CALIFORNIA STATE UNIVERSITY, NORTHRIDGE

THE DEVELOPMENT OF AN ELECTRONIC THESIS AND DISSERTATION
SYSTEM FOR CALIFORNIA STATE UNIVERSITY, NORTHRIDGE

A graduate project submitted in partial fulfillment of the requirements
For the degree of Masters of Science in Computer Science

By

Katherine Vigna

December 2011
The graduate project of Katherine Vigna is approved:

___________________________________________  __________________

Gloria Melara, D.Ed.  Date

___________________________________________  __________________

Steven Fitzgerald, D.Sc.  Date

___________________________________________  __________________

Jeff Wiegley, Ph.D., Chair  Date

California State University, Northridge
ACKNOWLEDGEMENTS

Thank you to Dr. Fitzgerald for involving me in this project (even though it has nothing to do with parallel programming or compilers). Thanks to Dr. Wiegley, Dr. Melara, Karl Rink, and Harold Spencer for their continual help and guidance throughout the development and thesis process, and to everybody else in Academic Resources and Planning, the Library, and IT for helping move this project forward. Finally, thank you to everyone at GRIP for being the most patient clients ever!
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signature Page</td>
<td>ii</td>
</tr>
<tr>
<td>Acknowledgments</td>
<td>iii</td>
</tr>
<tr>
<td>List of Figures</td>
<td>vii</td>
</tr>
<tr>
<td>Abstract</td>
<td>iv</td>
</tr>
<tr>
<td>Chapter / Section 1: Introduction</td>
<td>1</td>
</tr>
<tr>
<td>Existing Literature on Electronic Thesis and Dissertation Systems</td>
<td>1</td>
</tr>
<tr>
<td>The CSUN ETD Project: Engineering a New ETD Workflow System</td>
<td>2</td>
</tr>
<tr>
<td>Chapter / Section 2: History and Business Context of the CSUN ETD System</td>
<td>5</td>
</tr>
<tr>
<td>Chapter / Section 3: Current CSUN Thesis Workflow</td>
<td>7</td>
</tr>
<tr>
<td>Thesis/ Project approval – Content</td>
<td>8</td>
</tr>
<tr>
<td>Thesis / Project Approval – Formatting</td>
<td>8</td>
</tr>
<tr>
<td>Thesis/ Project Approval – Submission</td>
<td>9</td>
</tr>
<tr>
<td>Thesis / Project Binding – Payment</td>
<td>10</td>
</tr>
<tr>
<td>Thesis / Project Binding – Binding and Distribution</td>
<td>10</td>
</tr>
<tr>
<td>Thesis Storage – Library Workflow</td>
<td>10</td>
</tr>
<tr>
<td>Chapter / Section 4: ETD / Digital Workflow</td>
<td>13</td>
</tr>
<tr>
<td>Thesis / Project Approval – Planning Form</td>
<td>13</td>
</tr>
<tr>
<td>Thesis / Project Approval – Content</td>
<td>14</td>
</tr>
<tr>
<td>Thesis / Project Approval – Formatting</td>
<td>14</td>
</tr>
<tr>
<td>Thesis / Project Approval – Submission</td>
<td>15</td>
</tr>
<tr>
<td>Thesis / Project Binding – Payment and The Bindery</td>
<td>16</td>
</tr>
<tr>
<td>Thesis Storage</td>
<td>16</td>
</tr>
<tr>
<td>Chapter / Section</td>
<td>Title</td>
</tr>
<tr>
<td>-------------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>5</td>
<td>Benefits of Moving to an Electronic System</td>
</tr>
<tr>
<td></td>
<td>Benefits of Making the Graduate Studies Workflow Electronic</td>
</tr>
<tr>
<td></td>
<td>Benefits of Thesis Storage in ScholarWorks</td>
</tr>
<tr>
<td>6</td>
<td>Developing ETD – Decision Criteria and Challenges</td>
</tr>
<tr>
<td></td>
<td>Characteristics of Web Application Development</td>
</tr>
<tr>
<td></td>
<td>Characteristics of the University Environment</td>
</tr>
<tr>
<td></td>
<td>Application of Decision Criteria</td>
</tr>
<tr>
<td>7</td>
<td>ETD Development Process Model</td>
</tr>
<tr>
<td>8</td>
<td>Requirements Engineering</td>
</tr>
<tr>
<td></td>
<td>Defining Initial Requirements: Analyzing the Business Context</td>
</tr>
<tr>
<td></td>
<td>Workflow Capture</td>
</tr>
<tr>
<td></td>
<td>Organizational Setting and Impact on Users</td>
</tr>
<tr>
<td></td>
<td>Iteratively Redefining Requirements and Controlling Scope Creep</td>
</tr>
<tr>
<td></td>
<td>Reverse Scope Creep – Losing Sight of User Acceptance</td>
</tr>
<tr>
<td></td>
<td>Involving Stakeholders and Negotiating Expectations</td>
</tr>
<tr>
<td></td>
<td>Iteratively Expanding Stakeholder Groups</td>
</tr>
<tr>
<td></td>
<td>Risk Mitigation</td>
</tr>
<tr>
<td></td>
<td>Unanticipated User Groups</td>
</tr>
<tr>
<td></td>
<td>Involving Users Too Late in the Process</td>
</tr>
<tr>
<td></td>
<td>Negotiating Expectations</td>
</tr>
<tr>
<td>9</td>
<td>Product Implementation</td>
</tr>
<tr>
<td></td>
<td>System Architecture</td>
</tr>
<tr>
<td></td>
<td>Server Side Architecture</td>
</tr>
</tbody>
</table>
## LIST OF FIGURES AND TABLES

<table>
<thead>
<tr>
<th>Figure/Chart/Entity</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ETD Process Model</td>
<td>29</td>
</tr>
<tr>
<td>Sample Requirements Evolution Across Iterations</td>
<td>30</td>
</tr>
<tr>
<td>ETD Architecture Diagram</td>
<td>55</td>
</tr>
<tr>
<td>Dialog Box Functionality</td>
<td>58</td>
</tr>
<tr>
<td>Ajax Submit</td>
<td>58</td>
</tr>
<tr>
<td>Ajax Return and UI Update</td>
<td>59</td>
</tr>
<tr>
<td>Hiding Content with jQuery</td>
<td>59</td>
</tr>
<tr>
<td>Planning Form – State Chart Diagram</td>
<td>67</td>
</tr>
<tr>
<td>Preliminary Draft Approval – State Chart Diagram</td>
<td>67</td>
</tr>
<tr>
<td>Individual Drafts – State Chart Diagram</td>
<td>68</td>
</tr>
<tr>
<td>Final Approval Form – State Chart Diagram</td>
<td>68</td>
</tr>
<tr>
<td>Entity Relationship Diagram – Part I</td>
<td>71</td>
</tr>
<tr>
<td>Entity Relationship Diagram – Part II</td>
<td>72</td>
</tr>
<tr>
<td>Planning Form Presentation Model</td>
<td>80</td>
</tr>
<tr>
<td>Planning Form Activity Diagram – Overall Summary</td>
<td>82</td>
</tr>
<tr>
<td>Activity Diagram – Student – Submit or Modify Planning Form</td>
<td>83</td>
</tr>
<tr>
<td>Activity Diagram – Committee Members and Graduate Coordinator</td>
<td>83</td>
</tr>
</tbody>
</table>
ABSTRACT

THE DEVELOPMENT OF AN ELECTRONIC THESIS AND DISSERTATION SYSTEM FOR CALIFORNIA STATE UNIVERSITY, NORTHRIDGE

By

Katherine Vigna

Master of Science in Computer Science

The CSUN Electronic Thesis and Dissertation System (ETD) is a web application designed to replace CSUN’s paper-based thesis approval workflow and facilitate the digital archiving of theses. This case study presents a description and analysis of the ETD design and implementation process at CSUN. By doing so, it seeks to extend the existing literature on ETD systems – which focuses largely on the process of policy and technology adoption – to address some of the software engineering and development challenges involved in technology creation within a public university setting. Through this analysis, the case study finds that the cultural and organizational issues involved in technology adoption at a university significantly impact how software should be engineered, designed, and implemented. Based on the experiences of CSUN’s ETD development team, a successful software implementation must incorporate flexibility into its code and architecture to meet evolving workflow needs; emphasize usability to ensure user acceptance; and follow an engineering process focused on refining requirements and being responsive to stakeholder needs.
INTRODUCTION

Existing Literature on Electronic Thesis and Dissertation Systems

Since the early 1990s, there has been a growing movement among universities to store student theses and dissertations digitally in institutional repositories (IRs). As universities have made the shift from print to electronic theses and dissertations (ETDs), they have produced a large body of work detailing the process of change, the challenges involved, strategies employed, and advantages/ disadvantages of storing student research online.

With the shift to ETDs at these universities originating in the library, this literature focuses exclusively on the workflow and issues surrounding thesis storage in an institutional repository. The ETD teams described in the literature made the explicit decision not to change any of the faculty or Graduate Studies workflow with regard to thesis approval. Most universities echoed the sentiment of researchers at Cranfield University in the UK: “The current thesis submission workflow would not change until right at the end of the process when students would be mandated to submit an electronic copy of their amended and corrected thesis” [1].

As these universities limited the scope of their ETD systems to final thesis submission, the technical issues they encountered were limited mainly to the selection of an IR and handling the transfer of data between the IR and the main library catalogue. These “technical and archival issues were the most easily addressed,” and no universities reported any major technical issues with the systems [2].

The universities in the literature did, however, experience significant cultural challenges in going digital. Common issues encountered were faculty concerns over
copyright, prior publication, plagiarism, and the publication of sensitive material [3]. A case study on Aberystwyth University describes the approach taken by most universities to build support. Universities would “examine the concerns that were being raised, to dispel those that were based on misconceptions and to establish robust procedures for managing those which remained” [4]. In addition to this emphasis on strong communication and policy development, successful ETD implementers also focused largely on providing support for students through the development of ETD workshops.

The current literature, therefore, presents many useful political and cultural lessons learned for universities looking to switch from storing bound, paper theses on library shelves to storing digital thesis files in an institutional repository. However, because the scope of existing ETD projects is limited to library storage, and most libraries elected to use existing software applications for their IRs, the discussion of development and software engineering issues in the literature is minimal. The discussion is largely one of policy and technology adoption, rather than policy, culture, and technology creation.

*The CSUN ETD Project: Engineering a New ETD Workflow System*

In contrast to the project scope at other universities, California State University Northridge (CSUN) initiated an ETD project designed not only to store theses in an IR, but to replace the entire Graduate Studies workflow as well. To meet the latter goal, the CSUN ETD project undertook the development of a complete thesis approval web application from scratch, and its integration with an out-of-the-box digital repository.

As the CSUN ETD Project encompasses the scope of the ETD efforts at other universities (making the switch to a digital repository), the challenges and strategies
described in the literature on ETDs are directly relevant to the ETD efforts at CSUN. However, these existing systems do not change their universities’ thesis approval workflow, and were undertaken with little effort in terms of software development. Therefore, the CSUN ETD project encountered cultural and technical challenges that the existing literature does not address. These challenges, of course, included standard software development issues such as eliciting requirements and product implementation. More interestingly, however, several unique software engineering issues arose specific to developing software within the organizational structure of a public university. The cultural challenges outlined in the existing literature on ETD adoption, it turns out, extend their reach beyond policy decisions to impact the processes of software development and implementation.

In working through these technical and cultural challenges to develop ETD, one lesson continually reemerged. *It is impossible to separate workflow and cultural issues from development and design issues*. The technical implementation of a system is only as valuable as its ability to meet the needs of its users, fit within its overall business context, and address the real requirements of the business processes which it intends to replace or enhance. As such, successful software design and implementation must incorporate flexibility into its code and architecture to meet evolving workflow needs; it must emphasize usability to ensure the “acceptability” of the product; and its development must follow an engineering process that is responsive to stakeholders. These principles are particularly salient in the public university environment, where the organizational structure places a high priority on shared governance and user buy-in for technology adoption.
The following case study presents an analysis of the ways in which the CSUN ETD project was successful in adhering to the above principles, as well as a discussion of several lessons learned that could be applied to future software engineering efforts in a campus environment. Because an understanding of the business workflow is essential to understanding the design and implementation decisions that the ETD development team made, the case study begins with a discussion of the business context of the ETD project, as well as a description of the existing thesis workflow. From there, it goes on to describe the electronic workflow and detail some of the design and implementation decisions made in developing ETD, particularly as the design decisions relate to the above principles. Finally, the case study discusses several lessons learned with regard to the process of developing ETD—from requirements elicitation to application deployment—specific to both web application development and development within a public university setting.
HISTORY AND BUSINESS CONTEXT OF THE CSUN ETD SYSTEM

The ETD project at CSUN began in the fall of 2008 as a collaborative effort between several administrative departments. On the one hand, the Graduate Studies Office was looking to improve their efficiency and eliminate the unnecessary work involved in dealing with paper theses. Aware that other universities were making the switch to ETDs, the Graduate Studies Office was eager to join the ETD movement and eliminate what they saw as an outdated and wasteful process. Particularly, Graduate Studies was interested in eliminating the influx of bound theses to their office every semester, and the need to initiate the distribution of these copies across campus.

At the same time, CSUN’s Provost was conducting extensive analysis on the cost of a student’s education at CSUN. Particularly, he was focusing on the “unspoken costs” related to students that came in the form of wasted time, extra fees, etc. One area that the Provost identified was the thesis approval and binding process. The cost of binding theses, along with the time it takes to complete paperwork and obtain signatures, could be eliminated through the use of an ETD system. Here, the idea of replacing the entire thesis approval workflow, and not just the thesis printing and binding processes, began to take shape.

This idea was supported by the department of Academic Resources and Planning, and work that the department was doing in coordination with the CSUN Sustainability Institute. The goal of eliminating wasted paper and the unnecessary fuel consumption from student trips to campus solely for paperwork completion supported the idea of building an entire electronic thesis approval workflow.
While the conceptual foundation for ETD was beginning to form, the CSUN Library was moving forward with the process of implementing an institutional repository for campus. The Council of Library Directors, a group of California State University (CSU) deans and directors, was leading a CSU-wide effort to adopt and implement dSpace (an open source application for institutional repositories developed at MIT) as the framework for CSU’s institutional repository. The first use of dSpace at CSUN was to store learning objects. However, dSpace would also serve perfectly as a repository for ETDs. Key staff members at the library supported the expansion of the repository for this purpose and were eager to move forward with the adoption of ETDs.

With the support of all these departments, the CSUN ETD project was born. The impetus behind the project was to reduce costs to students; improve the efficiency in campus business processes; reduce the workload for the Graduate Studies Office; and develop the ETD catalog in dSpace as a campus asset. To meet these goals, the scope of the ETD system was to join the current Graduate Studies thesis approval workflow with CSUN’s institutional repository into one seamless electronic system [5].
CURRENT CSUN THESIS WORKFLOW

The current, paper-based thesis workflow at CSUN consists of three main stages – thesis/ project approval, thesis/ project binding, and thesis/ project storage. The Graduate Studies, Research & International Programs department (GRIP) is primarily responsible for the approval and binding workflow, while the Oviatt Library is primarily responsible for thesis and graduate project storage.

Thesis/ Project Approval – Planning Form

As the first step in thesis/ project approval, students complete the “Thesis/ Graduate Project Planning Form.” This form contains basic student information (such as name, major, etc.), the working title of a student’s thesis/ project, information about research subjects, and a listing of a student’s thesis/ project committee. The form also contains a section for changing the thesis/ project title or changing its committee members.

The committee consists of at least three people – one committee chair and at least two additional committee members. The chair and at least one of the committee members must be full-time CSUN faculty. The other committee member(s) may be an off-campus expert in the student’s field of study, or a part-time faculty member. If a student is working with an off-campus committee member, the student must attach that committee member’s “curriculum vitae” to the Planning Form. If a student is working with more than three committee members (including the chair), s/he must attach a page to the Planning Form with the additional committee information.

Once a student completes the “Thesis/ Graduate Project Planning Form” s/he brings the form to each committee member to sign. Upon obtaining all committee
signatures, the student brings the form to the Graduate Coordinator of the student’s academic plan for signing. Finally, once all signatures are received, the student submits the form to the Graduate Studies Office, where a copy of the form is retained in the student’s file.

**Thesis/ Project Approval – Content**

Students work directly with their committee, particularly their committee chair, to obtain approval of their thesis/ project content. This process consists, at a minimum, of submitting two drafts. An initial draft goes to the committee chair for feedback. The student then incorporates this feedback into another draft that goes to the entire committee for approval. Once the committee approves all draft modifications and the draft has been approved for formatting (see below), committee members sign a printed signature page, which becomes the first page of the official thesis or graduate project.

**Thesis/ Project Approval – Formatting**

As a student’s thesis or graduate project is “a visible and permanent measure of the quality of scholarship expected of a graduate student[,] . . . the Graduate Studies Committee has adopted a [set of] criteria that establishes uniformity in physical format for all theses, graduate projects and artistic abstracts” [6]. To ensure that students adhere to these formatting requirements, students must obtain formatting approval from the Graduate Studies Office. The process of formatting validation is as follows.

First, students contact Graduate Studies to schedule a preliminary review appointment with a Graduate Evaluator. When students attend the preliminary appointment, they bring a printed copy of their thesis or graduate project. During the
appointment, the Graduate Evaluator reviews the printed copy for adherence to all formatting requirements, and completes the “Thesis/ Project Preliminary Checklist.”

The “Thesis/ Project Preliminary Checklist” contains space to indicate required formatting changes, by category. For example, there are sections to indicate formatting changes required for the Title Page, the Signature Page, the Text, etc. The form also contains information about the number of printed final copies that are required for final submission, as well as the paper type required for the final copies and information about how to submit the final copies.

If a student’s thesis or graduate project requires minimal or no formatting changes, then the student schedules a final appointment with Graduate Studies for thesis or project submission. If, however, the student’s document requires significant formatting changes, s/he must schedule a “Post Preliminary Appointment” for formatting approval. The student makes the indicated changes on the “Thesis/ Project Preliminary Checklist” and brings a printed copy of the changed draft to the “Post Preliminary Appointment.” Once the Graduate Evaluator has validated the thesis for project formatting, the student can go on to schedule the final appointment for thesis or project submission.

**Thesis/ Project Approval – Submission**

Once a student’s committee has approved the thesis/project content, and the Graduate Studies Office has approved the thesis/project format, the student is ready to submit his/her thesis for binding and storage.

Based on instructions on the “Thesis/ Project Preliminary Checklist,” the student prints the appropriate number of final thesis/ project copies on the appropriate type of
paper. The student must submit two printed copies for the library, at least one printed copy for the student’s academic department (some departments may require more), and any number of optional copies for the student’s own use. The original copy for the library is printed on 100% white cotton paper, and must contain the original ink signature page. The other copies are printed on 25% white cotton paper, and contain copies of the original signature page. The student places each copy in an individual, labeled envelope for delivery to the Graduate Studies Office.

Before attending the final appointment for submission of the thesis/ project with the student’s Graduate Evaluator, the student completes the first portion of the “Thesis and Project Approval Form.” This portion of the form contains basic student information and thesis/ project information required for binding. The latter includes the number of copies to be bound, and the title and year of the work as it is to appear on each bound copy.

At the final appointment with Graduate Studies, the Graduate Evaluator reviews the “Thesis and Project Approval Form,” signs the form, and indicates the total fees required for binding.

*Thesis/ Project Binding – Payment*

Upon completion of the final appointment with Graduate Studies, the student takes a copy of the “Thesis and Project Approval Form” to Cashier Services. Here, the student pays the indicated binding fees and obtains a receipt of payment. The student brings this receipt of payment back to the Office of Graduate Studies, indicating that the student’s printed copies can be sent to the bindery.
Thesis/ Project Binding – Binding and Distribution

Under the current workflow, Graduate Studies stores all printed theses submitted during the Final Approval process in their office. At the end of each semester, Graduate Evaluators package the theses and send them to the bindery. Once bound, the physical theses return to the Graduate Studies Office for distribution. Graduate Evaluators sort through the bound theses, separating the copies into stacks for distribution to the library, departments, and students. Evaluators then call each thesis recipient, indicating that the theses are available for pick up in the office. Often, Evaluators must make several calls to the thesis recipient before the recipient actually collects the thesis.

In addition to handling the collection and distribution of paper theses, the Graduate Studies office is also responsible for maintaining a contract with the bindery.

Thesis Storage – Library Workflow

The library currently receives two bound copies of each thesis. The first one goes to the library archives, where it receives a barcode and location code. The other copy is stored in the library stacks. Each thesis is reviewed by a bibliographer in the subject area of the assignment (for example, a health sciences librarian) and is examined for topical subject headings (those specific by the Library of Congress). If the bibliographer believes that there are any applicable headings, s/he denotes this by physically inserting a slip of paper in the thesis.

From here, the thesis goes through cataloguing. All theses receive descriptive metadata, and those marked by the bibliographer are reviewed for subjects headings. Each thesis is catalogued in the WorldCat OCLC system, and then downloaded from there into the local system.
Once catalogued, the thesis goes through physical processing (is stamped, given a due date slip, security devices, etc.) and shelved.
ETD / DIGITAL WORKFLOW

The Electronic Thesis and Dissertation system aims to streamline, as much as possible, the current paper-based workflow. To do this, it integrates the three main workflow components – thesis/ project approval, thesis/ project submission, and thesis/ project storage – into one seamless student-initiated workflow. This workflow combines the ETD web application with ScholarWorks – CSUN’s digital repository for thesis storage (built on the dSpace platform).

Thesis/ Project Approval – Planning Form

To submit the “Thesis/ Graduate Project Planning Form” using ETD, students login to the system, and complete both a web form with general student information, as well as a web form with the Planning Form content. The web forms contain the same information as the paper form, excluding the portion for changing the title or committee. Students can modify a form in ETD by resubmitting it, rather than appending it.

Once a student submits the online “Thesis/ Graduate Project Planning Form,” the system emails each of the student’s committee members with information about how to login to ETD to view the form. Once logged in, committee members can elect to approve the form, reject the form, or defer decision until the student contacts them. If the committee member is not approving the form, s/he must indicate required changes.

If any committee member defers decision, an email is sent to the student notifying the student to contact the committee member. If any committee member rejects the form, the system sends an email to the student prompting the student to revise and resubmit the form. When the student resubmits the form, the process repeats.
Once all three committee members have logged in and approved the form, the system sends an email to the graduate coordinator with information about how to login and view the Planning Form. Once logged in, the graduate coordinator has the same options for approving, rejecting, or deferring action on the form as the committee members. Just as before, if the graduate coordinator rejects the form, the student is notified to revise and resubmit the form; if the coordinator defers decision, the student is notified to contact the coordinator; if the graduate coordinator approves the form, the system emails the student indicating that the form is complete.

*Thesis/ Project Approval – Content*

The process for approving the student’s thesis or project content is the same in the digital workflow as it is in the paper workflow. Students continue to interact directly with their committee members for content approval.

*Thesis/ Project Approval – Formatting*

When a student’s draft is ready for formatting approval by Graduate Studies, the student logs in to ETD and uploads a pdf of the draft. Upon upload, the file immediately becomes accessible by the student’s Graduate Evaluator through ETD.

To review the file, the Graduate Evaluator logs in to ETD and downloads the student draft. The evaluator then completes the web version of the “Thesis/ Project Preliminary Checklist,” indicating any formatting changes that need to be made to the document. This form is equivalent to the paper-based form, except that it does not contain information about printing the draft.

If the Graduate Evaluator feels the required formatting changes are minimal, s/he selects the approve option on the form. This sends the required changes to the student.
The student does not need to schedule an appointment with the evaluator. On the other hand, if the Graduate Evaluator feels that significant changes are required, the evaluator may require the student to either submit a modified file to ETD (in which case the process repeats), or schedule an appointment with the evaluator.

If an appointment is required, the student will not be unable to submit a new file through ETD until meeting with the evaluator in person. After completing the scheduled appointment, the evaluator indicates in ETD that the appointment is complete, and the system makes the list of required changes available to the student. The student can then submit a revised file, and begin the formatting approval process again.

_Thesis/ Project Approval – Submission_

Once a student’s committee has approved the thesis/project content, and the Graduate Studies Office has approved the thesis/project format, the student is ready to complete the “Final Approval” form by logging in to ETD. The form contains a section to upload the final draft (in pdf format), and any additional supporting files, such as audio or video files, images, application code, etc.

In addition to the file upload sections, the form also contains several sections of library metadata. This metadata will be associated with the thesis files in the campus repository, and includes information such as the author, keywords, bibliography information, etc. Finally, the student must acknowledge acceptance of a license agreement (so that the files can be stored in the campus repository), and is given the option of embargoing his/ her work if such a need exists.

Once the student has completed and submitted the form, ETD notifies all committee members that the form is ready for review. Committee members can login to
ETD, download all thesis/project files, and review the metadata, license agreement, and embargo request. Committee members are given two of the options they were given for the Planning Form—approve or reject the thesis.

If any committee member requires changes to the form, ETD sends an email to the student, telling them to login and make the appropriate changes. When the student resubmits the form, ETD notifies all committee members of the resubmission.

Once all committee members have approved the form, the form is made available through ETD to the student’s Graduate Evaluator. When the Graduate Evaluator confirms that the student has met all of his/her graduation requirements, the evaluator approves the “Final Approval Form” for the student. At this point, ETD submits all of the student’s thesis/project files, along with the corresponding metadata, to the online repository for storage.

*Thesis/Project Binding – Payment and The Bindery*

Under the ETD workflow, final theses and projects are stored digitally in the campus repository. They are neither printed nor bound. As such, there is no bindery payment required, and no workflow for contracting with the bindery or handling bound copies.

*Thesis Storage*

Under ETD, there are three main changes to the library storage workflow. First, since there are no bound copies, theses no longer go through physical processing. Second, under ETD, the descriptive metadata for the record is student-entered. The library cataloguers still have to review and clean this data, but most of the metadata content will already have been generated by students or the ETD System. Finally,
because there is no physical thesis, there is no way for the bibliographer to indicate that a specific thesis has potential subject headings (no such tags exist in the electronic catalogue). As a result, all records have to undergo review and assignment of subject headings.
CSUN’s ETD system offers its users several benefits through the replacement of the current thesis workflow process and the use of a digital repository for thesis storage.

Benefits of Making the Graduate Studies Workflow Electronic

For students, the current Graduate Studies workflow consists of the completion of paper-based forms for thesis approval, attendance of at least one formatting appointment with a Graduate Evaluator, and at least one appointment for the final submission of a printed thesis for binding. By replacing this current workflow, ETD offers students several benefits. First, the ability to complete forms online translates to significant time savings for students. Students can complete thesis forms at their convenience, and do not need to physically meet with their committee members to obtain approval signatures. For many graduate students – especially those who work full time and have small course loads – this often means saving the time associated with a trip to campus. With the average CSUN commuter travelling 14.5 miles to campus, this time can be significant [7]. While perhaps less significant, avoiding such a commute to campus also means avoiding unnecessary transportation costs.

In addition to saving time in obtaining signatures, ETD creates the potential for students to avoid both preliminary approval appointments and the final thesis submission appointment with Graduate Evaluators. If GRIP approves the files a student submits through ETD, the student is not required to schedule or attend an appointment on campus. Again, this translates to time and cost savings for students.

The biggest cost reduction for students through ETD results from the elimination of thesis binding. Because ETD stores files electronically, there is no longer a need for
students to print drafts or to print final thesis copies on expensive 100% cotton or cotton blend paper. Nor is there a need for students to pay for the binding of multiple thesis copies. The current cost of binding a thesis ranges anywhere from $17 to $22 and students currently bind a minimum of 3 thesis copies (2 for the library and 1 for their department). ETD completely eliminates this cost for students.

The Graduate Studies Office also receives significant time savings through the use of ETD. Graduate Evaluators can review drafts and final theses as students submit them online. In this way, evaluators need to schedule fewer appointments with students, and have more flexibility and control over their time. Currently, appointments consume approximately the last six weeks of each semester. Of course, Graduate Evaluators still have to spend time reviewing documents, but they can do so more efficiently and without unnecessary appointments.

By eliminating the binding process, ETD further benefits Graduate Studies. ETD reduces the scope of the Graduate Studies workload, as the office no longer has to handle bindery contracts or contact students and departments upon the office’s receipt of the bound theses. In addition, the office receives space savings and an improvement of their work environment, as they no longer store cases of bound copies in their office every semester.

Finally, ETD provides faculty and graduate coordinators with benefits, as well. ETD reduces the level of work disruption that faculty experience with form signature requests. Forms can now be signed online, at the convenience of the signer. Likewise, ETD provides both groups with an overview section with which they can view student progress through the thesis process. For faculty, this overview shows all of the students
under their advisement. For graduate coordinators, this overview shows all students in their department or academic plan. This added feature improves the ability of faculty and graduate coordinators to advise students.

In addition to providing benefits to individual users, eliminating the binding process provides system-wide benefits. Cash Services benefits from a reduction in workload as the office no longer has to deal with student bindery payments. The university and community benefit as well, as the elimination of paper forms, and bound/printed copies reduces paper waste on campus.

*Benefits of Thesis Storage in ScholarWorks*

In addition to saving students money through the elimination of binding fees, publication of student work in ScholarWorks offers many benefits to students and the CSUN community as a whole. From a practical standpoint, the use of a digital repository provides students with persistent, permanent access to their work. Within the ETD workflow, students receive a permanent handle (or link) to their thesis once it is deposited in ScholarWorks. Anyone can use this handle to access the student work, and the handle is guaranteed to permanently point to the work. The student will never need to update the handle or update their thesis files.

Once in an online repository, student work becomes searchable not only in library catalogues but through search engines online, as well. As a result, experience at universities with ETDs has shown that repository publication increases the breadth and speed with which student work is disseminated. For example, experience at Virginia Tech – a leader in the ETD movement – shows that eTheses are 100 times more likely to be circulated than their paper equivalent. At West Virginia University, “access to theses
increased a staggering 145,000% -- from 813 issues in 1998/99 to 1,181,111 accesses in 2003 [1]. A University of Glasgow case study shows similar results. Here, the most downloaded e-Thesis was downloaded 37,000 times as of 2009. At Brunel University Library, downloads of theses in March 2010 were up to 764,444. From an anecdotal perspective, some universities point to cases in which publication in their ETD led to book contracts and teaching positions for students [4].

This improved dissemination of student work offers universities several benefits. First, through the increased number of citations and awareness of student work, the profile of the author and the campus are raised. In other words, the collection of academic work in the repository becomes a visible university asset. Students and faculty, in turn, benefit as researchers from ETDs at other universities. ETDs “ensure that a huge, untapped information resource is made available more widely” [1]. Researchers have access to a greater range of information, resulting in “less duplication of research efforts” [3].

ETDs do not just increase the dissemination of student work. ETDs have the potential to change the very nature of student research itself by expanding the methods and modes by which students can express their knowledge and research. Digital repositories can accept work of all file and media types. As such, students publishing their work in an ETD can make use of “new technologies [that] can substantially encourage creativity, experimentation, and exploration and [that] can literally transform the types of contributions to knowledge made in theses and dissertations” [2]. Through the use of digital repositories, students are no longer bound to express their work in a traditional text format. Instead, students are free to “develop new ways of creating access
to their research – they [can] create an electronic work for which there could be no paper equivalent” [1]. At its simplest, this expanded means of expression might take the form of performing arts students submitting video of their actual performance; a computer science student publishing an application; or a music student uploading elements of a composition. Over time, however, as students begin to take advantage of the digital format, theses may become composites of media types that structure information and knowledge in entirely new ways. As such, ETDs offer the potential to improve academic research by expanding the variety of production formats available to students.
DEVELOPING ETD – DECISION CRITERIA AND CHALLENGES

In developing ETD, the ETD team had to balance three sets of decision criteria. First, wherever possible the development team followed best practices for application design and programming. Second, the team made development decisions to address the challenges of web application development. Finally, the unique needs of development within a public university setting served as another essential set of decision criteria. The ETD development team applied these criteria to all development decisions – from process model selection, to code implementation, to application deployment. Often, these three sets of decision criteria complemented one another. Occasionally, however, the criteria conflicted and the development team had to make trade-off decisions with regard to which criteria to prioritize for any given portion of the development process or application design.

Characteristics of Web Application Development

Noting that web development differs significantly from traditional software development, Engels, Lohmann and Wagner identify several unique requirements for a web application development process. Among others, these requirements include the need to handle short development cycles; the ability to handle changing requirements; and the ability to provide releases with fixed deadlines and flexible contents (201-205). The key to all of these characteristics is flexibility and development speed—two criteria which became the foundation for process, design, and implementation decisions for ETD.
Characteristics of the University Environment

In addition to considering the nature of web development in general, all processes and decisions in the development of ETD were largely informed by the characteristics and challenges of developing software within a public university environment.

From an organizational perspective, CSUN has several defining characteristics. First, the university has a wide, distributed organizational tree. Therefore, developing an application like ETD, which spans several administrative and academic departments, involves bridging the political and technical gap between largely independent organizational units. Likewise, spanning such an organizational structure means that the development process for ETD must be responsive to a large number and variety of stakeholders. Not only does the application have several user groups. Each user group has a significant amount of internal diversity. For example, the single user group of “faculty” actually consists of faculty from different departments, with different needs and interests. Faculty in the humanities department cannot necessarily be treated as if they are the same user group as faculty in the biology department. The same applies to students, graduate coordinators, and off-campus committee members.

In addition to the organizational structure, the decision-making process in the university environment offers several challenges for application development. Stakeholders on campus are accustomed to a system of shared governance. This concept of shared governance has come to incorporate “two complementary and sometimes overlapping concepts: giving various groups of people a share in key decision-making
processes, often through elected representation; and allowing certain groups to exercise primary responsibility for specific areas of decision making” [8].

There are two main consequences of this governance structure that impact application development. First, in contrast to a traditional, hierarchical environment where decisions are handed down an organizational tree, stakeholders on campus are accustomed to a relatively large degree of involvement in decision-making. This makes user acceptance more critical for technology adoption in the university environment than it is in a traditional private organization. Therefore, user acceptance was a major priority in all design and deployment decisions for ETD, from its earliest stages of development. The decisions of the ETD team with regard to generating user buy-in can be understood in terms of the User Acceptance Model presented by Venkatesh, et al in “User Acceptance of Information Technology: Toward a Unified View.” According to the model, four constructs influence user acceptance of a technology: performance expectancy, effort expectancy, social influence, and facilitating conditions [9]. When considering user acceptance, the ETD development team sought to minimize effort expectancy, and maximize performance expectancy, the role of social influence, and the existence of facilitating conditions.

The second consequence of shared-governance is that broad participation in decision-making translates to slower policy making processes. As a result, some policies that impact the workflow and code structure of an application cannot be defined fully, prior to implementation. Furthermore, in the case of new policy that needs to be made, the implementation of the application can be critical in helping stakeholders visualize and define the policy in question. (This is similar to the “I’ll Know it When I See it
Phenomenon” in requirements engineering, but is more challenging as it relates to the development of policy outside the application itself.) As a result of this slower policy-making process, implementation decisions need to be made based on “guesses” and assumptions about policy decisions that have yet to be made. Clearly, this presents several challenges for both requirements engineering and implementation.

Finally, as a public university, CSUN has limited resources that can be allocated to application development. Yet at the same time, the university is subject to additional ideals and accountability levels as a public institution. For example, as a community leader in universal design, CSUN has higher standards of accessibility than might be pursued in private institutions.

Application of Decision Criteria

The challenges of developing within a university environment, along with the unique requirements of developing web applications, impacted every facet of development for ETD, including process model selection; implementation decisions; usability decisions; and the deployment process of ETD. Each of the following sections addresses the impact of these challenges on a single component of development, the successful strategies the development team used to address these challenges, and the lessons learned in the process.
ETD DEVELOPMENT – PROCESS MODEL

Based on the unique characteristics of web application development, Paul Grunbacher concludes that the best process model for requirements elicitation and development is an iterative definition of requirements. In such a model, “at project inception, key requirements are typically defined on a higher level of abstraction.” Then, through the creation and evaluation of prototypes, developers can “concurrently refine both requirements and the system architecture in an iterative manner with a continually increasing level of detail” until the final application is developed [10].

The process of requirements elicitation and application development for ETD followed such an iterative process model. Early in the process, business context analysis, document and workflow analysis, and consultation with a core group of stakeholders served as the basis for defining higher level-requirements. These requirements, in turn, informed the design of an initial prototype. Once the core group of stakeholders interacted with this prototype, requirements were refined, and a new prototype developed. This cycle continued, and with each new prototype, an increasing number of stakeholders were folded into the process to expand the level of feedback and further refine requirements.

In addition to addressing the characteristics of web application development, such an iterative process model was well-suited to address the unique challenges of developing software within a university environment. First, it addressed the challenge of designing an application for a large number of stakeholders working across several horizontal departments. By gradually incorporating an increasing number of stakeholders with each
iteration, the process model allowed the development team to incorporate feedback and requirements from a large number and variety of stakeholders in a controlled fashion.

Second, the process model was well-suited for a context in which decisions are made through shared governance. It emphasizes user acceptance by starting with an analysis of the business context as a foundation for requirements definition; by incorporating user feedback on each iteration; and by allowing gradual deployment across departments to build support for, and comfort with, the general ETD system. Finally, the iterative development process allowed the development team to move forward with development while policy decisions were being made, helping to mitigate the risk of wasted development time.

The following figures captures the process model used to develop ETD, as well as the evolution of a specific requirement across development iteration.

ETD Process Model

Requirements Elicitation Through Iterative Prototype Development
Sample Requirements Evolution Across Iterations

<table>
<thead>
<tr>
<th>Early Prototype Requirement</th>
<th>Mid-Development Requirement</th>
<th>Late Development Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Workflow Requirement: Once all committee members approve the Final Approval Form, the form goes to GRIP, who can reject or approve the form. UI Requirement: GRIP has one tab for pending Final Approval Forms.</td>
<td>Workflow Change: In addition to approving or rejecting a Final Approval Form, GRIP can place the form on “hold” for students who have not been cleared for graduation (for non-thesis related reasons.) UI Requirement: GRIP has a Final Approval tab and a Held tab.</td>
<td>Workflow Change: When GRIP rejects a Final Approval Form, student no longer need to resubmit the form to committee members. This allows GRIP to reject a thesis for formatting reasons.</td>
</tr>
</tbody>
</table>

The following sections describe the individual components and sub-processes of this development model in more detail, listing ways in which the goals of the process model were successfully met, as well as lessons learned and areas for future improvement in the execution of such a process model. The sections discuss the sub-processes of requirements engineering, implementation, and deployment.
REQUIREMENTS ENGINEERING

The process of requirements engineering for ETD consisted of defining initial requirements, delimiting project scope, identifying and involving stakeholders, and negotiating expectations. Each one of these activities was successful to the extent that they took into account the characteristics of the university environment in which ETD was to be deployed. In other words, when requirements engineering took into account the importance of user acceptance, the value of user feedback, and the broad, decentralized structure of the university, the ETD development team was able to accurately capture and prioritize requirements. When the development team lost sight of these characteristics, however, requirements elicitation presented several difficulties and several lessons were learned.

**Defining Initial Requirements: Analyzing the Business Context**

In defining initial requirements for any web application, Grunbacher emphasizes the importance of understanding the context in which a system will operate. The act of “analyzing and describing existing business processes and the rationale of the system to be developed” greatly improves the process of requirements elicitation through “identifying success-critical stakeholders, familiarizing [oneself with the system’s] intended use, and analyzing the [system’s] constraints” [10]. In the case of ETD, the business context can be understood in two parts. First, the business context consists of the actual business workflow that the application is replacing. Second, the business context consists of the organizational environment in which users will interact with the system.
Workflow Capture

As ETD directly replaces an existing paper-based system, requirements elicitation naturally began with an emphasis on understanding the first component of the business context – the business workflow. Early stages of requirements elicitation concentrated on an analysis of existing thesis documentation and paperwork to develop a rough application flow chart and outline of general use cases. Analyzed documentation at this stage included existing Graduate Studies forms (Thesis/Project Planning Form, Preliminary Draft Approval Form, and Final Thesis Submission Form), the online thesis process description on the Graduate Studies website, and an examination of ScholarWorks.

While this document-based approach was successful in defining the general framework for ETD, it neglected to adequately involve stakeholder knowledge about the details of the thesis workflow, and failed to consider the ways in which the thesis workflow might vary across a wide organizational tree with largely independent departments. As a result, initial attempts to define workflow failed to capture use case exceptions that would greatly impact code development.

In the case of ETD, which was replacing an existing paper-based system, capturing these use case exceptions was particularly important to development success and efficiency. There is a certain amount of flexibility inherent in paper-based forms. For example, it is not uncommon with thesis paperwork for a faculty member, department chair, or graduate coordinator to sign a form on behalf of another faculty member. This is done in cases of illness, when a faculty member is out of town, or when
other similar needs arise. In these cases, the paper forms are simply amended by adding hand-written comments to the form itself. To replicate this ability in the electronic system, however, an “on-behalf-of signature” requirement would need to be formally defined and implemented, such that one user would gain permission to access and view another user’s pending forms; someone could sign off on-behalf-of another user; and the system could store and note such a signature detail appropriately in the database.

Similarly, in the current system, if a student has more than three thesis committee members, information for additional committee members is simply attached to the Thesis Topic Planning Form. There is no official form for this addition, but paper-based forms are flexible enough to accommodate it. Again, to programmatically replicate the capacity for unlimited thesis committee members, this exception would need to be formally defined as a requirement. The user interface on the Thesis Topic Planning Form would have to allow the addition of $3 + n$ committee members; the code for inserting into and retrieving committee members from the database, as well as checking for the status of committee member signatures, would all need to include the flexibility of a variable committee count.

Unfortunately, the requirements elicitation process for ETD failed to capture most of such workflow exceptions within a productive time frame. In the case of the $n$-committee member count requirement, the exception was noted late in the development process. Because the requirement had to be implemented, the result was a significant amount of code restructuring and rewriting. In the case of the on-behalf-of signature option, the exception was discovered too late in the process to be implemented.
These examples highlight the importance of discovering workflow exceptions early in the requirements engineering process. The development team failed to capture these exceptions, however, for three reasons. First, the team assumed that core stakeholders would raise issues of important exceptions without being prompted. This, of course, is not the case. Exceptions are so easy to deal with in a paper-based system, and so much a normal part of the user’s routine, that they do not stand out as “separate workflow.” Furthermore, users cannot be expected to know the programming implications of workflow differences. When eliciting requirements, therefore, it is essential that the development team ask explicitly about workflow exceptions.

The second reason that the development team failed to adequately capture workflow exceptions is that the team assumed most exceptions to the standard workflow were rare and not worth the implementation time required to accommodate them. For example, one team member early in the process indicated that handling off-campus committee members within ETD was unnecessary because the involvement of off-campus committee members was such a rare use case. As it turns out, however, according to Graduate Studies estimates, off-campus committee members serve on approximately 20% of all thesis committees. The development team member simply assumed that the use case was rare because of its rarity within certain academic departments. As a result of these misassumptions, the development team failed to ask questions about exceptions that occur, the frequency of exceptions, and the best way to programmatically handle exceptions. Had the development team focused less on document-based analysis and more on stakeholder conversation and feedback, some of these exceptions would likely have been captured earlier in the process.
Organizational Setting and Impact on Users

In addition to understanding the general business workflow, business context analysis also includes understanding the driving motivation for converting to an electronic system as well as how such a change will impact university staff, faculty, and students. With regard to this portion of business context analysis, the process of requirements elicitation was successful to the degree that it took into account this impact and focused on user acceptance. Without such a focus on understanding of how ETD could impact its users, an analysis of the business workflow would likely fall short of defining equally critical (but less obvious) requirements. For example, early in the requirements elicitation process, the development team met with Graduate Studies to discuss the Graduate Evaluators’ concerns and expectations regarding the application. One issue that immediately arose was the ability to control workload fluctuation.

Under the current paper-based system, most students wait until the submission deadline approaches to schedule their preliminary draft appointments, and even to submit Thesis / Topic Planning Forms. As a result, Graduate Studies experiences a huge spike in workload toward the end of the term. Currently, GRIP controls this workload through their ability to schedule appointments. Students may wait until the last minute to schedule an appointment, but the allotment of appointments ensures a certain distribution of workload. Because ETD largely eliminates the need for face-to-face preliminary appointments, the concern naturally arose as to how Graduate Evaluators would be able to control the end-of-semester spike in workload. If students waited until the submission deadline to submit drafts, Graduate Evaluators would be left with months worth of document reviews to complete in a matter of days.
Once this concern arose, it was easy to find solutions – such as shifting the submission deadline to earlier in the term, and refining the requirements to include a “deadline notification” system whereby lagging students would receive email notifications that deadlines are approaching, etc. In this way, requirements changed to ensure that the application not only duplicated the current business workflow, but could also address the needs of the greater business context. Without this change, any time saving benefit to GRIP through the use of electronic forms would be meaningless, as GRIP could potentially be left with an impossible workload at the end of each semester.

In addition to helping identify functional requirements, an understanding of the greater business context helped define non-functional requirements that are essential for user acceptance. For example, by moving from storing bound copies in the library to publishing theses in the digital repository, several copyright and publication issues (both real and perceived) arose. Consulting with stakeholders from the library, as well as researching the impact of such issues on other universities, was essential in defining non-functional requirements for dealing with copyright. The perspective and knowledge of library staff was critical in understanding the greater business context of online publication; how it would impact student publication rights; university accountability for respecting third-party copyrighted material; embargo options; and how faculty and students might respond to these issues. Without this understanding, the implementation of ETD could meet all technical specifications but fall short of meeting the needs of university members and attaining user acceptance. Thus, an analysis of the greater business context helped not only define functional requirements, but also meet the non-
functional challenges involved in deploying a new application within the public university setting.

*Iteratively Redefining Requirements and Controlling Scope Creep*

After the initial process of requirements elicitation, the ETD development process entered several iterations of requirements refinement and prototype development. One goal of using such an iterative development model was to “react flexibly as the project evolved . . [by] selecting high-value requirements that need to be implemented first” in the face of firm deadlines [10]. While the process model provided the flexibility to address user and stakeholder needs as they arose, it also presented the challenge of constantly having to reorganize, renegotiate, and reprioritize requirements. Requirements specification was a continual process of controlling project scope by deciding which new features would increase user acceptance and benefit the most users for the implementation time required. Because this process model included constant requirements assessment, it was critical to develop a schema for incorporating new requirements and controlling scope creep. Such a schema would allow for more efficient and methodical requirements prioritization. More importantly, however, it would help effectively communicate to stakeholders the rationale behind requirements prioritization and, if defined correctly, increase the likelihood of user acceptance.

With these goal in mind, the development team began to classify requirements into three main categories. The first category consisted of requirements that were an exact digital translation of current workflow requirements. The functional requirement of being able to submit or modify a particular form fits within this category. The second category defined requirements that are not an exact duplication of the current workflow,
but that are required to achieve the same basic goals and user satisfaction level as the current paper-based system. For example, requirements involving the digital repository fit into this category. The current system does not include any digital repository storage, however, the general goal of thesis storage in the library is currently met through the binding process. Integrating the digital repository into the ETD workflow is required to achieve the same goal, but introduces new requirements (such as embargo handling and creating a process for library metadata definition and approval). The third category consists of added features and enhancements to the workflow that are possible through advantages in technology. For example, implementing form submission lockout upon missed deadlines, or creating a message board that stores committee member feedback about thesis content would both fit into this category.

Using this schema, the development team decided to prioritize requirements that fell into the first two categories, and those that were very easily implemented in the third. The reasoning behind this prioritization scheme is simple. The original impetus behind the development of ETD was to replace the existing paper-based workflow in a way that would create time and cost savings for campus members. Adding new functional requirements would move the domain of ETD outside of these original goals and into the domain of workflow modifications and campus feature additions. These feature additions raise significant opportunity cost questions that must be made outside the realm of ETD, with a greater perspective of campus needs. It would be easy to say, from the perspective of an ETD developer, that implementing a message board for communication between students and faculty within ETD would create value-add for users. However, it is impossible to say from the perspective of an ETD developer, whether the hours required
to develop this feature would add more value to campus than the pursuit of an entirely
different application for a different workflow.

In addition to reasons of opportunity cost, requirements in the third category were
rarely prioritized because there is very little consensus between stakeholders regarding
the desirability of new features. To many stakeholders, the addition or modification of
significant workflow features would mean an increase in effort expectancy, and possibly
a decrease in performance expectancy. In other words, to many stakeholders significant
workflow changes would create a barrier to user acceptance. For example, some faculty
strongly advocated for the incorporation of thesis content feedback and approval into
ETD. Others greatly opposed this feature, as they felt it would replace too much of the
face-to-face advisement they see as critical to the research process and make the
evaluation of student work more difficult. Depending on the individual users, in other
words, this feature would represent either an increase or decrease in both performance
expectancy and effort expectancy. However, not implementing this or other new features
would not be perceived as decreasing performance expectancy or increasing effort
expectancy, as it maintained the existing workflow.

By developing a prioritization scheme, it became easy to avoid conflicts between
user perspectives, to negotiate requirements effectively within the context of ETD’s
iterative development process, and to communicate requirements decisions to a large and
varied group of stakeholders. Fortunately, the development team always had the existing
process to fall back on as a guideline for selecting requirements and articulating those
selections to campus members. Without such a fallback it would have been extremely
difficult to prioritize requirements from the broad groups of stakeholders that exist on campus.

Reverse Scope Creep – Losing Sight of User Acceptance

While the analysis of ETD’s business context and the definition of a schema for controlling scope were successful largely because of their focus on user acceptance, challenges to development emerged when the ETD team lost such focus. This happened several times throughout development, as an interesting pattern emerged in which the development team and core stakeholders were sometimes willing to drop requirements that from a political point of view, may have been worth keeping. For example, because of the shared governance structure on campus, achieving user buy-in was particularly important in facilitating the adoption of ETD. With this in mind, the development team initially included several “transitional” requirements that were not functionally necessary. The idea behind these transitional requirements was to make campus members comfortable with the switch from a paper-based to a digital system. For example, early designs included a staged workflow in which initial versions of ETD would offer students the option of having their thesis printed and bound. Under this workflow, ETD would submit the thesis files to the bindery as well as the library. Ultimately over the course of the application design, the development team dropped this requirement.

On the one hand, dropping a transitional requirement can be seen as a positive indication of growing confidence in the system. In the case of the staged workflow, the requirement really should have been dropped. Even though several participants in the pilot with the Ed.D. department still expressed the desire to have their thesis bound, the feature simply was not worth the development time it would have taken to implement.
Very few users turned out to need this transitional requirement, and dropping the requirement prevented the waste of developing a soon to be phased out feature.

There were, however, some cases in which dropping transitional requirements led to user apprehension or dissatisfaction with the system. Most prevalent among these cases was the way in which the development team and core stakeholders handled issues surrounding thesis publication in the digital repository. In early designs, online publication of student work provided several options. First, there were various levels at which students could restrict access to their work; students could make their work open to the world, restrict access to campus members only, or embargo their work altogether. If students selected the embargo option, they would be given multiple embargo lengths from which to select. Early discussions included embargo lengths of up to five years. Finally, early discussions of publication options included a way for students to specify a creative commons license for their work, indicating the level of copyright sharing they were willing to provide.

As the development process continued, the development team and core stakeholders became increasingly willing to reduce the number of these publication options. In some cases, the ETD team dropped requirements strictly because of technical issues. This was the case in dropping the creative commons license requirement. It simply is not possible to specify a creative commons license for ScholarWorks through the SWORD API that ETD uses to make deposits to the repository.

In other cases, however, the team dropped publication options and transitional requirements that would have been technically feasible to implement. For example, restricted access options were reduced from three (campus, world, and closed), to just
two (world and closed). This decision was based partly on specialized library knowledge and partly on increased comfort and familiarity with ETD and digital repositories. To library staff, for example, it is common knowledge that in the current paper-based system, there is really no such thing as “campus only” access. Theses are available to the general public through interlibrary loan. Therefore, there is no effective difference between world access in an online repository and the availability of bound copies in the current paper-based system. The policies only seem different to those who are not comfortable with the ease of access that digital repositories provide. Furthermore, research done by the ETD team indicates that very few other universities offer the option of campus-only restricted access, and that these more limited policies have not been problematic.

Likewise, very few of the universities currently mandating online publication offer embargos as long as five years. With this awareness of the history of digital repositories, specialized knowledge of real-versus-perceived copyright issues, and a growing comfort with ETD, it made perfect sense for the development team to reduce the maximum embargo length accepted by ETD from five to two years (any longer period of time would require a manual request), and to drop some of the transitional requirements surrounding the repository. After all, dropping these unnecessary options and requirements would reduce the number of features to implement and simplify the system. From the point of view of the development team, there was really no reason to offer anything other than a limited embargo.

As it turns out, many users do not share this point of view. They have not received training on copyright; they have not researched other university policy; and they
are not comfortable with ETD. Far from being part of the digital repository movement, these users are just being introduced to digital repositories through the use of a brand new system. By defining limited embargo and publication options, the development team was now requiring twice as much trust from new users. Not only were users expected to trust that ETD would effectively replace a familiar paper-based workflow. They were also being asked to trust that student publication rights, grants, and research would not be compromised by online publication in a second unfamiliar system – ScholarWorks.

Interestingly, at the beginning of the development process, the development team did not expect users to take this double leap of faith. The team anticipated that copyright and online publication would be a major concern, and built transitional requirements (in the form of more expansive publication options) into the application design so that users could become more comfortable and familiar with digital repositories while using ETD. Yet as the development team and core stakeholders became increasingly comfortable and familiar with ETD, they became comfortable with dropping these transitional requirements. This process of gaining familiarity parallels the process of increasing user interface familiarity that application designers experience. It is commonly noted that developers should not test their own application for usability, because their familiarity with the application leaves them unable to objectively anticipate and recognize usability issues. Similarly, through their increasing comfort and familiarity with ETD, the development team somewhat lost sight of the apprehension that a typical, new user might have about the system. The result is that the team may have too easily and at too early of a stage, dismissed previous, valid concerns by losing sight of the priority of user acceptance.
Involving Stakeholders and Negotiating Expectations

According to Grunbacher, successful web requirements engineering entails involving stakeholders through the “(1) identification of success-effective stakeholders or suitable representatives; (2) understanding the stakeholder’s objectives and expectations; and (3) negotiation of different expectations, experiences, and knowledge” [10]. As with most other web applications, ETD has a large variety of stakeholders. This multiplicity of interests is amplified by the fact that ETD will be used by, and will impact, many different departments on campus. It was essential, therefore, to meet the needs of developing a web application within a campus environment, to effectively understand and negotiate a broad range of stakeholder needs.

Iteratively Expanding Stakeholder Groups

The process model used in the development of ETD attempted to address this challenge by gradually folding in new stakeholders with each iteration. In its early stages, requirements elicitation focused on identifying and working with a core group of stakeholders. This group consisted of representatives from each of ETD’s user groups—students, Graduate Studies staff, faculty, graduate coordinators, and library staff. As development continued, the following stakeholders were gradually added into the process: the graduate studies committee, additional library staff (from both the local CSUN library and the central library at the California State University’s Chancellor’s Office), campus members attending technology forums, representatives from campus IT, the provost, graduate coordinators from additional departments, faculty who had interacted with the prototype, and the campus accessibility group.
To the extent that this process identified key stakeholders early on, it was successful in eliciting requirements, obtaining user feedback, and mitigating the risk of wasted development time. However, when the ETD team failed to identify the full range of stakeholders that exist on campus, or the diversity that exists within stakeholder groups, the process was less successful.

Risk Mitigation

By following an iterative, prototype-based requirements cycle, the development of ETD allowed for “risk mitigation . . . to take place as early as possible . . . [by helping avoid] the IKIWISI problem . . . collect user feedback . . . [and] avoid late and severe integration problems” [10]. In other words, by having stakeholders interact with prototypes throughout the development process, user feedback was more targeted and accurate. In turn, evolving requirements were more accurate and less development time wasted than if requirements were defined without any user-system interaction.

For example, early in the development process the development team needed to define how to translate the preliminary appointment process into an electronic workflow. Under the paper-based system, all students are required to attend an appointment in which a Graduate Evaluator reviews the formatting of the thesis. From here, students whose work requires significant changes are required to schedule a post-preliminary appointment. Students who do not need to make significant changes, on the other hand, do not schedule an appointment until they are ready for the final submission of their thesis. With the electronic system, some questions immediately arose as to how this workflow would be duplicated. How many times would a student be allowed to resubmit changes electronically before having to schedule an appointment? What would the
appropriate verbiage be for the status of reviewed drafts? What information should be sent to students if they were required to schedule an appointment? Would these students see the changes they were required to make? On the one hand, this information might make the appointments more productive. On the other hand, it might detract from the appointment process and leave the student wanting to resubmit their updated work electronically.

There were several nuanced requirements issues that needed to be resolved for the complete implementation of the appointments system. However, it would be very difficult for Graduate Evaluators to make decisions about this workflow without having any experience interacting with the new system. It is easy in discussing implementation options to arbitrarily pick an option that sounds good. It is difficult, however, to anticipate which workflow details would be the most effective when faced with real students and real evaluations. Furthermore, to implement a complete system before obtaining user feedback would run the risk of wasting development time. Using an iterative process model helped mitigate the risk of poorly defining this workflow. Rather than implement a more nuanced system, the development team implemented a basic workflow in an early prototype, and demonstrated it to Graduate Studies. From here, Graduate Evaluator feedback was based on concrete experience with the system, and requirements decisions and discussions based on common knowledge of the prototype and shared perspective. Thus, the prototype-based process model provided a means of more accurately defining requirements throughout the development process.
Unanticipated User Groups

As user feedback was essential for requirements capture, the ETD development team attempted to define and include as many stakeholder groups and representatives as possible in the requirements elicitation process. However, the organizational structure of the university presented unanticipated challenges to identifying stakeholders. With such a wide distribution of independent departments on campus, selected user groups often proved to be more heterogeneous than anticipated. For example, the development team generally considered “faculty” to be one user group and “students” to be another. Only occasionally did the development team identify differences and subgroups within these groups. For example, the development team did consider differences in comfort with technology. Likewise, the team identified discipline-specific issues such as the different media and file-types a discipline might use to capture student theses (audio files for the music department versus video files for performance art). However, these considerations were more the exception than the rule, and the team was far from successful in identifying all of the ways in which subgroups and departments on campus differ.

As a result of this oversight, stakeholder feedback often came too late in the development process to adequately incorporate it into the product requirements. For example, ETD restricts access to students, faculty, Graduate Evaluators, and graduate coordinators. Each user group has access to different theses, with graduate coordinators having full access to all theses and student information for their department. No other user groups are defined. Consultation with stakeholder representatives from initial pilot departments formed the basis for defining these user groups and access levels. As a result, early pilots with the computer science department, as well as the educational
doctorate program presented no issues with this structure. However, later presentations to other departments and campus groups raised several issues for the definition of user groups. Several of these departments indicated that they needed an additional user group—an administrative user with full view access to all department theses, but without the signature access provided to graduate coordinators. Specifically, this was a feature request of department chairs when ETD was presented (late in the development cycle) to the council of chairs. Likewise, other department administrators currently use some of the paper thesis forms for department specific tasks. Because the early pilot departments have their own paperwork for handling these administrative tasks, the development team did not anticipate the potential role of ETD for such department-specific tasks. In this way, it was not safe to assume that all academic departments would have the same requirements or workflow, even though ETD replaces campus-wide thesis paperwork. Unfortunately, however, the feedback from these departments came too late in the development process to incorporate into the application.

Involving Users Too Late In the Process

Clearly, it is important for risk mitigation to receive user feedback as early as possible in the development process. The ETD development team attempted to do this by incorporating an increasing number of stakeholder representatives with each prototype’s iteration. However, in retrospect, there is a clear and significant difference between a small set of stakeholder representatives and a large number of users within each stakeholder group. Increasing the user count of a prototype increases the likelihood of noticing gaps in functionality and simple oversights made in application design. Often, these users did not even need to interact directly with the application. They simply
needed to watch a demo of the application to provide helpful feedback. For example,
faculty members watching a demo of the application noted the need to handle more than
three committee members. Had the development team included a larger number of
faculty members in initial prototype demonstrations, the requirements engineering
process might have been able to capture this exception earlier in the process.

Because user examination (as opposed to interaction) of the product was often
sufficient to provide useful feedback, it might have been more productive for the
development team to create wireframes or other non-functioning application descriptors
through the first few development iterations. This would have saved the time of
implementing a working prototype whose structure would likely change substantially
upon receipt of user feedback.

Negotiating Expectations

In addition to adequately capturing requirements, the need to negotiate
requirements across separate departments and department resource pools emerged as a
challenge specific to working in a campus environment with a wide, distributed
organizational tree. To the extent that the development team was able to incorporate
knowledge of a department’s business context into the design of the application, they
were able to successfully address this challenge by focusing on user acceptance.

For example, the deposit of Graduate Studies approved theses into the library’s
digital repository introduced the challenge of designing a workflow that bridges two
independent departments, the library and Graduate Studies. This new workflow—
although part of a general system expected to reduce overall costs – has several potential
new costs associated with it. These costs include, among others, the time to review and
edit student-entered metadata, the time to submit the actual thesis to the library, the time to transfer the library record from the digital repository to the library’s main catalogue, the time and resources required to provide copyright education to campus, etc.

Responding to these potential costs, the question arose early in the process as to which department would be responsible for these tasks and their associated costs. Interestingly, this question was made more difficult to answer because of the organizational and financial structure of the university. While the workflow of submitting student work to the library crosses departments, funding does not. This means that the group receiving the most benefit from the workflow modification may not be the same group that must absorb most of the costs. This is the case even within a department. For example, in the library, those who currently handle the printed bound copies should expect to see a workload reduction from the use of ETD. However, these are not necessarily the same group who will be responsible for the additional cataloguing tasks associated with maintaining ETD.

Hence, the situation commonly arose in which stakeholders agreed on a general requirement of the system (in this case electronic thesis storage), but may not have agreed upon the best way to meet that requirement. This presented a unique challenge for requirements elicitation. In response to this challenge, the development team took the general approach of (1) automate as much as possible, and (2) tie the new workflow as closely as possible into the existing workflow to work within the current business context. The goal behind this strategy was to minimize the costs to stakeholders, as well as to minimize the variance in cost-benefit distribution between stakeholder groups to increase the likelihood of user acceptance by all departments.
In the case of thesis storage, the resulting workflow is as follows. First, committee members approve the metadata in the final approval process. Since faculty already must review the Final Approval Form, this process does not add substantially to their workload. In addition, committee members are the most appropriate users to make metadata content decisions, as they are content area experts. By having faculty review the metadata, this relieved the library of having to review the metadata at a content-level. (It is important to note that at many other universities, this task is still undertaken by library staff who review the thesis content and compare it to the metadata or use it to create the metadata).

Once faculty members approve the metadata, the Final Approval Form goes to Graduate Studies for review. As they previously would, Graduate Evaluators confirm that the student has met all graduate requirements and that the thesis is ready for submission. In approving the Final Approval Form, GRIP initiates the upload of the thesis to the library. All of this is done automatically by the application, and the upload task is tied entirely to the existing workload task of thesis approval. Therefore, Graduate Evaluators complete the library submission task without any additional work.

Finally, the thesis is submitted to the library, which is left with the cost of cleaning up the final metadata for formatting, and handling the cataloguing of the record. This cost is consistent with the library’s expertise and their current role on campus. In addition, the cost is reduced through the modification of the repository configuration, to allow the deposit to automatically go all the way from submission to catalogued entry, rather than sitting in a pending thesis pool. By defining such a workflow, the costs associated with digital thesis storage are spread out across all of the departments who
have a stake in the system. Likewise, the benefits of ETD are also distributed across departments. Such a distribution of costs and benefits as a strategy for shaping requirements proved effective for requirements negotiation and meeting the needs of a diverse group of stakeholders on campus.
PRODUCT IMPLEMENTATION

Just as the shared-governance environment and wide, distributed organizational tree at CSUN presented several challenges for process model selection and requirements engineering, these characteristics of the campus environment presented significant challenges for design decisions in the implementation of ETD. The need to establish user acceptance in a shared-governance environment often meant that the development team had to prioritize criteria such as reducing effort expectancy at the expense of other important design criteria. Likewise, the need to work across a wide, de-centralized organizational tree presented many challenges to accurately capturing data structures and use case exceptions. To the extent that the ETD development team anticipated these challenges, they were able to successfully incorporate flexibility into the application design. Where exceptions and data structure differences were not anticipated, however, the development team had to make significant changes to the code base across programming iterations, resulting in unnecessary, wasted implementation time.

The following sections illustrate some of the design decisions the development team made to meet the challenges of application development within a campus environment, along with an analysis of the ways in which these decisions were more or less successful.

System Architecture

Two main criteria served as the basis for selecting the architectural components of ETD. First, the development team selected technology supported by CSUN’s IT environment. After an initial pilot period in a test environment, ETD was to be hosted in the IT infrastructure. It was important for gaining user acceptance of ETD by critical
stakeholders in IT, to reduce effort expectancy by working within this infrastructure. By doing so, the development team would minimize potential problems that could occur in porting the application from one site to the other. Second, in addition to selecting technology supported by IT, the development team always selected open source technology over closed source solutions. This would minimize the cost and effort involved in expanding the existing infrastructure, in cases where it could not meet the requirements of ETD.

Server Side Architecture

Following these two criteria, ETD’s architecture makes use of a standard LAMP stack (Linux, apache, mySQL, PHP). While other programming languages were initially considered (such as Ruby on Rails), PHP was ultimately chosen for the ETD code base because PHP applications are currently supported and used within IT’s infrastructure. Again, this decision was made in part to minimize the sense of change and effort expectancy experienced by stakeholders in IT.

In cases where the current infrastructure does not support all of ETDs functionality, the development team selected open source solutions. For example, one component of ETD’s server-side architecture is clamAV, an open-source anti-virus toolkit. ClamAV consists of a daemon (clamd) which runs on the web server and is available to scan files for viruses. ETD calls this daemon whenever a user attempts to upload a file to the system. The development team selected this software because it is open source and has one of the most up-to-date virus databases available. While not currently in use in the IT infrastructure, it is easy to install and requires minimal changes to the environment.
ETD Architecture Diagram

While ETD runs in a standard LAMP environment, it connects to several other services for key pieces of application functionality. These services are employed and integrated into ETD, in part, to increase usability and gain user acceptance. For example, ETD uses LDAP (Lightweight Directory Access Protocol) to connect to CSUN’s
directory and to authenticate users. This allowed the development team to provide easy access to the system for users accustomed to accessing CSUN services with their campus id and password. In addition, it allowed the development team to make use of existing campus resources and work within IT’s existing infrastructure.

In addition to connecting to CSUN’s campus directory, ETD connects to the library’s digital repository, ScholarWorks (which uses the dSpace application). There were several open-source options available for connecting to this system. For example, stand alone web applications, such as SWORD easy deposit are available. These applications allow students to login to a web interface to submit their thesis documents and metadata to the library. Ultimately, however, the development team chose to use the SWORD API (Simple Web-service Offering Repository Deposit) to connect directly to the server, through the modification of an existing php-sword library. By using this library, it was possible to integrate the deposit process directly into the ETD PHP code and user interface. This kept everything in one system for users, rather than having two separate systems that users had to access (one paperwork system and one deposit system). In this way, it reduced the amount of learning and effort required by users to work with the system.

Client Side Architecture

On the client side, ETD users jQuery to make AJAX calls. While the use of AJAX increased the amount of effort required to implement ETD, it makes the application more responsive to users and increases application usability. In this way, ETD incorporates additional architectural features to increase user satisfaction and user acceptance. For example, ETD uses AJAX to process dialog box form content.
student clicks the “submit” button in a dialog box form, ETD uses jQuery to make an AJAX call. This call submits the form data to a specified php script asynchronously. The AJAX call receives the results of the php script and modifies the DOM based on these results. This allows users to submit and save form segments without requiring page refreshes, making the application more responsive to user actions. Without using AJAX, users would need to submit several individual forms for processing to save information before completing one GRIP form, and each form submission would require a page reload.

In the case of file uploads, it is not possible to use AJAX, as for security reasons javascript cannot access file content from a file input element. In these cases, ETD emulates AJAX functionality by using a hidden frame on the main page into which the file upload form loads. ETD then uses jQuery functions to access the results of the file upload and modify the DOM to reflect these results. This provides the same benefits of an actual AJAX call, making the application more responsive to user action without requiring the time to reload an entire page.

The screenshots on the following pages illustrate how ETD makes use of jQuery and AJAX to improve application responsiveness and streamline the user interface.

Identity Management System for Off-Campus Users

One consequence of focusing on user acceptance is that the development team often had to prioritize usability ahead of other fundamental application requirements. This was the case in implementing the ability for students to include off-campus committee members on their thesis committee. In designing the workflow for this feature, a major implementation decision was determining how to provide off-campus
Dialog Box Functionality

Dialog box content is loaded during initial page load, but hidden.
Clicking the "Enter Topic info" button reveals the dialog box.

AJAX Submit

Clicking "submit" makes an AJAX call that processes and saves the form contents.
AJAX Return and User Interface Update

Hiding Content with jQuery
committee members access to ETD. Campus members accessed ETD through their CSUN login and password, which off-campus committee members did not have.

There were two options for this feature. First, the ETD team could create their own login system for off-campus committee members. This would involve the creation of a basic identity management system, including login and password assignment, password reset options, and ETD user authentication. Second, the ETD team could make use of the existing campus identity management system by requesting account creation through IT. The ETD team chose the latter option because the campus identity management system was already built; was more robust than the proposed ETD-only identity management system (it included various options for resetting passwords, including security questions and support from the IT help desk); and would provide off-campus committee members with additional features such as access to the campus wireless network. Building a separate identity management system for ETD seemed like a waste of development time, as it would mean the partial duplication of an existing system.

During the second pilot, in which most users had off-campus committee members, it quickly became clear that the development team had made the wrong design decision. Although the use of the CSUN identity management system for off-campus committee members saved development time, several usability issues arose. First, because IT had to provision accounts manually, there was a delay between the time at which a student selected an off-campus committee member and when that committee
member received an email from IT with login information. As a result, these users often watched their email accounts at the wrong time and believed they had not received email that did, in fact, get sent to them. Second, the use of two separate systems – the IT identity management system and ETD – meant that off-campus committee members received two separate emails for each form they needed to complete. It also meant that off-campus committee members would have to login and interact with two distinct, unfamiliar systems. They first had to login and interact with CSUN’s campus-wide identity management system to set up their password and optional security questions. They then had to login to ETD to sign off on paperwork. This caused confusion for several-off campus committee members who were not familiar with CSUN’s website, or the distinction between ETD and other CSUN applications. Finally, if off-campus committee members forgot their password, or were assigned an incorrect email address by students, password and email reset requests would have to go through the IT help desk. This would not normally create usability issues for campus members who are familiar with requesting help from IT. For off-campus committee members, however, it presented real usability issues.

As a result of these usability issues, the development team received a significant amount of negative user feedback during and immediately following this pilot. All of the feedback, however, related to usability issues. There were, in fact, no bugs or functional deficiencies in the off-campus committee member portion of the system. It would have been possible not to change any of the off-campus committee member code, and still have a system that met all workflow requirements. In many such scenarios, an organization might simply expect users to learn how to use a new system, or at the very
least, expect users to wait for the next development cycle for revisions. Because user acceptance was so critical to the success of ETD, however, the development team decided to change the handling of off-campus committee members. This meant having to build an ETD-only identity management system to replace the IT-dependent ETD workflow.

There were several consequences of this decision. First, the decision resulted in campus-wide code duplication, as the new identity management system was simply a subset of IT’s identity management system. In a sense, this meant “wasting” development time to achieve user buy-in through the implementation of functionally unnecessary features. Second, to prevent any additional negative word of mouth about ETD, the new identity management system had to be built before the launch of the next pilot. This meant that building the new login system would push back the deadlines on the implementation of other requirements. Often, these were major functional requirements, such as implementing the push of files from ETD to the institutional repository. In this way, user buy in was clearly one of the highest priorities in the project management of ETD.

Although the decision to prioritize usability often meant deadline slippage and duplicated development effort, it was essential for user acceptance. In the case of the off-campus committee member workflow, the decision to rework code was clearly the right decision. During several subsequent demos to faculty and students, users frequently raised questions about off-campus committee members. These questions were the result of either negative user-experience, or negative word-of-mouth. Dissatisfaction with the usability of the system was impacting user acceptance, and it was critical to be able to
present to users the new workflow and a sense that the development team was addressing user concerns.

While the decision to build an ETD identity management system for off-campus committee members prioritized usability over code duplication efforts on campus, the actual implementation of the system at times prioritized usability over some programming best practices. Generally, the ETD team used best practices for the security of the identity management system. ETD stores all passwords as an encrypted string, and makes use of the phpass library for the generation of the string. This library incorporates a salt into the encryption of passwords to protect against rainbow table attacks. It also uses the bcrypt algorithm to protect against brute force attacks against the encrypted password in the database. In addition to the secure storage of passwords in the database, ETD follows best practices with regard to password authentication, password creation, and password resets. For example, ETD ensures that every authentication request calls the phpass functions (whether or not the user is found) to prevent against timing attacks that would reveal whether the uid or password was incorrect. Likewise, rather than send temporary passwords through email to users, ETD sends users a token string which users can use to set their password, and which is nullified upon first use.

While generally following security best practices, the ETD development team decided not to follow the best practice of creating a short expiration period for the password token. While most applications allow the token to be used only for a period of 24 – 48 hours, ETD provides users with a long expiration period. This decision was made as a result of experience with off-campus users’ response time to ETD emails. It was not uncommon during pilots for off-campus users to wait over a month before
responding to an ETD password set request or other ETD email. To ensure these users still had immediate access to ETD, the development team abandoned the practice of creating a real expiration period. In this way, the development team favored usability over security. Fortunately, in the case of ETD, this would not provide a significant security issue for the system, as all committee members attend a thesis defense, and provide an ink signature at the end of the process. In other words, there are other security checks of user identity outside of the system.

Application Workflow – Moving a User Through the Thesis Process

While many implementation decisions were made to increase user acceptance, other implementation decisions were made to incorporate the flexibility needed to handle changing requirements and to address the large number of use case exceptions that exist across CSUN’s wide organizational tree. This was the case in designing the application workflow for moving a student through the thesis process. In this case, the development team foresaw the need to be able to change application code to accommodate changes in workflow requirements, and successfully built flexibility into the application to handle these changes.

As students move through the thesis approval process in ETD, they must complete three main form processes – Planning Form Approval, Preliminary Draft Approval, and Final Thesis Approval. Each form process, in turn, has a workflow in which the state of a process moves from incomplete to complete through a series of form submissions, feedback, modifications, and approval.

ETD controls this workflow and the progress of a student through the system through a series of status codes. So, for example, when a planning form has been
submitted and is awaiting committee signature, the thesis has a planning code status of 2. When it has received all committee signatures and is awaiting graduate coordinator approval, the thesis has a planning code status of 3, and so on.

The use of status codes requires one additional database insert when conditions are met. For example, when a student submits a form, in addition to inserting a student signature into the signatures table, an update must be performed on the thesis table to change the status of that form. However, status codes make the application code substantially more readable than if status-defining-conditions needed to be checked to determine the actual form state. For example, it is much more readable to check that the planning status of a form is 2, rather than querying the database and checking to see that the committee member signature count is less than the total number of committee members. In addition, querying for a thesis status code is often simpler than querying the database for the conditions that define the status code. Obtaining the thesis status code is a simple one-table query, whereas in the above example, checking for signatures would require a query with at least one join. In this way, the increased readability of status codes combined with the simpler condition querying they provide, more than compensates for the additional maintenance of updating and storing the status values.

The initial status code design for ETD included just one set of status codes that progressed from the first state of the Planning Form through the last state of the Final Approval Form. In this way, the status codes defined the thesis approval process as one continuous series. However, very early in the development process, to provide more flexibility for the workflow and easier maintenance and code modification, the status codes were split into three sets – one for each thesis form.
Once again, structuring this type of flexibility into the data and workflow design proved extremely helpful. For example, it wasn’t long into the development process before form dependencies were changed. Initially, the workflow between forms was required to be continuous. So, a student could not progress from one form to another without completing the previous form. Likewise, any modification in an earlier form would reset all later forms. These dependencies were later changed, such that resetting an earlier form might not reset a later form. Having separate status codes for each form made this workflow modification much simpler to make, requiring fewer code changes.

Similarly, the workflow for modifying individual forms changed throughout the development process. For example, the Final Approval Form had a “held by GRIP” status added. By having separate status code sets, modifications to the workflow of one form did not affect the PHP code for processing other forms.

Following are four state chart diagrams that document the progression of each of the three main thesis processes, as well as the progression of a draft through the Preliminary Approval Process.
Database Design – General Principles

In designing the ETD database, the development team followed the best practice of normalizing the database to the third normal form. According to the mySQL documentation, for data size optimization, it is a best practice to “keep all data non-redundant (observing what is referred to in database theory as third normal form).” Instead of repeating lengthy values such as names and addresses, assign them unique IDs, repeat these IDs as needed across multiple smaller tables, and join the tables in queries by referencing the IDs in the join clause” [11].

The Entity Relationship Diagram that follows shows how ids were used to join tables. For example, in the thesis table, rather than use a longer string to indicate research subjects (which would appear in every thesis entry), a researchSubjects table was created through which the description of the research subject could be referenced by id.

There were several cases in which violating normalization rules would have made the initial application code simpler. For example, the current ETD database has a thesis table and a users table. A committeeMembers table references both of these to store the users serving on a given thesis committee. During initial database design, the development team considered eliminating the committeeMembers table and adding three fields to the thesis table (one for each committee member of a given thesis). This would have avoided the need to perform joins across three tables, and would have made querying the database simpler (but not necessarily more efficient). As a result, the development team considered violating normalization rules.
Ultimately, however, there was no justification to de-normalize the database, so the team consistently adhered to third normal form. The result was not only a more efficient database structure, but a more flexible one as well. Generally speaking, keeping the database in third normal form provided flexibility for requirements and workflow changes that were made during the development process. For example, in the case described above, the initial requirements specified three committee members per thesis. However, midway through the development process, this requirement changed to $n$ committee members per thesis. A de-normalized database structure could not have handled this requirements change. To maintain the structure, an arbitrary number of committee member fields would have had to be added to the thesis table to accommodate the upper bound on committee size.

Had the tables not been built in third normal form, the requirements change to $n$ committee members would have required the database to be restructured, and in turn, all of the code that queried the database and processed the query results, as well. While changes in the front-end code for building a thesis committee still needed to be made, these changes were minimal compared to the code changes that would have been required had the database violated normalization rules. In this way, adhering to third normal form not only provides for a more efficient database, but more flexible and maintainable application code. It ensures that workflow rules are expressed through the application code rather than the database structure, and in turn that changes in workflow do not change the underlying data layer of an application.
Entity Relationship Diagram – ETD Database Part I

- **Deadlines**
  - PK: term, year
  - planning, prelim, final

- **StudentLoads**
  - PK: uid
  - start, stop

- **CoordinatorsByPlans**
  - PK: planID, coordinatorID

- **Plans**
  - PK: id
  - name, deptID, libDescr

- **Departments**
  - PK: id

- **Comments**
  - PK: id
  - formID, facultyID, contrRole, tid

- **UsersByUserRoles**
  - PK: uid, role_id

- **UserRoles**
  - PK: id
  - name

- **Users**
  - PK: uid

- **Signatures**
  - PK: id
  - signerID, signerRole, formID, tid

- **Forms**
  - PK: id
  - name

- **GuestAccounts**
  - PK: uid, loginID

- **CommitteeMembers**
  - PK: facultyID, tid
  - committeeRole

- **CommitteeRoles**
  - PK: id
  - name

- **Thesis**
  - PK: tid
  - creatorID, plan, planningStatus, prelimStatus, libData1, libData2, researchSubject, embargoReasons

- **PlanningStatus**
  - PK: id
  - name

- **PreliminaryStatus**
  - PK: id
  - name

- **FinalStatus**
  - PK: id
  - name

- **Resumes**
  - PK: fileID, uid

- **Files**
  - PK: id
  - tid, fileType, fileName1...

- **FileTypes**
  - PK: id
  - name

- **EmbargoReasons**
  - PK: id
  - name

- **ResearchSubjects**
  - PK: id
  - name

- **LibDegrees**
  - PK: id
  - name
Database Design Challenges – Handling Differences Across A Wide Organizational Tree

While adhering to best practices of database design ensures a certain level of flexibility in application maintenance, it was not always adequate or ideal for handling all campus data relationships. Because CSUN has such a distributed organizational environment, different departments inevitably handle the same campus-wide process
differently. These differences are not always apparent, and may not represent the majority of cases. In other words, there may seem to be a level of consistency that does not actually exist. It is important, however, for successful application design that these differences and exceptions be identified such that the application code can address them in its database structure and php code.

The first critical step for identifying these differences is to work with real and complete data sets wherever possible. This lesson became apparent when working with the database and code structure for assigning graduate coordinators to departments. Early in the development process, the ETD team created a data structure whereby each campus department had one graduate coordinator. The development team defined departments by the department codes associated with faculty members in the campus directory. The mapping in ETD between graduate coordinators and these department codes was one to one. Once assigned to a department, the graduate coordinator for each department would use ETD to sign off on department Planning Forms, and to monitor the progress of students within a department.

Unfortunately, this data structure failed to capture the organizational reality of graduate coordinators on campus. First, within each academic department, there are often several academic plans. For example, within the Computer Science department, students can receive a Master’s in Computer Science, or a Master’s in Software Engineering. Each one of these plans has a different code associated with it. Often, different plans have different graduate coordinators. In addition, some plans have multiple graduate coordinators. The development team, however, only considered certain departments in designing the data structure for ETD. Specifically, the team looked at
Computer Science (which has two plans but only one coordinator), and the Ed.D. program (which also only has one graduate coordinator). As a result, the initial database design for handling graduate coordinators did not accurately reflect the real departmental structure on campus.

By the third pilot, when assigning actual graduate coordinators to departments in ETD, this deficiency became obvious. There was no way to assign the appropriate graduate coordinators to academic plans within ETD, and therefore no way to direct student forms to the appropriate graduate coordinators for approval. All of the application code and database structure that relied on graduate coordinator information had to be rewritten. This included adding a plans table, and a coordinators_x_plans table to the database, rewriting queries for finding coordinators’ student loads, modifying the database to connect a thesis to a plan rather than a department, changing the administrative page for maintaining academic plans, updating all of the PHP code related to graduate coordinators, etc. All of this rework could have been avoided had the development team looked at the complete list of graduate coordinators for departments earlier in the process. The difference in structure was not difficult to implement. It was, however, time-consuming and wasteful to re-implement upon discovery of the design flaw.

While working with real data sets certainly would have made for a more effective initial ETD database design, it is not always possible to obtain such complete information. For example, even in looking at the complete association of graduate coordinators with academic plans for the entire campus, some use case exceptions would have been missed. There are a handful of departments across campus, for example, who
provide non-graduate coordinators with the same level of access to thesis forms as graduate coordinators. These staff members rely heavily on such access to complete their department-specific job functions. Because these functions are outside the scope of the campus-wide graduate studies workflow, it is unlikely that any data which graduate studies could provide would contain this information. Furthermore, there was no reason to anticipate such exceptions or to elicit these cases from departments.

In such cases, to make the database flexible enough to handle use case exceptions across departments, one approach could be to treat all data sets as if they require exceptions. Most of the database and code changes required throughout the development process had to do with the way in which data relationships are defined. Initially, for example, the relationship between graduate coordinators and departments was one-to-one. In addressing department differences, the relationship between graduate coordinators and departments became many-to-many. This is where the \textit{coordinators\_x\_plans} table became necessary. From a purely technical point of view, it is possible to treat one-to-one and one-to-many relationships as a special subset of many-to-many relationships. In the graduate coordinators example, the \textit{coordinators\_x\_plans} table would exist, but each coordinator and each plan would appear only once. Queries of the table, in turn, would return only one result.

To build maximum flexibility into ETD, the development team could have built out this type of relationship for all one-to-one data relationships for which it could anticipate the need in the future to accommodate expanded use cases. To do so would simply require building additional tables into the database, and iterating over all query result as if more than one record were expected to be returned. This would provide a
great deal of flexibility and minimize code changes that would need to be made through the evolution of the system. However, this approach has its drawbacks. The resulting database structure and code would require much more overhead. In addition, the code would be less readable and would obscure actual workflow rules. For example, if a student is limited to producing one thesis, treating the relationship between students and theses as many-to-many for maintainability would obscure the real workflow limitation.

Frequently, the ETD development team compromised between these two approaches by building the flexibility into the database, but by programming the actual workflow rules into the php code. For example, the development team created a `users_x_user_roles` table to allow for future many-to-many relationships to exist between users and user roles. However, only relationships that are actually allowed by the current business flow are treated as many-to-many in the application code. This provides a certain degree of flexibility in that changes to workflow rules will not require changes to the underlying database structure. However, it does not build unnecessary inefficiencies into the code, nor negatively impact code readability.

*Database Challenge – Data Integrity on Campus / Storing Data Over Multiple Systems*

Just as the wide organizational tree at CSUN presented challenges for defining data relationships and accommodating use case exceptions, the variety across departments also has implications data integrity across campus.

There are several cases in which ETD uses data that is currently stored in other campus systems. For example, ETD makes use of SOLAR/ HR department codes, academic codes, and faculty information for defining department and plan relationships, and for generating faculty lists for academic plans. Likewise, ETD draws from LDAP to
access basic student information and user roles (to identify a user as faculty, staff, student, etc.). From a purely data-integrity and data-redundancy point of view, the ideal way for ETD to access this information would be to dynamically pull data from SOLAR or the campus directory (either upon every needed access, or with periodic updates), integrate this data with user input, and store only ETD specific data (or frequently accessed data to maintain efficiency). In this way, there would be no data redundancy across database systems. ETD would store only data uniquely relevant to itself, and IT / SOLAR would remain the central data store for campus data.

Such a system of data maintenance, however, is not possible because of the differences in how different departments on campus define the data used by ETD. For example, there is no direct mapping between all Academic Departments and Human Resources departments in SOLAR. Most academic departments map to an HR department; however, there are some cases – such as the Business Department and Tseng College that do not map. This presented several problems for assigning graduate coordinators to academic programs within the Tseng college. ETD uses IT’s department and program codes to define relationships between departments and academic plans. Likewise, ETD assigns graduate coordinators to academic plan codes. This assignment becomes impossible when HR assigns faculty members to a department code that does not map to an academic plan. To avoid this problem, additional codes, assignment methods, or database tables needed to be built into ETD.

Likewise, Graduate Studies only deals with a subset of academic departments, as not all academic departments have graduate programs. This information is not directly searchable through the LDAP system. As such, to limit the departments displayed in
ETD to only those relevant to Graduate Studies, ETD needed to incorporate its own departments table, which largely duplicated existing data sets maintained by IT.

Finally, the library assigns metadata descriptors to each academic plan that produces theses. While most of these descriptors can be mapped to, and programmatically generated from, Graduate Studies department names, there are some cases in which this is not possible. Therefore, the ETD database needed to provide a field in which this information could be stored, even though it largely overlaps with other, existing information.

In all of these cases, the various treatments of similar data across different departments required the use of extra tables in ETD, along with the storage of redundant data (from a campus perspective), and the creation of admin pages as a means of updating and maintaining this data in the ETD database. Of course, this separate storage and maintenance of data sets in ETD will lead to another set of inevitably divergent data on campus. From a practical standpoint, however, this was unavoidable as already existing data differences on campus made direct mapping between systems impossible.

*Different Form Views*

While several challenges to database design emerged from the need to span departments and systems on campus, a less challenging issue of generating form views emerged from the fact that the ETD workflow spans several departments and stakeholder groups. In ETD, there can be up to four user groups acting on a single form—students, committee members, graduate coordinators, and graduate evaluators. Each of these user groups requires a different view of a form’s content, as well as different options for processing a form. In addition, a single user group might require different form views
depending on the status of the form. This need presented two main programming options. First, the development team could develop different pages (and different PHP files) for different users. Second, the team could develop a single PHP file which customizes the form based on the user’s role and the form’s status.

Based on the initial requirements and prototypes of the application, the form views for different user groups were highly divergent. As a result, the development team initially selected the first option, creating a separate PHP file for students on the one hand, and faculty, graduate coordinators, and graduate evaluators on the other hand. However, as development continued, the requirements and components of the different forms merged significantly. To accommodate this change, the development team restructured the code, adopting a hybrid of the two display options. All form views were merged into a single file for each form. This allowed the team to avoid the duplication of code which was beginning to emerge as the views increased in similarity to one another. However, the execution pages for the forms remained split. One file was kept to process faculty, graduate coordinator, and graduate evaluator input. Another file was kept to process student form submission. While the code for these groups to view a form largely overlapped, the form processing code remained distinct. Therefore, making this division improved readability without introducing duplicate code.

The following UWE presentation model presents the way in which the “Thesis/Project Planning Form” was customized based on user group and form status.
Planning Form Presentation Model

- Modification Warning
- Student Information
- Graduation Information
- Topic Information
- Committee Formation
  - Enter Topic Info
  - Enter Committee
- Committee Signatures
  - Student Signature
  - Graduate Coordinator Signature
- Form Approval Options
  - Return Home
  - Submit

Dialog Boxes:
- Topic
- Committee

Customization:
- If topic info exists: display info
- Else: display instructions.
- If committee info exists: display info
- Else: display instructions.
- Do not display if (userRole==student && viewOnly)
- Display only if (userRole != student) && viewOption
- "Not yet submitted," "Requires Changes" or "Modified"
- Load if (userRole==student && viewOnly)
- Display upon respective button click.

- If userRole==Faculty: display committee options;
- If userRole==grad coordinator: display coordinator options;
- If userRole==GRIP: display GRIP options.
In addition to customizing individual form views through the use of user roles and form status codes, it was important to restrict access to forms. Several mechanisms were used to control form access. First, each file in ETD checks the validity of a user’s session (established upon initial login). Users without a valid session are redirected to login again.

For users with a valid session, the user role in combination with the form status are then checked to see if the user has access to the requested form during that form’s particular state. For example, when students have partially completed the Final Approval Form but have not yet submitted it, the contents of the form are not available to that student’s committee members. In such cases, ETD throws an error indicating that the user does not have access to the requested form.

Finally, only a subset of theses are available to any given user. Students can only access their own thesis; faculty can only access theses on whose committee they serve; graduate coordinators can only access theses within the academic plan which they coordinate; and Graduate Evaluators have full access to all thesis forms. Based on these criteria, a function checks that the requested thesis information is within the set of available theses for a user based on an attempt to match the user’s login information with the thesis information in the ETD database. Again, when users try to select a thesis for which they do not have access, ETD throws an error indicating the access restriction.

**Workflow of a Form Between User Groups**

As each user group views a customized presentation of each form, they take different actions on the form at different stages of the form process. Following are activity diagrams that illustrate the workflow and activities involved in the processing of
Planning Forms. Similar workflows exist for the Preliminary Draft Approval Process, as well as the Final Approval Process.

Planning Form Activity Diagram – Overall Activity Summary
Activity Diagram – Student – Submit or Modify Planning Form

Activity Diagram – Committee Members and Graduate Coordinator –

Review Planning Form
DEVELOPING THE USER INTERFACE

In designing the User Interface for ETD, the ETD development team sought to make the application as responsive as possible to students, and as simple to use as possible for all stakeholders, to reduce the perception of effort expectancy and increase user acceptance of the system.

*Using JavaScript to Increase Application Responsiveness*

On the client side, the ETD System uses jQuery (a javascript wrapper) to create a streamlined user interface and increase the responsiveness of the system. In designing the user interface, the ETD development team sought to keep as much information immediately accessible to users as possible, without overloading the user’s screen. Using jQuery UI components such as tabs and dialog boxes made it possible to have form elements or item lists hidden, but viewable without page refreshes. For example, the committee member Home Page separates pending forms into lists by form type. Each list appears in a separate tab – one for Planning Forms, one for Final Approval Forms. Upon page load, tab contents are loaded, but hidden by jQuery. When a faculty member clicks a tab, jQuery functions display the tab contents without having to load them. There is no delay in switching tabs. In this way, the jQuery UI components help emulate the responsiveness of a desktop application in a web application.

Likewise, ETD uses jQuery dialog boxes to hide and reveal form input elements on student forms. Dialog boxes load when the page loads, so the contents are instantly available upon clicking a button. However, only one dialog box is viewable at a time. When dialog boxes are submitted, the content of the forms becomes viewable on the
main screen. This keeps the form input elements from overloading the screen on the main page, and emphasizes the content of the forms over the html form elements.

*Minimizing Effort Expectancy Through the Creation of a Familiar User Interface*

As students typically only interact with the thesis workflow process once, they have little expectation as to how this process should take shape. Most faculty, however, are already accustomed to working with the paper-based system, with which they are very familiar and comfortable. To obtain user acceptance by faculty, therefore, it was essential to minimize the effort expectancy they had in interacting with the system. To do so, the ETD team focused on simplifying faculty workflow and reducing the sense of change faculty would experience in switching to ETD. This, in turn, would reduce the learning curve of faculty interacting with the system.

To minimize the sense of change that faculty would experience, the ETD team sought to keep the electronic forms as similar as possible to the paper forms. As much as possible, the electronic forms contained the same information set as their paper form equivalents. Sometimes this meant retaining unnecessary student information on an electronic form for the sole purpose of duplicating the paper form. For example, ETD sends all form-related correspondence automatically via email between students and faculty. As such, including a student’s phone number and address on a form is no longer necessary. In fact, students would probably prefer not to have their contact information appear on the forms as such. However, because the ETD team focused user acceptance efforts on faculty, and faculty are familiar with the paper form layout, the design decision was made to retain such information.
The ETD team applied the same principle of familiarity to developing the visual layout of the electronic forms. As much as possible, elements in the electronic faculty view of a form appear in the same location as they did in the form’s paper equivalent. Likewise, the css styles applied to the form are minimal and function mainly for information alignment, rather than applying a strong “look and feel” to the site. Again, this decision emphasizes the user acceptance of faculty over the user experience of students, who would most likely gravitate toward a more contemporary layout.

While the ETD team designed the new system to leverage the familiarity of faculty with paper thesis forms, it is important to note that these efforts were always balanced with the effort to keep the web-based user interface consistent with standard web forms. In other words, the ETD team was not willing to sacrifice web familiarity for form familiarity. For example, after the initial pilot with the computer science department, some faculty requested that the online forms duplicate the paper-based forms exactly. This could be done by scanning and splicing the paper forms, and applying the image as a background to the ETD website. Likewise, faculty from this pilot suggested using object character recognition to take completed paper forms, parse the form field content, and insert the data into the database. In addition to being outside the scope of the project, the ETD team opted not to pursue these ideas because they would violate standard UI principles for web design by deviating from standard web form layout and violating user expectations for interacting with a website.

In addition to focusing on familiarity, the ETD team designed the application to be as simple for faculty as possible. When faculty need to review a form, they receive an email with a link to the form. The link prompts the faculty member to login, and then
redirects the faculty member directly to the student’s form rather than to the faculty home page. This redirect process was implemented specifically for the purpose of eliminating unnecessary clicks by faculty. Landing on the home page (default login behavior) would require faculty to make up to two more clicks – the first to select the appropriate list of forms, and the second to select the pending form itself. In this way, the application design required additional implementation time to reduce effort expectancy of faculty members.

*Maximizing Performance Expectancy Through Additional Data Views*

In addition to focusing on reducing effort expectancy through UI decisions, the ETD development team focused on maximizing performance expectancy by creating additional data views for faculty. The ETD team focused on increasing faculty acceptance of ETD by providing simple features that would enhance a faculty member’s ability to perform his or her job with respect to research advisement. First, ETD provides faculty with a student overview section, which lets faculty members view the progress of all their students through the thesis process. This section does not provide new data for faculty—it simply prevents existing data with a different organization and within a different UI. This feature aims at increasing performance expectancy as it allows faculty to identify and advise students who are lagging in the thesis process. Students who, in the current system, would simply “disappear” as they stopped making progress and stopped communicating with faculty, remain listed in the student overview section. In this way, faculty are kept aware of lagging students, and may be better able to advise them as needed.
**DEPLOYMENT**

While the focus on performance expectancy and effort expectancy impacted application design decisions, the ETD team’s focus on the remaining constructs of user acceptance – facilitating conditions and social influence – impacted the way in which ETD was deployed. In other words, these constructs influenced the organizational and political focus of the ETD Team with regard to user acceptance.

The concept of social influence – “the degree to which an individual perceives that important others believe he or she should use the new system” [9] – recognizes that user acceptance by one group is actually a factor in determining user acceptance by another group. The ETD team acknowledged this tendency from the earliest stages of development and sought to harness social influence by gradually deploying ETD in supportive departments and groups on campus. By starting the deployment process with supportive departments, the ETD team hoped to create positive word-of-mouth about ETD, and build application acceptance by leveraging the social influence of students, faculty, and graduate coordinators in these departments.

During the process of gradual deployment, the development team focused significant effort on providing facilitating conditions for user acceptance. Facilitating conditions are “the degree to which an individual believes that an organizational and technical infrastructure exists to support use of the system” [9]. To provide facilitating conditions to early ETD adopters, the ETD team focused heavily on creating user tutorials and help documentation for users. These tutorials consisted of extensive step-by-step instructions with application screen shots geared toward the three main ETD user groups – students, faculty, and graduate coordinators. Extensive tutorials were also
included to aid users in peripheral tasks, such as creating pdfs to submit through ETD. In addition to providing tutorials, the ETD help section also provides answers to frequently asked questions, background information about ETD, and information about how to obtain technical support.

Beyond providing this documentation, the ETD team provided demonstrations and training for all departments before deploying ETD. In these training sessions, the ETD team presented the scope and background to the ETD project, walked through each form for each user group (i.e. faculty completion of the Thesis/ Project Planning Form, and student completion of the Final Approval Form), and gave attendees the opportunity to ask questions about using ETD.

In this way, by providing significant support during pilot deployment as well as deploying the system slowly across departments, the ETD Team hoped to create the organizational atmosphere necessary for user acceptance of ETD.
CONCLUSION

The development of CSUN’s ETD system faced several challenges. These challenges included standard software engineering challenges, web-application specific challenges, and challenges unique to software development within a public university setting. To address these challenges, the ETD development team followed a lightweight, iterative, prototype-based model in which an increasing number of stakeholders was folded into the requirements engineering process with each iteration. The model allowed the ETD development team to successfully negotiate a wide range of stakeholder interests, and handle short development cycles designed to meet demonstration deadlines, elicit user feedback to enhance usability, and build user acceptance. However, the process of rapid prototyping fell short in several areas specific to development within a university environment. Based on lessons learned through this process, the ETD development team recommends balancing a rapid prototype approach with a more detailed planning period designed to adequately define user groups, identify use case exceptions, incorporate real and complete datasets into application design, and program additional flexibility into the application code.
REFERENCES


