

# TLC2

Texas Learning and Computation Center - White Paper

WEB ○ PRINT ○ DEVELOPMENT

**TABLE OF CONTENTS**

<b><u>I. OVERVIEW</u></b>	<b><u>4</u></b>
<b><u>II. NEEDS ANALYSIS</u></b>	<b><u>5</u></b>
DEFINITIONS:	5
DISTRIBUTED CONTENT MANAGEMENT	5
UPLOAD AND DOWNLOAD CONTENT	5
HOT PAGE	5
SEARCHING CONTENT	6
CONTENT VERSIONING	6
DISTRIBUTED SECURITY MODEL	6
EVENT CALENDAR	7
KISS: KEEP IT SIMPLE AND STUPID	7
DESIGN TEMPLATES	8
WORKFLOW MODEL	8
REGISTERED USER SERVICES	8
DOCUMENTATION	9
TRAINING	9
CODING SPECIFICATIONS	9
SYSTEM ARCHITECTURE	9
<b><u>III. SPECIFICATIONS</u></b>	<b><u>11</u></b>
HARDWARE, SOFTWARE, AND NETWORK CONSTRAINTS	11
LIFETIME OF PRODUCT	11
<b><u>IV. ANALYSIS OF SOFTWARE SOLUTIONS</u></b>	<b><u>13</u></b>
ZOPE	13
DOMINO / LOTUS NOTES	15
INTERWOVEN	18
VIGNETTE	20
COLD FUSION	21
PHP, PERL, ASP, ETC.	21
<b><u>V. SOFTWARE COMPARISON</u></b>	<b><u>23</u></b>
MATRIX OF NEEDS	23
<b><u>VI. RECOMMENDATIONS</u></b>	<b><u>26</u></b>

<b>HARDWARE</b>	<b>26</b>
<b>SOFTWARE</b>	<b>28</b>
<b><u>VII. CONCLUSION</u></b>	<b><u>29</u></b>
<b><u>VIII. REFERENCES</u></b>	<b><u>30</u></b>

## **I. Overview**

This document will review the needs and wants of the TLC2 website. It is our intention to provide a detailed analysis of the requirements outlined by TLC. Sources for these requirements include the following: conference calls, tlc2\_contentmatrix.xls, TLC2Websitemap.pdf.

A comparative analysis of software components, applications, and development environments will be included to assist in the decision for the appropriate tool sets to develop, maintain, and administer the final product (TLC2 website).

Unless noted otherwise, this document refers only to Phase I development. Phase I development is defined as being a functional site with the following needs (below) operational to an acceptable degree. This document is not a final roadmap or promise for such features described. Rather, this document's intention is to provide a common ground of understanding in which both parties (TLC2 and Illust Designs) can share the common vision for the development of TLC2's web site. We expect this document will be modified and amended to provide the true goals of TLC2, and reasonable Phase I objectives.

Upon agreement of the elements within this document, Illust Designs will set out to jointly draft a design document with TLC2 that contains the details of functionality and the exact development approach to reach these goals. A design document will contain flow diagrams, workflow instances, server architecture and configuration (relating to this site only), content structure, look and feel, all graphical design, and even pseudo code where applicable.

## II. Needs Analysis

### **Definitions:**

**Content:** Content is any data object that is intended to be viewed or accessed from a browser via a web site. This may include, HTML, text documents, other docs (PDF), images, movies, and other sources.

### **Distributed Content Management**

The ability to create, store, and catalog content (documents, html, etc.) from any browser (location) and manage that content from any location.

#### *Possible use of CVS*

The use of CVS may be used to house versions of the Page Templates, CSS documents, and other external code elements. This may also be used for managing text-based content efficiently. Scripts may be setup in CVS to simply place current versions of content to a working file system directory that is viewable (more to come on this—diagrams of possible architecture).

### **Upload and Download Content**

This is the ability to add content, making it available for others. This feature must allow some, if not all, content to be cataloged or logically sorted. Content must then be available for viewers to download existing documents. The assignment of attributes to content must be present at a basic level. Note: not all documents may have attributes or metadata. Some content may simply exist on file servers depending on the data type and size of such data. The details of this will be determined at a later time.

### **Hot Page**

The PACI partners have developed a web page displaying statistics of various supercomputing resources within the partnership. TLC2 is very interested in developing a part of the site dedicated to this type of interactivity. Generating tools to mine this information from other sources are within the scope of this project. On the other hand, developing the means to extract this data from various TLC2 resources is beyond the scope of this project.

There are many great utilities for monitoring systems and passing back valuable information. The one that comes to mind is MRTG, a package that presents statistics about machines, networks, and even applications. We highly recommend exploring this as well as other methods to extract information such as CPU load, uptime, and machine status.

## **Searching Content**

This site must contain a central searching component that will allow contributed content and site content to be searched. Searching must also accommodate site section searches (i.e., just search publications, or research section). In addition, searching by author, subject (keyword), and date would greatly contribute the search features. Additional features will likely be search product specific.

## **Content Versioning**

Some content (to be identified) will have versioning features. It is imperative that the applicable content be versioned with a true versioning software such as CVS.

## **Distributed Security Model**

It is key that the security model be solid and flexible with the ability to designate separate security across distributed facilities, while at the same time possess the features to control the entire enterprise from a central source if needed. Any prospective application that cannot satisfy this core need should be eliminated as a possible solution. This distributed model must interact heavily with the site architecture, and therefore we recommend this security need should be part of the base application that will create the TLC2 site. We do not recommend using a separate security package that will in turn be integrated into the site application. This will prove to be more burdensome than not.

There are many possible roles that people may have while interacting with this site. Below are a few of those possible roles. The names and permissions can all change - the point is to illustrate the complexity of roles. We'll need to refine this during a design stage. It should be noted that there are two key elements of security: authentication and authorization. Authentication is the process of finding out that a user is, and authorization determines what that user can do.

This security model should accommodate the following structures (or something similar):

- **Viewer.** A person who can see any material Published on the web site.
- **Registered User.** A person who has registered / created an account on the site / facility. This user may have permissions to specific objects that a Manager's may assign permissions to.
- **Author.** A person who can add their own content and only edit that content. This person must request that a document be published (made visible to the public).
- **Editor.** A person who has all of Author's privileges plus the ability to edit anyone else's documents within their facility. This person must request that content be published.

- **Publisher / Reviewer.** A person who can do everything Editor can do, with the ability to publish any content within their facility section.
- **Super Publisher.** A person who can publish anything, anywhere.
- **Designer.** A person who can modify elements within the presentation layers.
- **Developer.** A person who can modify page templates and site architecture. Ability to checkout, update, and move code objects. Ability to stage new code on separate server instance for testing.
- **Manager.** A person who can do everything within the application realm.
- **Administrator.** Though not part of this model directly, is a person whose duty to integrate web applications to web servers, manage version control systems, and other supportive software components for the site.

It is important that a Role and Permissions model be flexible to where new security necessities can be satisfied with the creation of specific roles, while not breaking the existing security structure.

### **Event Calendar**

The complexity of calendaring software needs to be considered within this application. Most often, a calendar component is heavily integrated into the mail solution for an enterprise. The purpose of this calendar is to have a dynamic area displaying current calendar entries. Multiple people can enter these entries from multiple sources. For simplicity, we will take a central calendar approach for Phase I development.

### **KISS: Keep it Simple and Stupid**

When developing any website, one must resist features and focus on the core of what you are trying to provide. This site must be maintained, developed, and administered by as few people as possible. The addition of content must be driven by users and assigned facilities. For this to occur, we should take the 'path of least resistance' in development considerations. While the core of this web site must be elegant and flexible, we should not trade elegance for simplicity. It will be our approach to develop as much code / scripts / templates etc. in a simple, linear approach. This will be discussed in further detail within a design document. For now, let's note we have the following needs:

- Ease of administration
- Small set of tools for management
- Easy transfer of knowledge to new staff
- Fewest 'points of failure' possible

## **Design Templates**

There are several concepts behind the term ‘template.’ We should separate this into two distinct categories:

- **Presentation Templates.** These templates are the visual aspects of the site--the look-and-feel, the navigation, the level motifs, the consistent verbiage, etc. These elements may be changed a certain degree by individuals other than developers, and should require simple object text changes and not core template changes. This template design concept will be addressed in greater detail within a design document.
- **Object Templates.** Technically, even though Presentation Templates are Object Templates, it is important to differentiate the two. Object templates are elements that drive business rules, events, and logic for the presentation layers. These may also be controls for reusing sections of content across multiple sections of the site. For example a ‘What’s New’ box may be reused in multiple locations.

The ability to change and add templates will be considered a need for phase II. This is not to say that the architecture in Phase I will not support this, rather, we want to focus on the integration of presentation with the defined logic of the site during Phase I and not necessarily the ability to use custom management tools for modification of the Phase I branded look.

## **Workflow Model**

As with the security model, the workflow model is key to the onset of this development. We do not want to build future unforeseeable and complex workflow paths from scratch. Therefore, the selection of the site development environment must contain a foundation for building simple to complex workflow. An application that has this component integrated within the security model is even better. The use of roles and permissions on this level are key to the success of the application built.

## **Registered User Services**

Allowing registered users certain abilities within the site. In the future users may be fed news and events through subscriptions. TLC2 may also desire to allow all or a subset of users (such as students, researchers) to not only have links to their own sites, but to provide them an area with TLC2 for just that service. By doing so, TLC2 can leverage the searching and sharing objects within the user built area. Without a doubt, not a Phase I concern, but mentionable here.

- Subscription service
- Press room extensions
- User created site nested within TLC2 site. May or may not be forced to use templates defined for other areas of the site.

## **Documentation**

This is to state that above and beyond any existing documentation of the technologies used, we will provide documentation for the following:

- **Architecture Documentation.** This will explain and diagram how the entire system interacts at a high level. This will explore the dependencies, configuration settings, and what was generally discovered to work and not work throughout the development and integration of the project.
- **Development Documentation.** Above and beyond the basic architecture, this documentation will describe the why's and how's of the custom code created. In addition, all code will be commented in adequate detail where applicable (scripts, design templates, etc.).

## **Training**

The process of training will be ongoing throughout development, with a final training period after the system is built. We will dedicate as much time as needed to provide thorough training to developers and administrators. This may include code walkthroughs, application demonstrations, and code temporary modifications as hands-on experience. This may be achieved by on-site visits and/or collaborative discussions through other media.

## **Coding Specifications**

Once selection of the most beneficial development environment is made, we will detail the languages and versions this project will be based within. This will include languages, the versions of those languages,

## **System Architecture**

It is important to separate environments along the way of the product's lifecycle. There are three distinct zones to mention here:

- **Development.** This is an environment that contains just enough functionality to develop elements for the site as well as test them before implementation on a staging system.
- **Staging.** This environment will allow newly developed code to be placed within a replica of the production site. This step is critical to stop unforeseeable bugs that are not evident in a development environment. Note: some applications have this feature built in which allows development to occur within a production environment. The objects modified are versioned and not publicly available until the version is committed.
- **Production.** This environment contains all stable and released objects available for the site to use. This is what the public sees and uses.

Separate systems do not always mean physically separated. Having separate instances of the servers and other components is enough for most HTML development. If, however, there are processes that utilize a servers resources outside the scope of a web server application, it is advisable to develop on a physically separate system to avoid ‘crashes’ and run-away processes that can lock a server’s CPU—those dreaded infinite loops.

### **Browser Support**

A full range of browsers will be supported by this system. Keeping the design simple and beautiful will comply with the large variety of browsers available to the users.

Personal devices and cell phones have had a rocky development path since the idea of web accessible, personal devices. The first standards of WML and WAP have brought about too much diversity in specific support of features in these devices. What may supplant these markup languages is the idea of proxying. Sprint (and soon to follow, AT&T) decided that most websites were not good at WML support. They also found that the bandwidth available on these networks limited the amount of content one could push to a PDA. They chose a proxy service that pre-rendered the website and sent links with highly compressed bitmaps to the devices. This provides instant PDA access, with better data representation than WML/WAP specific sites.

### **III. Specifications**

#### **Hardware, Software, and Network constraints**

##### ***Linux/Intel as primary environment***

Linux was voiced as the primary platform for development. Since RedHat is the most widely used Linux distribution, we will assume a RedHat 8.0 web server. Intel based systems are the most supported hardware for Linux. We will also assume a dual processor, multi-Gigabyte, Intel web server with a high-speed Internet connection.

##### ***Web Server sits in DMZ***

With a growing user population, much like a city, the Internet has its dark element. Security is just as important for our servers as it is in our cities. Our assumptions will be a self-contained web server with the ability to be backed-up and securely remotely administered. At this time, we do not see the need for a network file system or large-scale user interaction via Unix logins on this machine. When we spec the solution, we will list all of the active TCP ports and their usage within TLC2's network De-Militarized Zone (DMZ.)

#### **Lifetime of Product**

The academic environment traditionally promotes the "proof-of-concept" principle. Rapidly developing and experimenting with solutions to all sorts of problems allows a researcher to converge on solutions from different angles, learning about the instruments and their limitations.

##### ***"An Appropriate Lifetime"***

Every project builds in an appropriate lifetime of the product under development. Software, and in this specific case, websites, are no exception to this rule. While designing the SCI Institute's website, we came up with a multitude of solutions and features to deploy to the users within and beyond the Institute. After our initial brainstorming session, we had a morass of features, needs, and wants to wade through as possible design directions. After working with the various interested parties, we worked it down to a set of needs, opting to add features in the future.

Versioning becomes key to adding features. Once the core services have been determined, subsequent versions add neat new features and patch current features. Some of the biggest time sinks in SCI were when people wanted something developed rapidly outside the development cycle. While possible when it only happens once or twice, this sort of feature creep slowed many of our efforts, especially in our documentation section.

***Software evolves, needs change***

With each software development cycle, often we get to a point where we need some fundamental changes. Through each revision of the software, the project cannot support requested features because of fundamental workflow and data design decisions. When enough of these features become needs, it is time to rethink the fundamentals. About every 3 to 6 years this is true in the software industry. In the realm of web development, this can be as soon as two years down the road. Just because you know this, doesn't mean you should rethink your fundamental design. The educational value of designing, deploying, and gathering feedback from a "short-term" solution provides you sufficient information for your next big step.

In the following software analysis, many of the solutions may be too large for TLC2. As TLC2 uses their website and add users, they may find that a larger system will scale far better for their future. By providing a small "stepping-stone" environment, TLC2 will be familiar with why large system solutions address the issue they do.

## **IV. Analysis of Software Solutions**

### **Zope**

#### ***History/Origination***

Zope is an open source Web application server built on Python with several core components built in C. The company was formally known as Digital Creations. Zope appears to be supported by a medium-large active group of users, and continues to grow daily. <http://www.zope.org> .

- Number of users. It is hard to say. Certainly, the Zope community will continue to grow. We assume this based on what Zope offers as a free alternative to comparable expensive suites of applications.

#### ***Purpose.***

Zope was created to extend the CGI architecture concepts for web-based content. That is, the increasing need to share and modify documents across the Internet drove the core development of this product. Zope aims to satisfy allowing disparate content data-types to be managed in a distributed model, while at the same time providing all functionality for such actions through an Internet browser.

#### ***Evolution.***

Because Zope is such a new application, it is hard to say where the user base will drive this product. However, it is certain that considerable contributions will continue within the Zope architecture, extending functionality and refining the core.

#### ***How it works***

Once installed, the web-based interface allows developers to create the ‘visible’ site from within itself. With the use of roles, managers can assign permissions to every section of the site. Once a simple security model is setup, individuals can begin contributing content immediately.

Zope can be used to create web sites that are easy to manage based on a simple yet extensible template language called DTML (Document Template Markup Language). TAL (Template Attribute Language) recently expanded the page template abilities within Zope and has lead to an increasing number of sites using Zope.

#### ***Plug-ins and Third Party Extensions***

Presentation and data-type management: While Zope is very efficient at what it does, the core interface to the content users and viewers lacks considerable ‘out-the-gate’ presentation flexibility.

- **Plone.** Good news. Because Zope is open source, the endeavors of many are frequently made available to the public for free. A recently new product called Plone (see [www.plone.org](http://www.plone.org)) can extend the functionality of the basic Zope installation and immediately give developers a framework (or examples) to work within. Plone takes full advantage of reusable page templates that yield a cleverly built, consistent looking site. For examples, see: <http://plone.org/about/sites>. Organizations such as NATO and companies like CBS New York have based entire systems around the key features found within Plone on a Zope platform.
- **Content Management Framework (CMF):** The CMF is a separate product that can be installed within the Zope server. CMF extends the functions of Zope to allow for new data content types along with handy management and attribute assignment for that content. This is such a critical component of Zope that it will likely be merged into the core Zope product by next release (Zope Version 3.\*). To recreate what CMF does for Zope would be a daunting task. Therefore, the use of this free extension to Zope yields substantial savings in development time for TLC2. Unfortunately there is little comprehensive documentation on this product itself, but the Authors have resolved to complete the online book sections shortly. Much of this extension is self-explanatory and can be understood through heavily used user groups, or simply following the open source code through. See <http://cmf.zope.org>.

#### ***Support and Availability***

While there is no direct vendor support (as with many open source projects) there are several new books written on the subject. There is also an online Zope Book with user comments and discussions throughout its examples and explanations, found at:

[http://www.zope.org/Documentation/Books/ZopeBook/current/index\\_html](http://www.zope.org/Documentation/Books/ZopeBook/current/index_html).

Despite the lack of vendor support, it is nice to know that the Zope architecture can be extended with de facto languages, such as Python. There is little core proprietary construction. In addition, because Zope is open source, it is possible to review every detail of its construction, and modify components to suit the builder's needs. The newest code is always available to download from within its SourceForge CVS archive on the Internet.

#### ***Licensing***

This license has been certified as open source. The Free Software Foundation (FSF) has also designated Zope as GPL compatible. Certain Terms apply if modifying core code for distribution.

#### ***How Does It Meet the Needs?***

There is no doubt that this is production-worthy software. Zope will clearly satisfy all Phase I needs, without compromising any of the Phase II needs, and probably most unseen needs at this point.

### ***Key Features***

Many key features make the trio, Zope, CMF, and Plone, a robust environment to satisfy TLC2 needs. Noteworthy features include:

- Completed and functional presentation layer (called Skins) that can be used as a starting point for any development.
- Built in support of dynamic sections called Slots for modular and reusable components.
- Ability to choose web server architecture. Plays nice with Apache on Linux.
- Intuitive Scripting and HTML extensions.
- Rooted heavily in proven language (Python).
- Continuous community addition of new products and extensions. Free.

### ***Drawbacks***

- Beginning of its lifecycle.
- Most objects (documents, scripts, etc.) are stored within a single object repository. However, publishing file system directories allow for alternate methods of document publication and file access.

## **Domino / Lotus Notes**

### ***History/Origination***

Domino, formally known as Lotus Notes, began as a simple collaborative tool in the early 1990's. The original design appears to have been driven to satisfy the needs of the traveling salesperson. During its early years, Lotus packaged Notes within a suite called Lotus Suite, which competed alongside Microsoft Office. Both applications had email, spreadsheet, word processor, and database elements. The need to share documents and work 'off-line' gave birth to a very clever distributed replication model: Notes, the database component of the suite. Lotus created such a powerful replication model within Notes that no one has yet to surpass its ability to sync content and manage documents in an Object Repository framework. So impressive was its security and replication model, that the CIA chose this application to share documents within various departments of its organization. This exposure eventually led to a synergetic merger with IBM in the late 1990's. It wasn't until this merger that the product became web-enabled, with a branding change of Domino. Unfortunately, the core architecture did not work well for web-enabling its content framework. This resulted in a long re-development process spanning several major releases, until they finally ended up with a truly dynamic web-design environment that could leverage the product. This is known as the Gold 5 release. Recently, WebSphere (IBM's commerce web server solution) was introduced to serve as the middle tier engine for Domino.

***Purpose.***

The core purpose of the Domino application environment is to enable large enterprises to collaborate and manage content across multiple levels of business with workflow and replication abilities built into the core. There is not a better product on the market to accomplish such a task to this day. The power of Domino lends itself well to intranets. However, as an Internet tool, its scalability tends to fall short.

Most organizations are introduced to the power of Domino after migrating to Notes Mail. Because the mail component is built into the database / object repository construct, the Domino web server is instantly available to be used. It never takes long for companies to understand the power notes can provide to internal needs (work flow, document sharing, discussion boards, web site templates are all built-in features out of the box).

***Evolution***

It is like that Domino will continue to thrive as a popular application, if not only for it's email capabilities. The web-enabled component is starting to migrate entirely to intranet needs, while new server products such as WebSphere gain market share in the middle-tier and web server sectors. This is a stable product that is not going anywhere anytime soon.

***How it works***

Domino is composed of a Desktop Client (what end-users have installed on their PC's), a Designer Client (where developers spend all of their time), and an Administrator Client. The Desktop and Admin client are applications that run primarily on Microsoft Windows environments. In addition to this, the actual Domino Server can reside on nearly any operating system. NT, AS400, Linux, Sun, to name the prominent ones.

The management of the server can occur within any other OS via the Admin Client (most choose a Microsoft Windows product to do this). Because of the distributed replication environment, it's possible to administer, design and develop, or just use email from any location, such as a laptop anywhere you have an Internet connection. The Design Client contains a trimmed version of the web server software that enables rapid development of applications as if you were developing them on a production server. Once a change is completed (i.e., a new navigation tab) it can be replicated to the production server within seconds and be made live, instantly.

***Plug-ins and Third Party Extensions***

Domino leverages many technologies within its container model. For example, Domino ships with tools for easy deployment of Java, EJBs, and interfaces for relational database connectivity (ODBC, JDBC) to Oracle, DB2, SQLServers, and many others. It doesn't play that nice with Microsoft products, such as IIS.

### ***Support of and Availability***

There is a tremendous reservoir of Domino professionals and developers to draw from. There is an equal amount of publications focusing on specific types of development within the Domino arena; probably in the hundreds by now. The Domino community in Texas is extremely healthy, with consultants and end users in the thousands.

The users of this technology tend to be large corporations spread across different architectures and perhaps even countries.

Training for use of the development tools and administration abound. There are hundreds of Lotus Business Partners across the United States and all can refer anyone to nearest training facility. Training for development is usually \$1000 to \$2000 per week of off-site training. Most companies keep one or two developers and administrators on staff, while using professional consultants and developers to assist in Mission Critical needs. This is commonly a mentor process for internal staff.

As for support, a license of the software guarantees a fixed amount of support from IBM's technical support department. From our experience, it can be hit and miss with problem resolution this route, which is why so many Lotus Business Partners can exist within a single product market space.

### ***Licensing and Pricing***

IBM sells seats to this server software based on the number of processors within the installation server. On top of this, there are three distinct classes of licenses:

- **Mail Server.** Just that. Fees per user. No application development abilities.
- **Utility Server.** Application server. Not designed for extensive user interaction. EDU list price, \$1,000.
- **Enterprise Server.** Unlimited users, including email, web servers, and development. Unlimited use of Client software (Desktop, Designer, and Administrator). EDU list price, \$5,000 (compared to corporate price of \$11,500).

### ***How Does It Meet the Needs?***

The Domino set of applications and tools can certainly meet the needs of TLC2. However, there is a large amount of overhead associated to components that may never be used. This gives us the impression that the product is actually too much for the defined objectives of TLC2.

### ***Key Features***

Below are a few of the features Domino contains that provide extra value to TLC2's needs:

- **RAD.** The development environment allows for a truly rapid development. A Domino database file can be broken into a template side and a data side. The template can be turned into a separate database file which can then be worked with aside from production data (documents / content). This allows a developer to alter design elements, such as the look and feel, apart from data elements that can sometimes be difficult to work with if large amounts of documents exist. A change to a design template can then be applied to a development, or test, database. Once the design changes are reviewed, that same design template can then be replicated to the production servers via any desktop development workstation—even a laptop.
- **Prototyping.** Domino is an effective prototyping tool for web ideas, though an expensive one if that's its only purpose.
- **Security.** The security model is unsurpassed. IBM and Lotus take great pride in the security of their products. In fact, Zope's entire security model is based almost identically to this one (it's only missing group abilities).

#### ***Drawbacks***

- There is no good versioning mechanism for objects, both content and design. Copies of the design templates and databases must be archived separately for adequate versioning of application software.
- Because Domino attempts to solve so many solutions as a packaged product, many features remain idle, and, unfortunately cannot be completely turned off. This tends to make Domino a little slower as a web server than other servers. Hence, it is acceptable as an intranet solution, and not so for Internet uses, unless the workload is light. We should note, however, that with each release this problem is diminishing.
- Domino can be an expensive road to travel. Consultant costs in Texas can be \$150+ an hour for basic support. Licensing, even for educational uses, is steep. Training can be expensive, and administration can be time consuming.
- There are many tools and data applications that can extend the functionality of a Domino site, but these are rarely open source. Everyone wants to make a buck with this software and few share source code freely.
- Administration of a Domino server can be overly complicated. This server must be monitored extensively. This is nearly one person's full time job as development grows and content contributions are made.

#### **Interwoven**

##### ***History/Purpose***

Started in 1995 by Peng T. Ong, Interwoven built to address the needs of large-scale content management systems. Peng received most of his experience as co-founder and chief architect of Electric Classifieds, the people that brought us

Match.Com. Interwoven has embraced XML as their fundamental data representations backed by a fast database server. Overall, Interwoven provides solutions for Fortune 500 companies and large public institutions with thousands of employees and hundreds of thousands of users.

***How it works***

Interwoven developed their flagship product, called TeamSite, as a content management system. Within its XML infrastructure is a rich development environment designed to leverage the abstractions necessary to produce complex, content rich websites. Interwoven augments its product with OpenDeploy, an enterprise deployment tool, MetaTagger a content tagging and organizational

***Plug-ins and Third Party Extensions***

With over twenty premier partners including Oracle, IBM, SAP, Interwoven's third party extensions cover many different areas of data access and data repository. In addition, Interwoven works through many ISV's who consult for customer software development projects.

***Support of and Availability***

Varying degrees of support depending on the complexity of the project.

***Licensing and Pricing***

Interwoven negotiates licensing and pricing on a per project basis. Interwoven is not interested in giving a general price.

***How Does It Meet the Needs?***

Out of the box, Interwoven delivers a complete content management system. Interwoven provides all of the features TLC2 needs. Unfortunately, Interwoven is priced and designed for large company with needs above and beyond that of TLC2. The learning curve for one group on campus was over a year of training and experimentation with two developers, one designer, and two managers. That team scrapped the project to go with a "roll-your-own" solution after their users refused to use the complex Interwoven content management system. This may have not been the fault of Interwoven, but with the limited resource of people traditional within university research groups, TLC2 may find themselves tens of thousands of dollars short, two years down the road, and little or no progress towards their content management needs.

***Key Features***

Interwoven's suite of products fulfill the needs of content management within large-scale content management systems. Here's a bulleted list of some of the features described in their PDF on TeamSite:

- **In-line Content Editing:** The ability to edit the content of a page with some sense of what template you are using

- **Merge, Diff, and Rollback:** CVS like features for managing, versioning, and rolling back to previous versions of the site.
- **Hybrid Architecture:** Both a database repository and file system access when and where it makes sense.
- **Access Control:** User and group based access control for optimal site tool management.
- **Rules-based Searches:** TeamSite allows you to search metadata and templates within a site.

## **Vignette**

### ***History***

Vignette started nine years ago developing custom Internet publish tools for customers. Soon they found that a generalized content management system could be achieved and was desired by customers. They purchased publishing technology from CNET networks wrapped it in a different package, and sold it as one of the first commercial content management technologies. Over the years and through several key partnerships, Vignette has grown to be one of the top content management technologies available to large businesses.

(<http://www.vignette.com/contentmanagement/0,2097,1-1-30-1458-81-118,00.html>)

### ***Purpose***

Vignette is a content management system optimized to run on Windows/Intel and Solaris/Sun machine. Its initial design is still reflected by today's product: web-centric content delivery and management.

### ***How it works***

Another robust content management system, Vignette manages its content via a combination of XML, Java, ASP, and a database.

### ***Plug-ins and Third Party Extensions***

Supporting not only other enterprise software such as Oracle and .NET, Vignette also lets you use your favorite XML editor to develop content. From Macromedia Dreamweaver to Adobe GoLive, Vignette provides a good interface for XML content upload.

### ***Support of and Availability***

As in Interwoven, there are many ISV's and Vignette has a whole custom integration team to provide a complete solution to your data needs.

### ***Licensing and Pricing***

Much like Interwoven, Vignette negotiates licensing and pricing on a per project basis.

### ***How Does It Meet the Needs?***

Vignette fails to deliver affordable solutions to small businesses. From what I've read from web postings to what's on their website, Vignette is a cadre of web tools strapped together in a content management system. Most of their literature was devoid of feature listing and they painted each whitepaper from a viewpoint of a specific customer. This leads me to believe that their process for software development have not progressed as far as Interwoven. They most likely rely heavily on J2EE, ASP, and Java for scripting and dynamics.

### ***Key Features***

Vignette, as stated above, is another large-scale content management system. Developed for large companies with big IT staffs, the development environment provides big tools for managing, deploying, and tracking content.

## **Cold Fusion**

### ***Overview***

We will not go into great detail regarding Cold Fusion because it falls short in so many ways for the needs of TLC. To summarize, Cold Fusion is a great front-end tool when it comes to dynamic data presentation. This product works best when all content comes from a database, while the CF presentation code simply servers it up through an interpretive markup language similar to HTML. This code is very powerful when simple data sets need to be presented with a pretty front-end. Cold Fusion, however, does not have any notable security built into its framework, nor does it contain any recognizable form of workflow. In fact, Cold Fusion does a very poor job at handling any type of business logic. This is while Allaire introduced the its middle tier server to handle JSP, EJBs, and Java. This extended the model, but added a layer of complexity because the products are actually separate entities. It runs best under Microsoft IIS and typically lightening fast, even with making complex queries to a database.

This product is not suitable for the TLC2 site.

## **PHP, PERL, ASP, Etc.**

As the web evolved, many system administrators and developers wanted to quickly roll applications without having to compile code or adhere to any language formalisms. PHP and Perl were brought up from the UNIX side, and ASP was developed in the Microsoft suite of tools. While all of development environments are powerful tools, they have no underlying content management

concepts. The concepts of workflow, inheritance, and versioning can be added to these solutions with great difficulty.

Perl, originally a replacement for command line scripting applications, has its main strength in regular expressions. Beyond the command line or rapid development environments, Perl suffers greatly. PHP seems to be a fairly robust development environment for this type of work, but lacks elegance and language formalisms required to use concepts like object inheritance and reuse. ASP is derived from Microsoft Visual Basic, albeit a sufficient tool for small user bases, scales very poorly to be useful in the web environments discussed in this document.

Traditionally, complex Perl and PHP sites become nightmarish to maintain once a site evolves into more than just a few pages of slightly dynamic information. Certainly, TLC could develop its site through such tools and avoid costly license fees, but to build objects that can inherit a separate security model, or rely on an external workflow routing system would require a monumental development project that would likely result in something similar to Zope anyway.

## V. Software Comparison

### Matrix of Needs

This comparison only reviews elements associated to our web application needs.

	<b>Zope</b>	<b>Domino/Notes</b>	<b>Cold Fusion</b>	<b>Interwoven</b>	<b>Vignette</b>
Distributed Content Management	Yes	Yes	<b><u>NO</u></b>	Yes	Yes
Native Application Versioning	Yes	Yes	No	Yes	Yes
Native Content Version Control	No. Can be built with ease, or use CMF.	No. Can be built with ease.	No. Must be external application (i.e., database).	No, Must purchase additional product.	Yes
Ability to Stage Release (can you push new version to separate server prior to production release?)	Yes. Need separate instance of Zope.	Yes. Template applied to stage database.	Yes. Must use CVS (equivalent) to roll changes to separate server.	Must purchase addition product.	Must purchase additional product.
Content Upload / Download Controls	Yes. Even more extended in Plone.	Yes.	No. Only basic HTML.	Yes	Yes
Content Attribute Assignment (Document is Word, Text is HTML, Text is structured, etc.)	Yes	Yes, native features need modification	No	Yes	Yes
Logical Categories for Content (Subject, Metadata, Author, Date)	Yes, by default.	No. Can be built easily.	No	Yes	Yes
Search: All Content Searchable	Yes. Full text indexing.	Yes. Full text indexing.	No	Must purchase additional product	Must purchase plug-in
Search: Indexing	Yes. By any document attribute.	Yes. By any document attribute.	No	Yes	Yes
Search: Scheduled Tasks	No	Yes		Yes	Yes

Texas Learning and Computational Center Website White Paper

	<b>Zope</b>	<b>Domino/Notes</b>	<b>Cold Fusion</b>	<b>Interwoven</b>	<b>Vignette</b>
Native Distributed Security Model (can it be done?)	Yes	Yes, but server based.	No. Can it be done? Yes, with external database and custom, elaborate user schema.	Yes	Yes
Native Owner, Author, Editor Roles	Yes	Yes	No.	Yes	Yes
Native User Session Persistence	Yes	Yes	No. Can it be done? Yes, with external database or simple cookies.	Yes	Yes
Subscription List Services	No	Yes	No	No	No
Built-in Email Sending (SMTP)	Yes	Yes	Yes	Yes	No
Replication of Content	No. However, all objects can be exported. This is not automatic.	Yes	No	Yes, with an additional package	No
Replication of Design Templates	No. However, all objects can be exported. This is not automatic.	Yes	No	Yes	No
Global Calendar Entry and Publication	Yes	Yes	No	No	No
Decentralized Administration	Yes	Yes	No	Yes	Yes
Tool Set for Administration via Web	Yes	No Existing features could be extended for basic admin. May be complicated.	Yes	Yes	Yes
Design Template Development	Yes	Yes	No	Yes	Yes
Design Template Versioning	Yes	Yes	No	Yes	Yes
Reuse of Design Objects	Yes	Yes	Yes	Yes	Yes
Object Inheritance	Yes. Unique process called Acquisition	No	No	Yes	Yes
Workflow Engine	Yes	Yes	No	Yes	No

Texas Learning and Computational Center Website White Paper

	<b>Zope</b>	<b>Domino/Notes</b>	<b>Cold Fusion</b>	<b>Interwoven</b>	<b>Vignette</b>
Trigger Based Events	Yes	Yes	No	Yes	Yes
Schedule Events Non-OS (Scripts, etc.)	No	Yes	No	Yes	Yes
Direct Vendor Support	No	Yes	Yes	Yes	Yes
Training for development and administration	Yes. Non-vendor.	Yes	Yes	Yes	Yes
Documentation	Yes	Yes	Yes	Yes	Yes
Published Books for Development / Admin	Yes	Yes	Yes	No	No
Native Database or Repository	Yes	Yes	No	Yes	Yes
Database Connectivity (ODBC or JDBC)	Yes	Yes	Yes	Yes	Yes
Native Object API	Yes	Yes	Yes	Yes	Yes
Extended HTML Scripting (In Line)	Yes	No	Yes	Yes.	Yes
Server Side Scripting	Yes. DTML and TAL (proprietary), Python, PERL.	Yes. Java, LotusScript (Proprietary), Com,	No. Can call External Objects (Java, EJB) for middle-tier	Yes, Java.	Yes, Java, ASP.

## **VI. Recommendations**

### **Hardware**

We recommend deploying two web servers. One of these web servers will be in production, and the other will be the test and development server. Production, in this case, refers to the server containing web services that would be deployed for customers to see. This machine will sit outside of the DMZ (the demilitarized zone), to preserve network security. The development server provides a testing facility for new changes and additions to services for TLC2. Finally, this test machine provides a quick backup for the primary machine in case of critical failure.

As specified earlier in this document, we assumed both dual processor Intel machine running RedHat 8.0 or equivalent. In order to meet the needs of our recommendation, here are the specifics of this requirement:

- 2 CPU Intel Xeon machine
- 2 GB of RAM
- RAID5 SCSI controller
- 5 x 72Gb 10000RPM SCSI drives
- rack mountable (2U form factor, slide rails, etc.)
- Intel Pro1000 Gig-E card
- DVDROM drive
- Floppy drive
- Redundant Power supplies
- UPS protection (if you don't already have it)

We chose not to recommend AMD or IA64 machines for a couple of reasons. Not many vendors provide hardware support for a dual CPU AMD machine. The IA64 architecture, albeit tested in research environments, still has not been fully proven in industry. There have been reports outlining significant problems with maintenance among the research community with clusters of these machines. Finally, pricing and availability of the Xeon CPU makes it attractive for a production environment.

In the next few years, IA64 and AMD will become serious platforms that vendors will support. In the latest Fortune magazine, there was talk of traditional Intel vendors like Dell potentially embracing AMD if IA64 fails to meet its mark. If TLC2 waits this vendor war out, in 18 months to two years, when its time to start the upgrade on these systems, they will have a wider range of options to choose from and proven technology to depend on.

### ***Support***

In order to avoid excessive hardware failures, we propose a hardware solution that will be powerful enough to run the applications and at the same time provide a solid support agreement. We are familiar with Dell, and have been happy with their level of hardware support. There are other vendors that would work equally well, and if you were to go with them here's what we would recommend for purchasing and support:

- Dell
- IBM
- Penguin Computing

In any case, be sure the vendor will give you 24x7 support, overnight shipping of new parts, and no hassles. When a core service, such as a web server, fails, you don't want a local retailer saying that it will take a week before your web server comes back online.

### ***RedHat***

Traditionally, RedHat does not support RedHat Linux distributions. Their retail box is just the OS, no support contracts. You can't even purchase a support contract for RedHat 8.0 when you buy with a machine that ships with it. Most hardware vendors only guarantee that the hardware in the machine has been tested with Linux. Many customers and software vendors have complained bitterly about this lack of support, or lack of a "qualified" or "certified" platform.

In the last year or two RedHat has picked a good direction for their server offerings. They now have a product called Advanced Server, which is a "certified" platform for various Linux applications. You can buy a support contract through RedHat for this product. TLC2 may be interested in purchasing this product from RedHat to ease administration and support even further.

### ***Backups***

We have assumed that the hardware backup solution will be provided by TLC2. RedHat Linux is compatible with many backup software clients such as Veritas.

## **Software**

### ***Apache***

Apache is one of the most widely used web servers on the Internet. In fact, the most recent data indicates that over 60% of the web servers on the Internet are Apache or Apache based web servers. Most performance tests also show that Apache is one of the highest performance web servers available.

### ***Zope***

There is no doubt that Zope satisfies all of the needs for the TLC2 website requirements, as well as providing solid framework for future development. We recommend Zope as the development / publishing environment for the following main reasons:

1. Open Source.
2. True web-based distributed content management
3. True web-based administration and development environment (no other product can boast this). This allows development and maintenance to exist within a distributed structure.
4. Built on non-proprietary platform and developed in proven language (Python)
5. Extremely flexible acquisition based security model that is truly distributed.
6. Easy integration to larger database systems as TLC2's needs grow
7. Built in search objects that rival many third party extensions for web sites.
8. Easy to maintain.

### ***Security***

Security for Zope and all web services will be run using apache's modssl module. In order to provide a secure connection, TLC2 must purchase one or more SSL certificates. We recommend one of the big three Certificate Authorities: Verisign, Thawte, or RSA Data Security.

### ***E-mail and Calendaring Integration***

Out of the box, Zope is not an email and calendaring integration solution. We recommend developing an "all-in-one" calendaring, email, and directory solution. There are various providers of these solutions, and it would ease administration of these services. Possible directions are Novell's GroupWise, Sun's One Solutions, a Linux based solution, IBM's Notes Email, or Apple's MacOS Xserve.

## **VII. Conclusion**

When TLC2 and Illust Designs started this process, we were not comfortable with Zope as a solution based on our precursory review. Our broad experience with many of these Linux based, web application environments makes us weary to recommend such products because they are never as good as a fully integrated solutions like Lotus Notes. Historically, Open Source applications and servers tend to fall short on critical business needs because they are not driven by the appropriate capital to yield truly versatile products.

When we began the investigation on this whitepaper, we expected to come up with a list of reasons to go with a Lotus Notes or a Notes-like solution. However, to our surprise, we are very convinced that Zope possesses the breadth of features and robustness to provide a very healthy foundation for this project.

Zope is clearly a serious direction in content management and creation systems. It has a great developer base for an Open Software product, has companies and other open software solutions developing products based on it, and provides a better environment for development than commercial products such as Cold Fusion and ASP.

## VIII. References

### Some Title Here:

Why roll your own content management tools:

<http://www.truerwords.net/157>

[http://www.inside.com/product/Product.asp?pf\\_id={78AE4C3E-A0F6-4DEF-8678-1C4FB74A67CF}](http://www.inside.com/product/Product.asp?pf_id={78AE4C3E-A0F6-4DEF-8678-1C4FB74A67CF})

Why Interwoven and Vignette are potentially a bad way to go:

<http://techupdate.zdnet.com/techupdate/stories/main/0,14179,2852735,00.html>

“[Vignette](#) and [Interwoven](#), offer industrial-strength solutions for organizations willing to invest hundreds of thousands or more (often much more) in licensing, customization, and hardware. For many customers, these behemoths have proven overwhelmingly complex to deploy and manage”

Do you need a CMS?

<http://www.evolt.org/article/MartinB/20/5127/>

PHP v. <a language>

<http://www.php.net/manual/en/faq.languages.php>

This article brings up a good point. PHP, ASP, Perl, Cold Fusion, and the like all require some, if not a lot, of programming abilities. None of these environments are content management systems, they are just programming languages.

Content Management as an assignment? Check out Philip Greenspun’s website:

<http://philip.greenspun.com/teaching/psets/ps3/ps3.adp>

More Philip Greenspun: Interfacing a Relational DB to the Web

<http://philip.greenspun.com/wtr/dead-trees/53012.htm>

<http://philip.greenspun.com/panda/databases-interfacing.html>

His new book:

<http://philip.greenspun.com/internet-application-workbook/>

A cool magazine on content management:

<http://contentmanagementadvisor.com/>

### Zope / Zope Related

CMF (Content Management Framework) review:

<http://cmf.zope.org/>

Zope.com provides extensive training and support for Zope now:

<http://www.zope.com>

Zope's Development Zone (product releases, etc.):

<http://dev.zope.org/>

Zope's core philosophy and evolutionary objectives:

<http://dev.zope.org/Resources/ZopeRoadmap.html>

## **Lotus / Domino**

About Lotus

<http://www.lotus.com/engine/jumpages.nsf/wdocs/aboutlotus>

Content Management Software:

<http://www.lotus.com/lotus/offering4.nsf/wdocs/knowledgehome>

Collaborative Portal Solutions (WebSphere based):

<http://www.lotus.com/lotus/offering4.nsf/wdocs/knowledgehome>

Lotus Workflow Overview:

<http://www.lotus.com/products/domworkflow.nsf/a1d792857da52f638525630f004e7ab8/b7616e4aacf18afa85256a18006d6c10?OpenDocument>

Lotus / Domino Object security structure:

<http://www.developer.ibm.com/tech/faq/individual/0,,2:12968,00.html>