg_match($pattern, $l1[0]))
    }else{
        // IF: to pattern not turned on print all.
        $match = 'yes';
    } // END: of if($pattern)
    // IF: match
    if($match == "yes"){
        // OUTPUT: item
        $item = $doc->createElement("item");
        $list->appendChild($item);
        $item->setAttribute("n", $l1[0]);
        $ident1 = $doc->createElement("ident");
        $ident1->appendChild($doc->createTextNode($l1[0]));
        $item->appendChild($ident1);
        $ident1->setAttribute("type", $stopID_array[$cat]);
        // IF: extra Values
        if($l1[3]){
            $ident2 = $doc->createElement("ident");
            $ident2->appendChild($doc->createTextNode($l1[3]));
            $item->appendChild($ident2);
            $ident2->setAttribute("type", $extra_array[$cat]);
        } // END: if($l1[3])
        // FOR: pattern matches type
        foreach ($l1[1] as $l2){
            if($matchType1=='yes'){
                $matchType='yes';
            }else{
                if($pattern){
                    if(preg_match($pattern, $l2[2])){
                        $matchType = 'yes';
                    }else{

```

```

        $matchType = 'no';
    } // END: of if(preg_match($pattern, $l2[0]))
} else {
    $matchType = 'yes';
} // END: of if($pattern)
} // END: of if($matchType=='yes')
if($l2[0]!=''){
    if($matchType == 'yes'){
        // OUTPUT: types
        $types_element = $doc->createElement("list");
        $item->appendChild($types_element);
        $types_element->setAttribute("type", $bottomID_array[$cat]);
        $type_element = $doc->createElement("item");
        $types_element->appendChild($type_element);
        $type_element->setAttribute("n", $l2[0]);
        $type_element->setAttribute("xml:id", $l2[2]);
        $typename_element = $doc->createElement("ident");
        $typename_element->appendChild($doc->createTextNode($l2[0]));
        $type_element->appendChild($typename_element);
        $typename_element->setAttribute("type",
$bottomID_array[$cat]);
        // OUTPUT: Tablets
        $tablets_element = $doc->createElement("list");
        $type_element->appendChild($tablets_element);
        $tablets_element->setAttribute("type", "tablets");
        // FOR: each tablet
        foreach ($l2[1] as $l3){
            // OUTPUT: Tablet number
            $tablet_element = $doc->createElement("item");
            $tablets_element->appendChild($tablet_element);
            $tablet_element->setAttribute("n", $l3[0]);
            // OUTPUT: TabletNumber
            $tabletsnumber_element = $doc->createElement("ident");
            $tabletsnumber_element->appendChild($doc-
>createTextNode($l3[0]));
            $tablet_element->appendChild($tabletsnumber_element);
            $tabletsnumber_element->setAttribute("type", "tabletID");
            // OUTPUT: Number
            $number_element = $doc->createElement("num");
            $number_element->appendChild($doc->createTextNode($l3[1]));
            $tablet_element->appendChild($number_element);
            $number_element->setAttribute("type", "amount");
            // OUTPUT: Original spelling of the word in the text
            if($l3[2][0]!=''){
                for($h=0;$h<$l3[1];$h++){
                    $render_el = $doc->createElement("w");
                    $render_el->appendChild($doc-
>createTextNode($l3[2][$h]));
                    $tablet_element->appendChild($render_el);
                    $render_el->setAttribute("n", $h+1);
                } // END: for($h=0;$h<$l3[1];$h++){
            } // END: if($l3[2][0]!=''){
        } // END: of foreach ($l2[1] as $l3)
    } // END: of if($matchType == 'yes')
} else {
    $item->setAttribute("xml:id", $l1[2]);
    // OUTPUT: Tablets
    $tablets_element = $doc->createElement("list");
    $item->appendChild($tablets_element);
    $tablets_element->setAttribute("type", "tablets");
    foreach ($l2[1] as $l3){
        // OUTPUT: Tablet number
        $tablet_element = $doc->createElement("item");
        $tablets_element->appendChild($tablet_element);
        $tablet_element->setAttribute("n", $l3[0]);
        // OUTPUT: TabletNumber
        $tabletsnumber_element = $doc->createElement("ident");
        $tabletsnumber_element->appendChild($doc-
>createTextNode($l3[0]));
        $tablet_element->appendChild($tabletsnumber_element);
        $tabletsnumber_element->setAttribute("type", "tabletID");
        // OUTPUT: Number
        $number_element = $doc->createElement("num");

```



```

$number_element->appendChild($doc->createTextNode($l3[1]));
$tablet_element->appendChild($number_element);
$number_element->setAttribute("type", "amount");
// OUTPUT: Original spelling of the word in the text
if($l3[2][0]!=''){
  for($h=0;$h<$l3[1];$h++){
    $render_el = $doc->createElement("w");
    $render_el->appendChild($doc-
>createTextNode($l3[2][$h]));
    $tablet_element->appendChild($render_el);
    $render_el->setAttribute("n", $h+1);
  } // END: for($h=0;$h<$l3[1];$h++){
} // END: if($l3[2][0]!=''){
} // END: of foreach ($l2[1] as $l3)
} // END: of if($l2[0]!='')
} // END: of foreach ($l1[1] as $l2)
} // END: of if($match == "yes")
} // END: of foreach ($l as $l1)
} // END: of if($scans = scandir($dir, 0))
//Return status failed, if any error found.
if ($error_code == 1) {
  $ab7 = $doc->createElement("ab");
  $ab7->appendChild($doc->createTextNode('failed'));
  $encodingDesc->appendChild($ab7);
  $ab7->setAttribute("type", "status");
} // END: if ($error_code == 1) {
// Return status passed, if no error found
else if ($error_code == 0) {
  $ab7 = $doc->createElement("ab");
  $ab7->appendChild($doc->createTextNode('passed'));
  $encodingDesc->appendChild($ab7);
  $ab7->setAttribute("type", "status");
} // END: else if ($error_code == 0) {
return simplexml_load_string($doc->saveXML());
} // END: Function

// ---- GET ALL ---- //
/*
* Function which retrieves all the words in the tablets
* OPTIONAL PARAMETER
* $pattern: If a pattern is sent with the function only the words which fit the
pattern will be returned.
* Pattern wildcard is *
* Alternative characters can be presented like this: {abc}
*/
// FUNCTION: get_all()
function get_all($pattern = FALSE) {
  // SET: array
  $all = array(
    'word',
    'person',
    'placename',
    'military',
    'consul',
    'date'
  );
  // SET: category
  $cat = 'all';
  // SET: title
  $method_title = 'APPELLO method: Get ' . $cat;
  // SET: authority
  $authority1 = 'Made available by Centre for the Study of Ancient Documents,
University of Oxford (http://www.csad.ox.ac.uk/) and Henriette Roued-Cunliffe as
a part of her D.Phil at the University of Oxford (http://www.roued.com/e-doc/);
  // SET: todays date
  $today = date(d) . '-' . date(m) . '-' . date(Y);
  // SET: bibl array
  $bibl_array = array(
    'Tab.Vindol.I' => 'A.K.Bowman, J.D.Thomas, Vindolanda: the Latin writing-
tablets, Britannia Monograph 4. London, 1983 ',
    'Tab.Vindol.II' => 'A.K.Bowman, J.D.Thomas, The Vindolanda Writing-Tablets
(Tabulae Vindolandenses II). London, 1994',

```

```

'Tab.Vindol.III' => 'A.K.Bowman, J.D.Thomas, The Vindolanda Writing Tablets
(Tabulae Vindolandenses III). London: British Museum Publications, 2003 ');
$bibl_ref = array('Tab.Vindol.II', 'Tab.Vindol.III');
// SET: encoding
$encoding = 'This file was produced by APPELLO version 0.2';
// SET: list of categories
$list_of = $cat;
$source = "Vindolanda Writing Tablets";
// SET: Array of languages
$languages = array(
    'eng' => 'English',
    'lat' => 'Latin',
    'grc' => 'Greek',
);
$lang_list = array('eng', 'lat');
// SET: xPath for tablet number
$tabID_path = '//tei:rs[@n="1"]';
// SET: attribute for tablet ID
$tabID_attr = 'key';
// SET: top label
$topID_array = array(
    'word' => 'lemma',
    'person' => 'person',
    'placename' => 'placename',
    'military' => 'militaryOfficial',
    'consul' => 'consul',
    'date' => 'month',
);
// SET: bottom label
$bottomID_array = array(
    'word' => 'lemma_subtype',
    'date' => 'date',
);
// SET: xPath to tag
$tag_path = array(
    'word' => '//tei:w[@n]',
    'person' => '//tei:persName[@nymRef]',
    'placename' => '//tei:placeName[@key]',
    'military' => '//tei:rs[@type="militaryOfficial"]',
    'consul' => '//tei:rs[@type="consul"]',
    'date' => '//tei:rs[@type="julian_calendar"]',
);
// SET: local tablets directory
$dir="tablets";
// SET: Attributes for tags
$attr_top = array(
    'word' => 'lemma',
    'person' => 'nymRef',
    'placename' => 'nymRef',
    'military' => 'nymRef',
    'consul' => 'nymRef',
    'date' => 'nymRef',
);
$attr_bottom = array(
    'word' => 'subtype',
    'date' => 'rend',
);
$attr_nr = 'n';
$attr_key = array(
    'word' => 'lemma',
    'person' => 'key',
    'placename' => 'key',
    'military' => 'key',
    'consul' => 'key',
    'date' => 'key',
);
$attr_extra = array(
    'word' => 'lemma',
    'person' => 'person',
    'placename' => 'placename',
    'military' => 'militaryOfficial',
    'consul' => 'consul',
    'date' => 'month',

```

```

);
$extra_array = array(
    'person' => 'profession',
);
$attr_orig = array(
    'word' => 'rend',
    'person' => 'rendition',
    'placename' => 'rend',
);
// IF: there is a pattern - clean it
if($pattern){
    $search = array(
        '{',
        '}',
        '*',
    );
    $replace = array(
        '[',
        ']{1}',
        '([a-zA-Z]{1})',
    );
    $pattern = str_replace($search, $replace, $pattern);
    $pattern = '/' . $pattern . '/';
} // END: if($pattern){
// SET: error_code=0 for Success / error_code=1 for Failure
$error_code = 0;
// OUTPUT: the document
$doc = new DOMDocument();
$doc->formatOutput = true;
// OUTPUT: Response Element - building a TEI document
$tei = $doc->createElement("TEI");
$doc->appendChild($tei);
// OUTPUT: teiHeader
$teiheader = $doc->createElement("teiHeader");
$tei->appendChild($teiheader);
// OUTPUT: fileDesc
$fileDesc = $doc->createElement("fileDesc");
$teiheader->appendChild($fileDesc);
// OUTPUT: title
$titleStmt = $doc->createElement("titleStmt");
$fileDesc->appendChild($titleStmt);
$title = $doc->createElement("title");
$title->appendChild($doc->createTextNode($method_title));
$titleStmt->appendChild($title);
// OUTPUT: publicationStmt
$publicationStmt = $doc->createElement("publicationStmt");
$fileDesc->appendChild($publicationStmt);
$authority = $doc->createElement("authority");
$authority->appendChild($doc->createTextNode($authority1));
$publicationStmt->appendChild($authority);
$date = $doc->createElement("date");
$date->appendChild($doc->createTextNode($today));
$publicationStmt->appendChild($date);
// OUTPUT: sourceDesc
$sourceDesc = $doc->createElement("sourceDesc");
$fileDesc->appendChild($sourceDesc);
for($m=0;$m<count($bibl_ref);$m++){
    $bibl = $doc->createElement("bibl");
    $bibl->appendChild($doc->createTextNode($bibl_array[$bibl_ref[$m]]));
    $sourceDesc->appendChild($bibl);
    $bibl->setAttribute("n", $bibl_ref[$m]);
} // END: for($m=0;$m<count($bibl_ref);$m++){
// OUTPUT: encodingDesc
$encodingDesc = $doc->createElement("encodingDesc");
$teiheader->appendChild($encodingDesc);
$ab5 = $doc->createElement("ab");
$ab5->appendChild($doc->createTextNode($encoding));
$encodingDesc->appendChild($ab5);
// OUTPUT: profileDesc
$profileDesc = $doc->createElement("profileDesc");
$teiheader->appendChild($profileDesc);
$textClass = $doc->createElement("textClass");
$profileDesc->appendChild($textClass);

```

```

$catRef = $doc->createElement("catRef");
$textClass->appendChild($catRef);
$catRef->setAttribute("target", 'List of '.$list_of.' from '.$source);
// OUTPUT: langUsage
$langUsage = $doc->createElement("langUsage");
$profileDesc->appendChild($langUsage);
for($m=0;$m<count($lang_list);$m++){
    $language = $doc->createElement("language");
    $language->appendChild($doc->createTextNode($langs[$lang_list[$m]]));
    $langUsage->appendChild($language);
    $language->setAttribute("ident", $lang_list[$m]);
} // END: for($m=0;$m<count($lang_list);$m++){
// OUTPUT: Text
$text = $doc->createElement("text");
$tei->appendChild($text);
// OUTPUT: Body
$body = $doc->createElement("body");
$text->appendChild($body);
// OUTPUT: List
$list = $doc->createElement("list");
$body->appendChild($list);
// OUTPUT: Type of list
$list->setAttribute("type", $list_of);
// FOR: For each category in ALL
for($t=0;$t<count($all);$t++){
    // UNSET: array
    unset($l);
    // IF: Scan tablets directory
    if($scans = scandir($dir, 0)){
        // FOR: get data
        for($i=2;$i<count($scans);$i++){
            // SET: Connection to each tablet
            $tablet=$dir . '/' . $scans[$i];
            // IF: connected
            if($xml_tablet = simplexml_load_file($tablet)){
                // SET: Register tei namespace
                $xml_tablet->registerXPathNamespace('tei', 'http://www.tei-
c.org/ns/1.0');
                // SET: tablet nr
                $idents=$xml_tablet->xpath($tabID_path);
                foreach($idents[0]->attributes() as $a => $b) {
                    $$a=$b;
                    $tabletNum=$$tabID_attr;
                } // END: of foreach($idents[0]->attributes() as $a => $b)
                // SET: tags
                $tags=$xml_tablet->xpath($tag_path[$all[$t]]);
                // FOR: each tag
                foreach($tags as $tag){
                    // UNSET: attributes
                    unset($$attr_top[$all[$t]]);
                    unset($$attr_bottom[$all[$t]]);
                    unset($$attr_nr);
                    unset($$attr_key[$all[$t]]);
                    unset($$attr_extra[$all[$t]]);
                    unset($$attr_orig[$all[$t]]);
                    // SET: pattern matches to 'no' for default
                    $match = 'no';
                    $matchType = 'no';
                    // SET: each attribute in the tag
                    foreach($tag[0]->attributes() as $a => $b) {
                        // SET: the values
                        $$a=$b;
                    } // END: of foreach($tag[0]->attributes() as $a => $b)
                    $top1 = $$attr_top[$all[$t]];
                    $bottom1 = $$attr_bottom[$all[$t]];
                    $n1 = $$attr_nr;
                    $id1 = $$attr_key[$all[$t]];
                    $extral = $$attr_extra[$all[$t]];
                    $orig1 = $$attr_orig[$all[$t]];
                    $bottom = "" . $bottom1[0] . "";
                    $nr = "" . $n1[0] . "";
                    $id = "" . $id1[0] . "";
                    // IF: person

```

```

        if($all[$t]=='person'){
            $id = '_'.$id;
        };
        // IF: military
        if($all[$t]=='military'){
            $id = $id.'_' ;
        };
        // IF: top attribute
        if($top1[0]!=''){
            $top = "" . $top1[0] . " ";
        }else{
            $top = "" . $id1[0] . " ";
        } // END: if($top1[0]!=''){
        $tnr = "" . $tabletNum[0] . " ";
        // IF: extra attribute
        if($extral[0]!=''){
            $extra = "" . $extral[0] . " ";
        }else{
            $extra='';
        } // END: if($extral[0]!=''){
        // IF: original attribute
        if($orig1[0]!=''){
            $orig = "" . $orig1[0] . " ";
        }else{
            $orig='';
        } // END: if($orig1[0]!=''){
        $l[$top][0] = $top;
        // IF: date
        if($all[$t]=='date'){
            $l[$top][2] = strtolower($top);
        }else{
            $l[$top][2] = strtolower($id);
        };
        $l[$top][3] = strtolower($extra);
        $l[$top][1][$bottom][0] = $bottom;
        if($all[$t]=='date'){
            $l[$top][1][$bottom][2] = strtolower($id);
        }else{
            $l[$top][1][$bottom][2] = strtolower($bottom);
        };
        $l[$top][1][$bottom][1][$tnr][0] = $tnr;
        $l[$top][1][$bottom][1][$tnr][1] = $nr;
        $l[$top][1][$bottom][1][$tnr][2][] = $orig;
        $l[$top][5] = $cat;
    } // END: foreach($tags as $tag)
} // END: if($xml_tablet = simplexml_load_file($tablet))
} // END: for($i=2;$i<count($scans);$i++){
// FOR: each part of array
foreach ($l as $l1){
    // SET: cleaning for xml:id
    $search_clean = array(
        '[',
        ']',
        '?',
        ',',
        '-',
        '(',
        ')',
    );
    $replace_clean = array(
        '',
        '',
        '',
        '',
        '-',
        '',
        '',
        '',
    );
};
$l1[2] = str_replace($search_clean, $replace_clean, $l1[2]);
// IF: the pattern fits and pattern is turned on.
if($pattern){
    // IF: pattern matches top - cleaned id
    if(preg_match($pattern, $l1[2])){

```

```

        $match = 'yes';
        $matchType1 = 'yes';
    }else{
        $match = 'no';
        // IF: pattern is in bottom
        foreach ($l1[1] as $l2){
            if(preg_match($pattern, $l2[2])){
                $match = 'yes';
                $matchType1 = 'no';
            } // END: of if(preg_match($pattern, $l2[0]))
        } // END: of foreach ($l1[1] as $l2)
    } // END: of if(preg_match($pattern, $l1[0]))
}
else{
    $match = 'yes';
} // END: of if($pattern)
// IF: match
if($match == "yes"){
    $item = $doc->createElement("item");
    $list->appendChild($item);
    $item->setAttribute("rend", $l1[0]);
    $item->setAttribute("n", $l1[2]);
    $ident1 = $doc->createElement("ident");
    $ident1->appendChild($doc->createTextNode($l1[0]));
    $item->appendChild($ident1);
    $ident1->setAttribute("type", $topID_array[$all[$t]]);
    // IF: extra Values
    if($l1[3] AND $extra_array[$all[$t]]){
        $ident2 = $doc->createElement("ident");
        $ident2->appendChild($doc->createTextNode($l1[3]));
        $item->appendChild($ident2);
        $ident2->setAttribute("type", $extra_array[$all[$t]]);
    }
    // FOR: each part of the array
    foreach ($l1[1] as $l2){
        // IF: type matches
        if($matchType1=='yes'){
            $matchType='yes';
        }else{
            if($pattern){
                if(preg_match($pattern, $l2[2])){
                    $matchType = 'yes';
                }else{
                    $matchType = 'no';
                } // END: of if(preg_match($pattern, $l2[0]))
            }else{
                $matchType = 'yes';
            } // END: of if($pattern)
        } // END: of if($matchType1=='yes')
        $l2[2] = str_replace($search_clean, $replace_clean, $l2[2]);
        if($l2[0]!=''){
            // IF: type matches
            if($matchType == 'yes'){
                $types_element = $doc->createElement("list");
                $item->appendChild($types_element);
                $types_element->setAttribute("type",
$bottomID_array[$all[$t]]);
                $type_element = $doc->createElement("item");
                $types_element->appendChild($type_element);
                $type_element->setAttribute("rend", $l2[0]);
                $type_element->setAttribute("n", $l2[2]);
                $typename_element = $doc->createElement("ident");
                $typename_element->appendChild($doc-
>createTextNode($l2[0]));
                $type_element->appendChild($typename_element);
                $typename_element->setAttribute("type",
$bottomID_array[$all[$t]]);
                // OUTPUT: Tablets
                $tablets_element = $doc->createElement("list");
                $type_element->appendChild($tablets_element);
                $tablets_element->setAttribute("type", "tablets");

                foreach ($l2[1] as $l3){
                    // OUTPUT: Tablet number

```

```

        $tablet_element = $doc->createElement("item");
        $tablets_element->appendChild($tablet_element);
        $tablet_element->setAttribute("n", $l3[0]);

        // OUTPUT: TabletNumber
        $tabletsnumber_element = $doc->createElement("ident");
        $tabletsnumber_element->appendChild($doc-
>createTextNode($l3[0]));
        $tablet_element->appendChild($tabletsnumber_element);
        $tabletsnumber_element->setAttribute("type", "tabletID");
        // OUTPUT: Number
        $number_element = $doc->createElement("num");
        $number_element->appendChild($doc-
>createTextNode($l3[1]));
        $tablet_element->appendChild($number_element);
        $number_element->setAttribute("type", "amount");
        // OUTPUT: Original spelling of the word in the text
        if($l3[2][0]!=''){
            for($h=0;$h<$l3[1];$h++){
                $render_el = $doc->createElement("w");
                $render_el->appendChild($doc-
>createTextNode($l3[2][$h]));
                $tablet_element->appendChild($render_el);
                $render_el->setAttribute("n", $h+1);
            } // END: for($h=0;$h<$l3[1];$h++){
        } // END: if($l3[2][0]!=''){

    } // END: foreach ($l2[1] as $l3)
} // END: if($matchType == 'yes')
}else{
    // OUTPUT: Tablets
    $tablets_element = $doc->createElement("list");
    $item->appendChild($tablets_element);
    $tablets_element->setAttribute("type", "tablets");
    foreach ($l2[1] as $l3){
        // OUTPUT: Tablet number
        $tablet_element = $doc->createElement("item");
        $tablets_element->appendChild($tablet_element);
        $tablet_element->setAttribute("n", $l3[0]);

        // OUTPUT: TabletNumber
        $tabletsnumber_element = $doc->createElement("ident");
        $tabletsnumber_element->appendChild($doc-
>createTextNode($l3[0]));
        $tablet_element->appendChild($tabletsnumber_element);
        $tabletsnumber_element->setAttribute("type", "tabletID");
        // OUTPUT: Number
        $number_element = $doc->createElement("num");
        $number_element->appendChild($doc->createTextNode($l3[1]));
        $tablet_element->appendChild($number_element);
        $number_element->setAttribute("type", "amount");
        // OUTPUT: Original spelling of the word in the text
        if($l3[2][0]!=''){
            for($h=0;$h<$l3[1];$h++){
                $render_el = $doc->createElement("w");
                $render_el->appendChild($doc-
>createTextNode($l3[2][$h]));
                $tablet_element->appendChild($render_el);
                $render_el->setAttribute("n", $h+1);
            } // END: for($h=0;$h<$l3[1];$h++){
        } // END: if($l3[2][0]!=''){
    } // END: foreach ($l2[1] as $l3)
} // END: if($l2[0]!='')
} // END: foreach ($l1[1] as $l2)
} // END: if($match == "yes")
} // END: foreach ($l as $l1)
} // END: if($scans = scandir($dir, 0))
} // END: for($t=0;$t<count($all);$t++)
// SET: Return status failed, if any error found.
if ($error_code == 1) {
    $ab7 = $doc->createElement("ab");
    $ab7->appendChild($doc->createTextNode('failed'));
    $encodingDesc->appendChild($ab7);
}

```

```

        $ab7->setAttribute("type", "status");
    } // END: if ($error_code == 1) {
    // SET: Return status passed, if no error found
    else if ($error_code == 0) {
        $ab7 = $doc->createElement("ab");
        $ab7->appendChild($doc->createTextNode('passed'));
        $encodingDesc->appendChild($ab7);
        $ab7->setAttribute("type", "status");
    } // END: else if ($error_code == 0) {
    return simplexml_load_string($doc->saveXML());
} // END: Function

// GET NOTE //
// FUNCTION: get_note()
function get_note($tabletID, $notenr) {
    // SET: error_code=0 for Success / error_code=1 for Failure
    $error_code = 0;
    // SET: new DOMDocument
    $doc = new DOMDocument();
    // SET: Connection
    $dir="tablets";
    $tablet=$dir . '/' . $tabletID . '.xml';
    // SET: note
    $xml_note = simplexml_load_file($tablet);
    $xml_note->registerXPathNamespace('tei', 'http://www.tei-c.org/ns/1.0');
    // SET: Retrieve information from the XML
    $notes=$xml_note->xpath('//tei:note[@n="'. $notenr.'" and @type="line"]');
    // FOR: each note
    foreach($notes as $note){
        $xmlnote=$note->asXML();
    }
    // OUTPUT: note
    $doc->loadXML($xmlnote);
    return simplexml_load_string($doc->saveXML());
}

// SET: the server and the functions
$server = new Zend_Rest_Server();
$server->addFunction('get_tablets');
$server->addFunction('get_tablet');
$server->addFunction('get_word');
$server->addFunction('get_rs');
$server->addFunction('get_person');
$server->addFunction('get_geog');
$server->addFunction('get_all');
$server->addFunction('get_note');
// OUTPUT: handle the functions
$server->handle();
?>

```


Function name	Function case	Description	File	Diagram
doSomething()		runs StateChangedDo() with response from xml.php	DSS_scripts.js	1, 2, 3
	add=eng	UPDATE 'used = 1' in DSS_engines	xml.php	1
	del=eng	UPDATE 'used = 0' in DSS_engines	xml.php	1
	add=letter	INSERT a record in DSS_letters	xml.php	2
	del=letter	DELETE a record in DSS_letters	xml.php	2
	add=resolve	UPDATE 'resolve = 1' in DSS_letters	xml.php	2
	del=resolve	UPDATE 'resolve = 0' in DSS_letters	xml.php	2
	add=arg	INSERT a record into DSS_arguments	xml.php	3
	del=arg	DELETE a record from DSS_linkToURL	xml.php	3
	add=against	UPDATE 'boolean = 0' in DSS_arguments	xml.php	3
	add=for	UPDATE 'boolean = 1' in DSS_arguments	xml.php	3
	add=url	INSERT a record in DSS_linkToURL	xml.php	3
	del=url	DELETE a record from DSS_linkToURL	xml.php	3
	add=bibl	INSERT a record in DSS_linkToBibl	xml.php	3
	del=bibl	DELETE a record from DSS_linkToBibl	xml.php	3
	add=doclink	INSERT a record in DSS_linkToDoc	xml.php	3
	del=doclink	DELETE a record from DSS_linkToDoc	xml.php	3
	add=linkword	UPDATE 'linkType = word' in DSS_linkToDoc	xml.php	3
	add=linkletter	UPDATE 'linkType = letter' in DSS_linkToDoc	xml.php	3
stateChangedDo()		sends response text on to 'doSomethingText'	DSS_scripts.js	1, 2, 3

Function name	Function case	Description	File	Diagram
addText()		sends text onto xml.php	DSS_scripts.js	2, 3
	add=lettertext	UPDATE 'letter' in DSS_letters	xml.php	2
	add=argtext	UPDATE field 'text' in DSS_arguments	xml.php	3
	add=urtext	UPDATE field 'URL' in DSS_linkToURL	xml.php	3
	add=urtitle	UPDATE field 'URLtitle' in DSS_linkToURL	xml.php	3
	add=bibltext	UPDATE field 'biblURL' in DSS_linkToBibl	xml.php	3
	add=bibltile	UPDATE field 'biblRef' in DSS_linkToBibl	xml.php	3
view()		delay for view1()	DSS_scripts.js	1
view1()		runs stateChangedView() with view.php response	DSS_scripts.js	1
stateChangedView()		sends response text on to 'main-view'	DSS_scripts.js	1
showEng()		delay for showEng1()	DSS_scripts.js	1
showEng1()		runs stateChangedEng() with 'get=eng' response	DSS_scripts.js	1
	get=eng	SELECT records from DSS_engines	xml.php	1
stateChangedEng()		sends response text on to 'engbox'	DSS_scripts.js	1
showTop()		delay for showTop1()	DSS_scripts.js	2, 4
showTop1()		creates 'topbox' and 'screenverlay' and runs stateChangedTop() with response	DSS_scripts.js	2, 4
	get=word	gets and suggests words	xml.php	4
	get=char	SELECT records from DSS_letters	xml.php	2
stateChangedTop()		sends response text on to 'box'	DSS_scripts.js	2, 4
closeTop()		closes topbox and screenoverlay	DSS_scripts.js	2, 4

Function name	Function case	Description	File	Diagram
showArg()		delay for showTArg1()	DSS_scripts.js	3
showArg1()		creates 'argtopbox' and 'screenverlay' and runs stateChangedTop() with response	DSS_scripts.js	3
	get=arg	SELECT records from DSS_arguments	xml.php	3
stateChangedArg()		sends response text on to 'argbox'	DSS_scripts.js	3
closeArgTop()		closes argtopbox	DSS_scripts.js	3
showSug()		delay for showSug1()	DSS_scripts.js	4
showSug1()		creates sugtopbox and screenoverlay and runs stateChangedSug() with get=sug	DSS_scripts.js	4
	sug=word	getting suggested words from concordances	xml.php	4
	sug=lexicon	getting suggested words from lexicons	xml.php	4
stateChangedSug()		sends response text to 'sugbox'	DSS_scripts.js	4
closeSugTop()		closes sugtopbox	DSS_scripts.js	4
GetXmlHttpRequest()		creates an XML http object	DSS_scripts.js	
disableEnterKey()		so that pressing 'enter' doesn't reload the page	DSS_scripts.js	
enter_pressed()		so that pressing 'enter' doesn't reload the page	DSS_scripts.js	

Title	Description	Folder	Creator	Language	Format
index.php	Main HTML page containing <head> and <body> tags along with overlay tags for the AJAX scripts	DSS prototype	HR	English	PHP
config.php	Configuration file for set variables	DSS prototype	HR	English	PHP
DSS_style.css	CSS stylesheet	DSS prototype	HR	English	CSS
DSS_scripts.js	Javascript file containing all Javascript AJAX functions	DSS prototype	HR	English	JS
legend.jpg	Image showing the letter resolve legend	DSS prototype	HR	English	JPG
ajax_loader.gif	Image used when page is loading	DSS prototype	HR	English	GIF
670f_a.jpg	Vindolanda Tablet 670 side A	DSS prototype	HR	English	JPG
view.php	Displays main view with db-tei.php and view.xsl stylesheet	DSS prototype	HR	English	PHP
view.xsl	Used to format the db-tei.php Web Service into the main view	DSS prototype	HR	English	XSL
db-tei.php	Database to TEI Web Service - outputs DSS structure from database as TEI	DSS prototype	HR	English	PHP
xml.php	Deals with all requests from functions	DSS prototype	HR	English	PHP
viewlink.xsl	XSLT stylesheet for the js function showArg()	DSS prototype	HR	English	XSL
get_words.xsl	XSLT stylesheet for the js function showArg()	DSS prototype	HR	English	XSL
get_sugword.xsl	XSLT stylesheet for the js function showTop(get=word)	DSS prototype	HR	English	XSL

Title	Description	Folder	Creator	Language	Format
sug_word.xsl	XSLT stylesheet for the js function showSug	DSS prototype	HR	English	XSL
sug_lexicon.xsl	XSLT stylesheet for the js function showSug	DSS prototype	HR	English	XSL
DSS_670.sql	Database for DSS prototype using tablet 670	DSS prototype	HR	English	SQL
index.php	Web Service file for APPELLO	APPELLO	HR	English	PHP
Example1.swf	ADDING MEANING	Examples	HR	None	SWF
Example2.swf	ADDING ARGUMENTS	Examples	HR	None	SWF
Example3.swf	SEARCHING FOR CONCORDANCES	Examples	HR	None	SWF
Example4.swf	SEARCHING LEXICONS	Examples	HR	None	SWF
Example5.swf	VINDOLANDA TABLETS ONLINE II WEBSITE	Examples	HR	None	SWF