CS 501- Software Engineering

Legal Data Markup Software
Software Requirements Specification

Version 1.0

Document Revision History

Date	Version	Description	Author
09/29/00	1.0	First draft	LDMS Team
10/3/00	1.0	Final draft incorporating client-requested modifications	LDMS Team

Table of Contents

Software Requirements Specification

1. Introduction

The intent of this project is to create a software tool that will convert the US Code of law from its distribution ASCII format into well-formed, valid XML. The XML output would subsequently be utilized by our client, the Legal Information Institute, in next-generation applications that will make the U.S. Code available in a variety of different formats to the general public. Examples of such use include the electronic publication of the code on the Internet and downloadable versions in Folio Views format.

1.1 Purpose

This purpose of this document is to define all the requirements of the LDMS needed to:

- Acquire and convert the US code into XML.
- Assist the client in maintaining the XML conversion utility.
- Assist the client in customizing the program code to maintain flexibility of the product.

1.2 Scope

This document applies only to the LDMS XML conversion utility and any associated maintenance tools we provide. These requirements are by no means a comprehensive specification of future functionality that may be added by the developers or by the client. Only those requirements that are apparent at the time of this writing are included in this particular draft of the specifications document; future revisions may include extended or additional requirements that are added as the design of the final product's functionality becomes clearer.

1.3 Definitions, Acronyms and Abbreviations

DDD	DTD Design Document
DTD	Document Type Definition
HTML	Hypertext Markup Language
LDMS	Legal Data Markup Software
Leda	Name of server running development and application environment.
LII	Legal Information Institute
MTBF	Mean Time Between Failures
MTTR	Mean Time to Repair
PDD	Program Design Document
SRS	Software Requirements Specification
TOC	Table of Contents
US	United States
W3C	World Wide Web Consortium
XML	Extensible Markup Language
XSL	Extensible Stylesheet Language

1.4 References

The following documents are related to the project and have been consulted:

- Current version of source code for conversion from raw ASCII to HTML.
- http://uscode.house.gov/download.htm – U.S. Code related input formats.
- http://www.lexum.umontreal.ca/fr/equipes/technologie/dtd/LOIQ.dtd – Montreal's DTDs used for legislative purposes.
- http://elj.warwick.ac.uk/jilt/00-2/bruce.html – General background on legislative and legal publishing.
- http://elj.warwick.ac.uk/jilt/00-1/arnold.html – Provides history, motivation, and high-level implementation of EnAct, a project done in Tasmania similar to this one.
- http://nwalsh.com/docs/articles/xml/ – An XML tutorial.
- http://developer.irt.org/script/xml.htm – Archive of frequently asked XML questions.
- http://www.w3.org/TR/REC-xml – The W3C XML draft specification.
- *The Perl CD Bookshelf*, 6 bestselling books on CD-ROM, O'Reilly & Associates, Inc. August 1999
- *The House of Quality*, Hauser, J.R. and D. Clausing, *Harvard Business Review*, May-June 1988, pp. 63-73.
- *Writing Quality Requirements*, Wiegers, K. E., *Software Development*, May 1999

1.5 Overview

The rest of this document discusses the rationale and importance of this project to the client and provides a comprehensive specification of all requirements for the project. These specifications include functional requirements, as specified by the client; usability requirements, which identify the type of interfaces available to users and the accessible functionality of such interfaces; minimum performance requirements, which specify minimum required response times for transactions, throughput, capacity and resource utilization of the LDMS; design constraints mandated by the client; and supportability requirements, which indicate the steps required to make the system flexible enough to adapt to changes which may occur during the product's usable lifespan. In short, the specifications outlined by this document will provide a sufficient description of functionality that will be used to determine the product's fulfillment of the client's expectations.

1.6 Roles and Responsibilities

Name	Department	Responsibility
Thomas Bruce	Legal Information Institute	Project Sponsor
William Arms	Computer Science Department	Project Sponsor
Amy Siu	Computer Science Department	Project Reviewer
Ju Joh	Computer Science Department	Student Developer
Sylvia Kwakye	Computer Science Department	Student Developer
Jason Lee	Computer Science Department	Student Developer
Nidhi Loyalka	Computer Science Department	Student Developer
Omar Mehmood	Computer Science Department	Student Developer
Charles Shagong	Computer Science Department	Student Developer
Brian Williams	Computer Science Department	Student Developer

2. Overall Description

The LII was launched in 1992 by Peter W. Martin and Thomas R. Bruce, current directors of the institute, as a non-profit service of the Cornell Law School. The founders hoped to apply the revolutionary technology of digital information to the distribution of legal information, the practice of law and the delivery of legal education. In pursuit of these goals, Martin and Bruce created the first windows based web browser (Cello) and the first law site on the internet and established the standards and format for delivery of legal content on the internet. In addition they partnered with major law publishers, legal institutions and legal information users to create the most successful and most linked-to law resource on the web for legal professionals and the general public in the US and abroad.

The LII has sponsored several iterations of software to organize and present the US code of law, among other legal proceedings, into searchable and hyper-linked formats. LII utilizes software that converts the US code into HTML. Distinctive features of format and functionality offered by this HTML format draw over seven million visitors to their web site every week. Functional features of the HTML version of the code include keyword searches within titles or of all titles; next/previous click-through connectivity within sections of titles to facilitate browsing; the use of highlighted reserved words with legal meanings such as "repealed"; and notes on the history of the law. Refer to figure 1 for illustrations of the LII software in use.

In spite of the success of the HTML conversion utility however, several problems exist that make the technology unsustainable. HTML is primarily a presentation tool and is fundamentally designed to work for documents with fairly predictable structures. The raw US code document distributed by the House of Representatives varies in structure from title to title and even within titles. The code also has seemingly simple structures like tables and completely unstructured sections like footnotes and appendices that increase the complexity of conversion. As such, the HTML utility does not handle all content of the code equally well. It is also more difficult to export HTML documents to other formats because the markup tags in html do not describe the content of the documents.

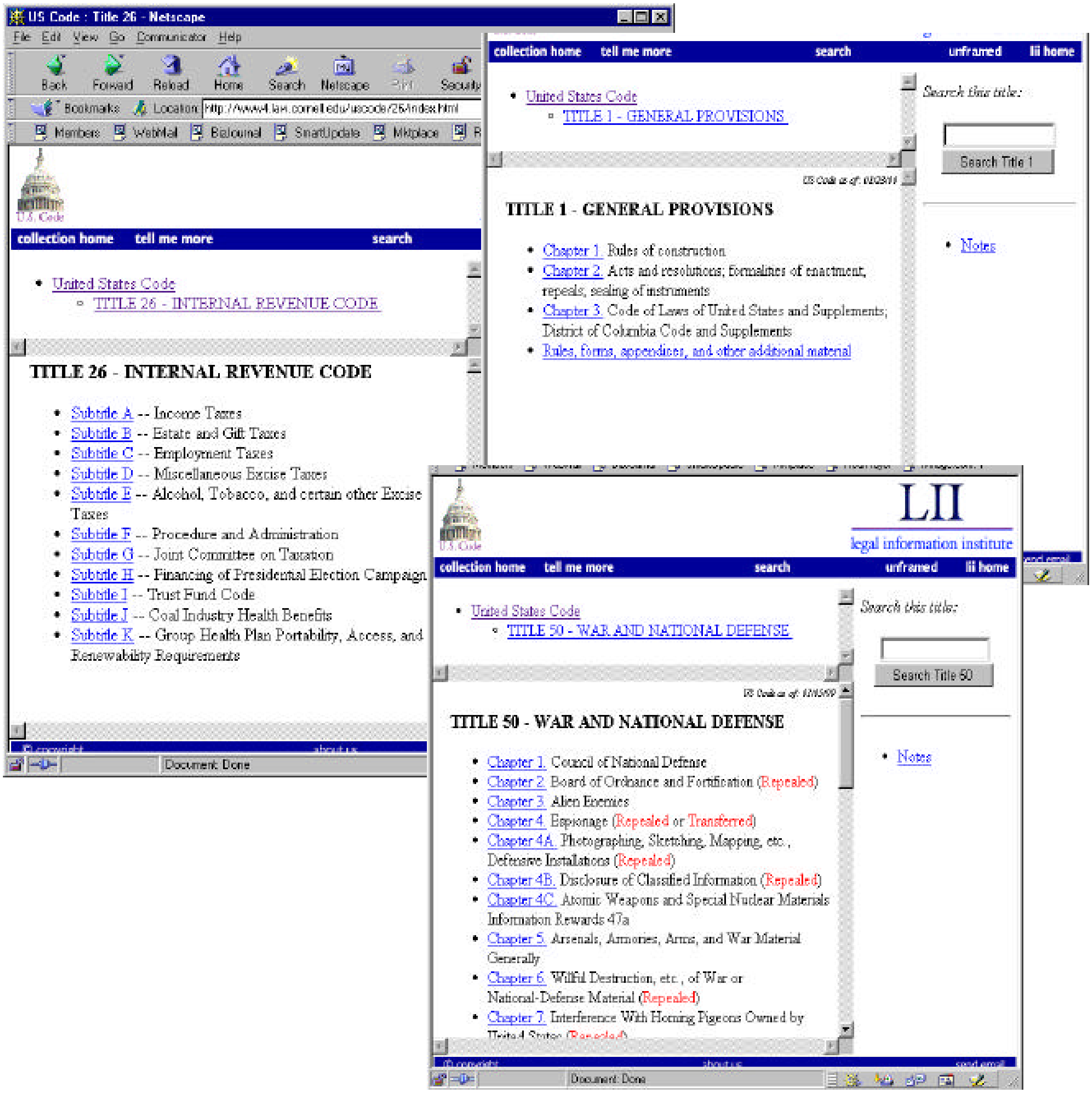


Figure 1: Various titles from the US Code displayed in Netscape Navigator after conversion to HTML. Note the differences between the titles displayed. Title 26 has subtitles, which the other two lack. Title 1 is very short while Title 50 is very large with many more chapters, some of which have the same number.

The LDMS project was conceived to essentially separate the content of the US code from its presentation while maintaining all the functionality of the HTML version. This will be accomplished with a DTD that logically binds the content of the US code to descriptive tags. The XML output based on this DTD will have information about the logical role of every element in the US code. Subsequently, presentation markup can be attached to the tags rather than actual parts of the text for export into any desired format. With well-formed, valid XML output, the LDMS can:

- Produce documents that maintain the familiar structural layout of the US code.
- Generate a cascading table of contents comprised of tagged elements such as chapter, subsection, etc.
- Highlight reserved words tagged in the XML.
- Conduct a title scope or full content scope search.
- Tag and preserve notes in the US code.
- Easily handle text tagged as tables.
- Link cross-references within the text of the US code.
- Preserve catchlines of the various subdivisions.
- Generate appendices for each title.

In essence, the LDMS will be a sophisticated pattern recognition algorithm that recognizes and tags all the major structural elements of the US code to output “smart” content that enable the functions above to be implemented.

3. Specific Requirements

3.1 Functionality

The functional requirements of the LDMS directly follow from qualities specified by the client. Accuracy of the structural layout to the original data format is imperative. To this end, most of the requirement items have been designated to strive towards a valid XML output, based on a Document Type Declaration (DTD) to be determined, while minimizing elements that cannot be characterized. Requirement items auxiliary to the primary function of the LDMS are designated to aid in the operation of the LDMS. A comprehensive listing of the functional requirement items follows.

3.1.1 Table of Contents Generation

The LDMS shall generate a table of contents of the US Code. This TOC shall directly represent the hierarchy inherent to the structural organization of the US Code itself. The TOC is intended to give a brief overview of the various titles and each of their subdivisions as organized into chapters, sections, and subsections.

3.1.2 Appendices Generation

Each title of the US Code can contain an appendix that may contain additional materials or rules pertaining to the text body. It is deemed important that such appendices be represented accurately. The LDMS shall recognize appendix sections and markup their constituent elements as appropriate.

3.1.3 Catchline Handling

Catchlines are short headers in the US Code that specify the current position in the organizational hierarchy by including a short description of the ensuing text. The LDMS shall recognize catchlines in the US Code and appropriately mark them as such.

3.1.4 Preservation of Cross-references

The US Code includes many self-referential links. They are often used to provide references to or establish context of a section of text. Preservation of such cross-references is vital in providing ease of use to readers. The LDMS shall recognize cross-references by establishing anchors and links between sections of text.

3.1.5 Table Handling

Some data of the US Code is represented in a tabular format. In the current ASCII format, columns are frequently delimited by an arbitrary number of white spaces. Moreover, the arbitrary white spacing is inconsistent across titles because they have been established by constraints of physical layout. The LDMS shall recognize tabular data. The LDMS shall handle them by marking up data elements and organizing the elements into the proper dimensions and indices.

3.1.6 Preservations of Notes

Body text of the US Code at any hierarchical level may be annotated with notes that contain additional information. Such information is critical in providing references, background information, and sources. Although they lie outside the direct text body of the Code, they are deemed highly significant. The LDMS shall recognize notes that annotate bodies of the US Code regardless of their hierarchical level. Notes themselves shall be organized and preserved according to their original format.

3.1.7 Reserved Words Recognition

Certain subdivisions of a title are qualified by a reserved word, such as *repealed*, *omitted*, and *transferred*. These reserved words provide critical attributes to entire subdivisions of text that must be preserved. The LDMS shall recognize reserved words and markup the body of text to which they apply, if applicable.

3.1.8 Graceful Failures

The titles of the US Code display a staggering number of variations in organizational layout and data representation. Moreover, these variations may increase with subsequent revisions. There lies the possibility that certain elements of the US Code may not be marked up properly without breaking the validity of the output. The LDMS shall markup such data that cannot be characterized. It is with the hope of either improving the LDMS engine or modifying the DTD that such graceful failures be reduced or eliminated.

3.1.9 Special Character Handling

The US Code includes not only conventional text, but also some non-standard characters. These non-standard characters can have different meanings such as shortened representations of full words or organizational markers within the text. The LDMS shall recognize and markup such non-conventional characters and represent them accordingly.

3.1.10 Navigational Aids

The hierarchical organization of the US Code lends itself well to the implementation of navigational aids. Aiding the user in traversing adjacent subdivisions within a title or adjacent titles within the Code is deemed a preferred feature. The LDMS shall provide reference links to the previous and subsequent divisions of the US Code as appropriate to the current division within the hierarchy.

3.1.11 Known Data Input Path

The LDMS shall assume the raw ASCII form of the US Code can be found at a known location prior to processing.

3.2 Usability

The development and application environment is Red Hat Linux running on Leda. Typically, the LDMS will be executed by the cron daemon at periodic intervals defined by the client. However, two levels of users, normal and power, are defined for the human operation of the LDMS.

Normal users are assumed to be computer literate and familiar with the linux operating system. They will be required to start and/or stop the program from the linux command line window. An application manual describing the utility of the LDMS and the commands required to run it shall be provided for training of the normal user.

In addition to familiarity with the linux operation systems, power users must be familiar with the Perl programming language, XML, DTDs and the US code. A standard development directory containing the LDMS source code, source code documentation, help files and a manual page shall be provided for power users. With these tools, the power user's higher level of programming sophistication will enable them to customize the LDMS code, maintain it and update it should the input data change.

3.2.1 Required Training Time for Normal Users

A normal user will need 30 – 60 minutes to review the LDMS application documentation and a manual page to learn the commands and corresponding parameters required to run the LDMS.

3.2.2 Required Training Time for Power Users

The power user will need a week to review all documents including application and source code documentation.

3.2.3 Estimation of Time Required for Measurable Tasks

Given the specifications of Leda (refer to supporting information section), our estimates are as follows for the conversion of all the fifty titles of the US code to XML.

- 30 minutes to read in the entirety of US Code.
- 12-24 hours for conversion processing.

3.2.4 Status Messages

During execution of the LDMS, the background task manager will display status messages at intervals specified by the client. If the background task processing is progressing normally, the name of the title and number of lines processed will be displayed. An error message will be displayed if the program stalls. A message will also be displayed to let the user know when a task is complete.

3.3 Reliability

3.3.1 Availability

The product will ideally be available for use 100% of the time—or rather, whenever it is needed to mark up a newly received set of documents from the House of Representatives. Maintenance operations on the information generated should not be a significant factor in its operation, as the product should not require an exclusive lock on the information repository (XML representations of the US Code) at all. Furthermore, as significant extensions to this product are slated for implementation as processes operating on the generated files, installation of new functionality should not require making the product unavailable to the system.

3.3.2 Mean Time between Failures

The product will be designed to fail gracefully, marking unrecognized text appropriately and continuing. Exceptional errors, such as segmentation faults, should not occur within the useful lifetime of 3 years.

3.3.3 Mean Time to Repair

If the product does indeed fail, the mean time to repair will vary depending on the nature of the fault. Should the fault be a transient error in the underlying platform, all that should be required will be the time taken for the job to be restarted. A fatal error in the underlying platform may require as much time as it takes to restart the system. A semantic error within the program will most likely require repair by reprogramming the offending part of the product, which is time-dependent on the technician repairing the product.

3.3.4 Accuracy

Accuracy of the generated XML output is paramount to the success of this project. It is assumed that the LDMS will generate XML that faithfully and deterministically reproduces the structure of the original ASCII representation within defined tolerances. Validation and integrity testing can be performed using an XSL stylesheet to view the generated XML. The various components and tolerance levels of accuracy has been determined and is as follows.

3.3.4.1 Structural Layout

The LDMS shall process the US Code in raw ASCII format and generate a well-formed, valid XML output that faithfully represents the structural layout with 95% accuracy. This includes proper recognition of all data elements, proper markup of all data elements, and the minimization of graceful failures.

3.3.4.2 Table of Contents Generation

Generating the table of contents, which represents the hierarchy of the US Code organization, is critical in determining how elements within the hierarchical levels are recognized and marked up. This significant precursor step that precedes the primary markup processing requires an accuracy of 95%.

3.3.4.3 Reserved Words Recognition

Reserved words can exert a vital influence over entire divisions of text that is significant to reference users of the US Code. In addition, they do not exhibit the sophisticated structure that other elements exhibit. It is therefore determined that the LDMS shall recognize reserved words and their annotated region with 95% accuracy.

3.3.4.4 Preservation of Cross-references

Cross-references are significant in providing ease of reference to the US Code. Their preservation is not, however, critically important because failure to preserve them does not change the structural layout of the US Code nor does it affect the manner in which other data elements are recognized. The LDMS shall preserve cross-references with 75% accuracy.

3.3.4.5 Appendices Generation

Appendices provide excellent additional materials that may be of use to reference users of the US Code. Failure to adequately recognize and properly generate appendices does not, however, change the structural layout of the US Code nor does it affect the manner in which other data elements are recognized. The LDMS shall generate appendices with 75% accuracy.

3.3.4.6 Catchline Handling

Catchlines have the quality of having a relatively short and unsophisticated structure. Failure to properly handle catchlines within the US Code could lead to changes in how the structural layout of the US Code is recognized. In that event, however, they are unlikely to cause severe undesirable effects. The LDMS shall recognize and handle catchlines with 95% accuracy.

3.3.4.7 Preservation of Notes

Notes annotate specific sections of the US Code to provide references, background information, and sources. Failure to adequately recognize and preserve notes does not, however, change the structural layout of the US Code nor does it affect the manner in which other data elements are recognized. The LDMS shall preserves notes with 75% accuracy.

3.3.4.8 Table Handling

The US Code presents some data in tabular form. Such data has the quality of being relatively easy to recognize since they have a consistent organizational layout with a consistent delimiter. Failure to handle tables properly should not significantly alter the structural layout of the US Code nor affect the manner in which other data elements are recognized. The LDMS shall recognize and handle tables with 75% accuracy.

3.3.4.9 Special Character Handling

Special characters in the body of the US Code can have an ambiguous definition as they may represent full words or carry structural information about the text. They, however, have the quality of easy recognition since they are non-standard characters. The LDMS shall recognize and handle special characters with 75% accuracy.

3.3.5 Acceptable Bugs

Although the development team will attempt to render the final product as bug-free as possible, a perfect program is often characterized as an impossible goal. Therefore, should there be bugs in the final product, any defects that do not directly affect the usability of the program or the accuracy of the output will be deemed tolerable. Examples of tolerable bugs include long-term memory leaks and spurious error messages.

3.4 Performance

3.4.1 Response Time for a Transaction

The average processing time per title will be 30 minutes +/- 10 minutes.

3.4.2 Degradation Modes

Refer to section 3.1.8 - Graceful Failures.

3.4.3 Capacity

The LDMS running on Leda can accommodate one transaction or one user at a time.

3.4.4 Resource Utlilization

The LDMS will require about 12MB of free memory with the breakdown as follows:

- 2 MB for the interpreted Perl code.
- 5 MB input data buffer.
- 5 MB for output data.

3.5 Supportability

3.5.1 Output File Naming Convention

For ease of reference between the input ASCII files and the output XML files, the LDMS shall output all files with the same filename as the original but with the file extension changed to ".xml".

3.5.2 Source Level Documentation

It is deemed vitally important for development and maintenance that all source code is thoroughly documented, especially when code segments are written with specific functional requirements in mind. No code shall be approved for inclusion in builds without accompanying source level documentation and peer review of such documentation.

3.5.3 Manual Page Reference

The manual page reference shall be accessible to the user outside LDMS operations. It shall outline all execution parameters and aid the user in properly operating all the LDMS features.

3.5.4 Program Design Document

Development of the LDMS shall be documented by a program design document (PDD) outlining the implementation. It shall be the central reference for developers responsible for understanding, maintaining, and extending the LDMS. The PDD shall contain a high level view of the LDMS processing engine, detail individual processing components, and display all interfaces, within and external to, the system. To aid in supporting the LDMS, no development diverging from the requirements shall occur without peer approval, without modifying requirements, nor without modifying the PDD.

3.5.5 DTD Design Document

Development of an appropriate DTD shall be documented by a DTD design document(DDD). It shall be the sole reference for developers responsible for understanding, maintaining, and extending the DTD. The DDD shall contain a list of all elements and element attributes with details of their use. To aid in supporting the LDMS, no modifications to the DTD shall occur without peer approval, or without modifying the DDD.

3.6 Design Constraints

3.6.1 Operating System

The LDMS shall require a Unix environment to operate.

3.6.2 Development Language

The LDMS shall be developed using Perl. Perl has been chosen because of its use of regular expressions in pattern matching. This feature is an essential component of developing components to markup various kinds of elements in the US Code that may be recognized by specific text patterns or conventions.

3.6.3 File Input Format

The LDMS shall require input data to be the US Code in ASCII format.

3.6.4 File Output Format

The LDMS shall output well-formed, valid XML.

3.7 Online User Documentation and Help System Requirements

Proper operation of the LDMS requires knowledge of the user interface and system execution parameters. When an invalid parameter is given to the LDMS, program execution shall not commence. The LDMS shall alternatively call up the manual page reference and display its contents to the user.

3.8 Purchased Components

3.8.1 Development Reference Books

Development of the LDMS is aided by the use of reference books purchased expressly for this project:

The Perl CD Bookshelf : 6 Bestselling Perl Books on CD-ROM
by O'Reilly, Inc. Associates, August 1999. ISBN: 1565924622

3.9 Interfaces

3.9.1 Software Interfaces

Development of the LDMS is with the intent to use it as part of a comprehensive US Code delivery system designed for ease of use coupled with a polished presentation. Its operation is crucial since it is the first transformation applied to the US Code after release by the House of Representatives. As a software component, the LDMS shall strictly accept US Code in ASCII format as its input and output well-formed XML that is valid according to a custom designed DTD suited to the nature of the input data.

3.10 Licensing Requirements

The final product should be extendable at the source level by the client. Additionally, the issue of possible revenues generated by such extension of the product must be addressed. As much of the code may result in deriving from freely available sources, care must be taken to ensure that use of such code does not entail legal duties, which are inconsistent with possible future commercial use of the product. Therefore, a contract has been drawn up to address these issues.

3.10.1 Joint Authorship Agreement

The undersigned agree to the following:

1. That all code, documentation and other copyright-protected material produced in the course of this CS501 project (PROJECT MATERIAL) shall understood by all to be the work of joint authors and not as a work made for hire;

2. That the joint authors shall include all the undersigned, the CS501 students working on the project and Thomas R. Bruce;

3. That despite joint authorship there will be no duty on the part of the student authors, individually or as a group, to account for any return on subsequent commercial use or development of the PROJECT MATERIAL;

4. That, in contrast, should Thomas R. Bruce or the Legal Information Institute realize royalties or other direct financial return from licensing any of the PROJECT MATERIAL there will be a duty to account to the other joint authors for any such revenue net of costs; and

5. That the undersigned will use care to assure that the PROJECT MATERIAL does not incorporate code covered by copyright and licensed on terms that are inconsistent with unlimited noncommercial distribution.

3.11 Legal, Copyright and Other Notices

At this juncture, the final product shall be distributed without any warranty, express or implied, and without even the implications of merchantability or fitness for a particular purpose. The developers will make every effort to ensure that the product fulfills the requirements listed above. There will, however, be no legal duties to ensure any of them are fulfilled.

3.12 Applicable Standards

Currently, there is one external standard that shall be directly supported by the final product: the XML language definition.

3.12.1 XML Language Definition

The XML language is intended to be the successor to HTML, and is poised to become a universal structured document standard. As it is based on SGML, most (if not all) of the language has been defined already. For more information, see the W3C draft specification.

4. Supporting Information

4.1 Hardware Specifications for Leda

Processor : Intel Pentium II, 233 MHz

Memory : 128 MB

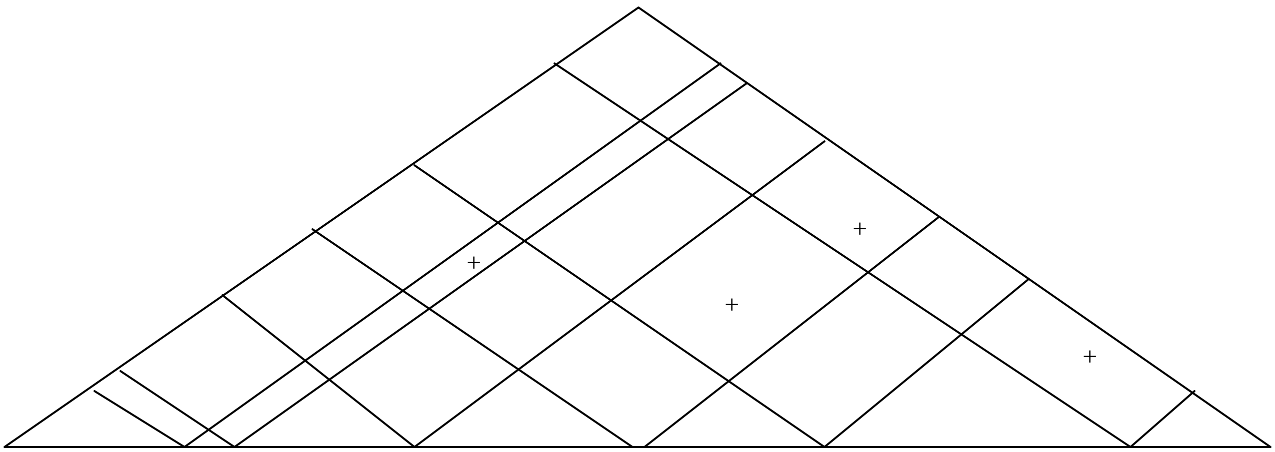
Hard drive: 28 GB

4.2 House of Quality

We used a house of quality model (figure 2) to help us define the engineering and development requirements that would be needed to satisfy client requirements. The house of quality below illustrates the relationship between functions our client requires and the engineering requirements we need to achieve or test the functions. The first column lists the various requirements that the client desires while the first row lists the various requirements that we derive from those requirements. "+" denotes the engineering requirement that must be employed to achieve a client's requirement. Using an example from our model in figure 2, one can see that to flag special characters, it needs to be recognized by pattern matching, tagged, and converted to unicode.

The roof of the house shows the relationship between the various engineering requirements. "+" denotes a possible synergy between the two engineering requirements. That is, positively connected engineering requirements are may be implemented together. Furthermore, improving one requirement also improves the quality of the other. A “-“ relationship indicates conflicting requirements such that improving one degrades the quality of the other.

The ratings of difficulty and importance rank that particular engineering requirement against other engineering requirements. Rank of 1 means it is the least difficult or least important. Rank of 6 means it is the most difficult or most important.



Engin. / Client	XSL	ASCII -> Unicode	Word Pattern Matching	Special DTD Tags	White Space Pattern Matching	State Machine
Appendices	+		+	+		+
Special Character	+	+	+	+		
Cross Ref.	+		+	+		
Structural Layout	+			+	+	+
Tables	+			+	+	
TOC	+		+	+	+	+
Catch Line	+		+	+		+
Notes	+		+	+		+
Next/Prev						
Graceful Failure			+	+		+
Magic Word	+		+	+		+
Difficulty	2	1	5	3	4	6
Importance	2	1	5	6	3	4
Least to Most (1 to 6)						
"+" Positive Correlation between two requirements.						

Figure 2: House of quality for the LDMS project.

In our model above, we have noted that the XML output will need to be parsed through an XSL processor which will evaluate and apply a presentation style for display in a browser. The XSL processor is not a deliverable, but a debugging utility that will quickly allow us to check for the correct (or otherwise) implementation of the client requirements marked with a "+" under the XSL column. Most of the work required to identify the major elements of the US code will be done by pattern matching as noted by the positive relationships between most of the client requirements and the pattern matching engineering requirement. Special DTD tags will need to be created for all the client requirements since they are closely associated with the content of the US code. White space is used extensively throughout the ASCII format of the US code to denote structure. Pattern matching of white space is therefore an engineering requirement identifying and listing components such as chapters or sections in a title, catchlines for those subdivisions and data tables embedded within text. In a manner similar to the relationships described above, quality, several requirements that need state machine analysis, are marked on the house of quality.

4.3 Description of Duties

Using the table of contents of this SRS document as a guide during meetings, all the LDMS developers contributed to the content of the SRS through discussion. Each developer was additionally charged with duties that oversee a major area of the delivery of these specifications to the client.

Ju Joh facilitated and scheduled all meetings, sent reminders and minutes of client meetings, introduced and explained the use of the House of Quality method of analysis, and will be presenting the SRS to the sponsors.

Sylvia Kwakye took notes during discussion sessions, wrote sections 1, 2, 3.2, 3.4 and 4 of the SRS, and did the final review and editing of the complete SRS document.

Jason Lee is responsible for putting the SRS document, presentation and all our email interactions online. He also edited the entire SRS document, wrote section 1.2 of the SRS, and will be presenting the SRS to the sponsors.

Nidhi Loyalka wrote the presentation slides 15-34, and edited both the entire SRS document and all of the presentation slides.

Omar Mehmood worked out the details of our copyright requirements with the client, wrote presentation slides 3-6, and 9-15, edited the entire slide presentation, and edited the entire SRS document.

Charles Shagong took notes during recitations, wrote sections 3.1, 3.3, 3.5, 3.6, 3.7, 3.8, 3.9, 3.10, 3.11, and 3.12 of the SRS, as well as review and revise the SRS document.

Brian Williams edited the entire SRS document, edited the entire presentation, wrote presentation slides 1, 2, 7, 8, and 35 through 42, and took notes at recitations and client meetings.