VT ETD-db 2.0: Rewriting the ETD-db System

Sung Hee Park
Department of Computer Science
Virginia Tech
shpark@cs.vt.edu

Paul Mather
Library Systems
Virginia Tech
pmather@vt.edu

Kimberli Weeks
Digital Library and Archives
Virginia Tech
kdweeks@vt.edu

Collin Brittle
Business Information Technology
Virginia Tech
rotated8@vt.edu

Edward A. Fox
Department of Computer Science
Virginia Tech
fox@cs.vt.edu

Gail McMillan
Digital Library and Archives
Virginia Tech
gailmac@vt.edu

ABSTRACT

This paper describes a new version of Virginia Tech's ETD digital library system. ETD-db 2.0 is a Web application using the Ruby on Rails framework. It continues to work with any database (e.g., MySQL, PostgreSQL, Oracle, etc.) and is hosted by a Web server. The objectives of rewriting ETD-db are (1) to improve the original powerful functionalities of the current ETD-db, and (2) to provide new reliable and secure features to handle ETD collections. We decided to use Ruby on Rails, a state of the art agile Web development framework, for the sake of improved maintenance. To strengthen the original ETD-db, the current ETD-db system was analyzed and additional needs also collected to be improved in the requirement specification process. To provide new, reliable and secure features, system administrators, managers, authors, catalogers and graduate school reviewers using ETD-db were interviewed to describe their maintenance/usage experiences. Collected requirements were analyzed using the Use Case-Based Requirement Engineering approach. According to the functional/non-functional requirements drawn through use case analysis, an ETD-db 2.0 prototype was designed and implemented rapidly and then a test driven development (TDD) approach was employed to ensure safer and more reliable code. As a result, we have designed additional safer and more reliable features for a new ETD-db system. We currently have an ETD-db 2.0 system prototype, being developed for release in the near future. It includes an integrated directory system using a secured LDAP protocol to authenticate users.

Keywords

ETD-db, repository, security, reliability.

INTRODUCTION

The ETD-db system ("ETD-db home page,") is a digital repository system for submitting, searching, browsing, and managing ETD collections. It was developed in 1998 by the Networked Digital Library for Theses and Dissertations (NDLTD) and Digital Library and Archives (DLA) at Virginia Tech, and many universities have adopted the current ETD-db system as their ETD repository and submission system.

Virginia Tech has been using ETD-db system as its main ETD repository since 1998. In addition to born digital ETDs, retrospectively, paper versions of theses and dissertation have been scanned and integrated into the ETD-db system. Currently, the VT-ETD db system ("Virginia Tech's ETD repository,") hosts over 18,000 ETDs. Over the years since ETD-db has been in production, stakeholders (e.g., system administrators, managers, reviewers, authors) have suggested various improvements. Moreover, a researcher compared DSpace ("DSpace home page,") to the ETD-db system when choosing software to manage ETDs and pointed out some advantages of ETD-db compared to DSpace (Jones, 2004).

This paper describes a new version of Virginia Tech's ETD digital library system. ETD-db 2.0 is a Web application using Ruby on Rails (Ruby, Tomas, & Hansson, 2011), a Model-View-Controller-based Web application framework. It continues to work with any database (e.g., MySQL, PostgreSQL, Oracle, etc.) and is hosted by any Web server supported by Ruby on Rails (e.g., Apache, NGINX). The major objectives of rewriting ETD-db are these: (1) to improve the original powerful functionalities of the current ETD-db, and, (2) to handle ETD collections more reliably and securely.

This paper is organized as follows: we describe the requirements of ETD-db 2.0 from stakeholders in the section on Requirements; illustrate authentication and authorization workflows and design objects diagrams in the section on Design; discuss evaluation of the functions and objects in the Test section; demonstrate usages with scenarios in the section on Scenarios; and, finally, conclude with our contributions and future work plan.

REQUIREMENTS

To provide new, reliable and secure features, system administrators, managers, and authors, catalogers and graduate school reviewers using ETD-db were interviewed to describe their maintenance/usage experiences. Collected requirements were analyzed using the Use Case Based Requirement Engineering approach. All use cases we discovered are illustrated in Figure 1. A blue color represents new features for ETD-db 2.0 and yellow indicates existing features of the current ETD-db system. Among the requirements, we focused on (1) improving the single password per role and (2) supporting fine-grained access control as new features for administration and security.

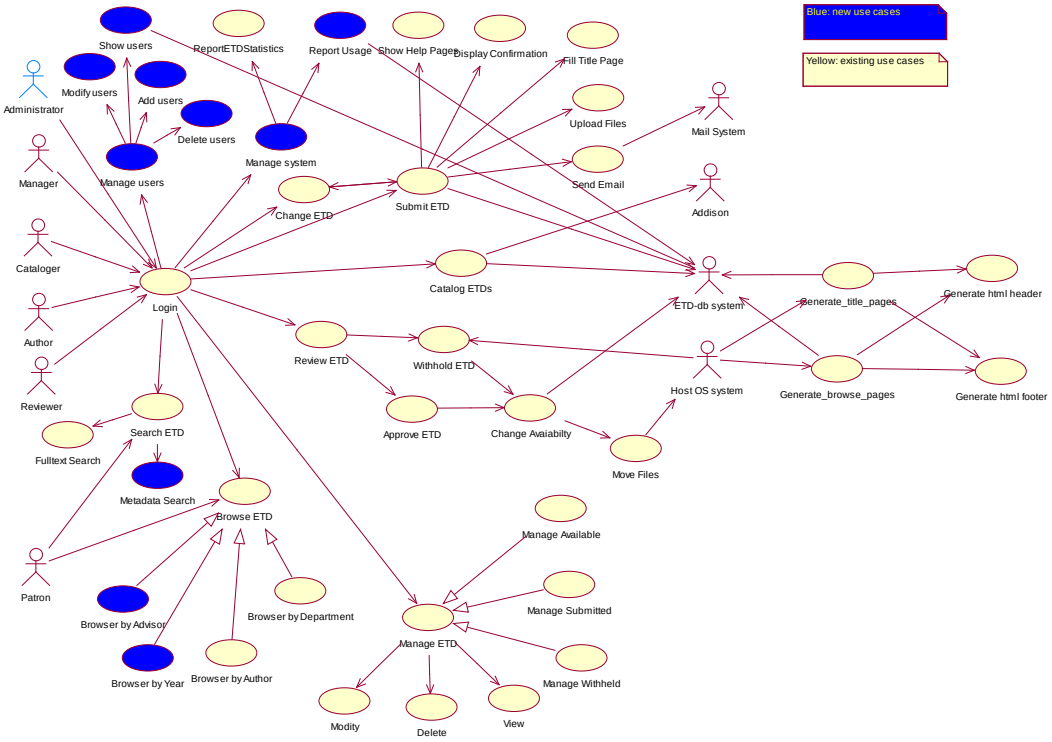

Figure 1. Use case diagram for ETD-db 2.0 system: Use cases (ellipses) and actors (human shape)

Improving one single password per role: In ETD-db 1.0, one single password per role exists even though more than two users have the same role (e.g., student workers for the reviewer). For a more reliable and safer system, role management functionalities with finer-grained permissions need to be extended to multiple admin users.

Supporting fine-grained access control: Currently, ETD-db provides different access permissions to different databases. For example, ETD-db 1.0 has *submitted*, *available*, and *withheld* ETD databases. System administrators, managers, or reviewers have access to all ETD databases while authors have access permission only to the *submitted* ETD database. Different roles should not have only database-level access permission, but also digital object-level and action-level access permission for more secure access control.

DESIGNS

According to the functional/non-functional requirements drawn through use case analysis, models for ETD-db 2.0 have been designed as shown in Figure 2. We also identified seven controllers through requirements engineering. In this section, we focus on the workflows of the authentication, authorization, and submission processes. Unlike ETD-db 1.0, ETD-db 2.0 introduces a Person class, Role, and Permission.

Person: A class that is neutral to roles and contains information about users such as a first name, a last name, an email address, roles, etc.
Role: A class that is neutral to person and contains information about the role itself, like role names.
Permission: A class that defines which actions may be taken on which digital objects by a given role.
Action: A class that describes an action like CRUD (Create, Read, Update and Delete) functions including approve, catalog, import & export.
DigitalObject: A class that represents a digital object, for example, *metadata*, *contents*, and each *role*.

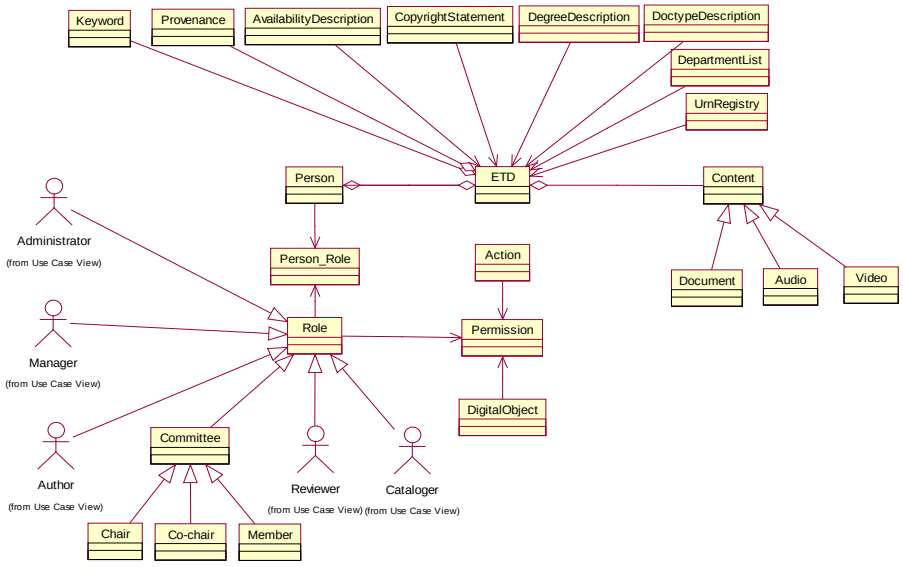

Figure 2. Objects diagram for ETD-db 2.0 system

Authentication

Authentication is a process to check if a user is a registered user (see Figure 3). This process will be verified by centralized authentication Web services (e.g., Google ID, Open ID, Shiboleth, CAS or ED-Auth)[1] or a user table in the database that ETD-db 2.0 hosts. If centralized authentication Web services are used for authentication, the corresponding authentication method will verify the user's credentials. If the user is successfully authenticated, workflow will be passed to the authorization process. If not, the control will be redirected to the 'Forgot Password' and 'Help' page. We are still refining documentation for the diagrams and charts.

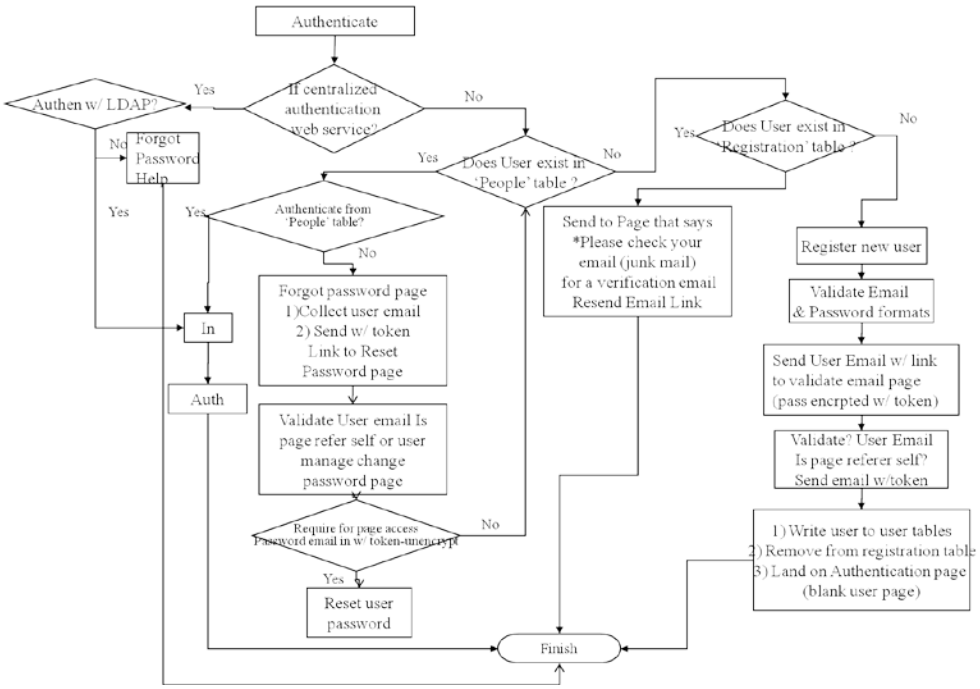

Figure 3. Flowchart for Authentication

[1] Although we designed centralized authentication by web services in the design phase, currently, we have just implemented authentication by only Ed-Auth, an LDAP-based authentication method used at Virginia Tech.

Authorization

Authorization is a process to verify whether a user has an appropriate privilege or not. Figure 4 shows a flowchart for authorization in case of one user who has multiple roles. In the ETD-db rewrite, we have focused on the case of the same person with multiple roles and different users with the same role. ETD-db 2.0 is designed to allow different users to have the same role (e.g., student workers who are authorized to have the privileges of a person in the reviewer's role in order to process a queue of ETDs awaiting approval) and authenticate with their own user name and password. In addition, ETD-db 2.0 aims to authorize a user with multiple roles (e.g., admin, manager or reviewer).

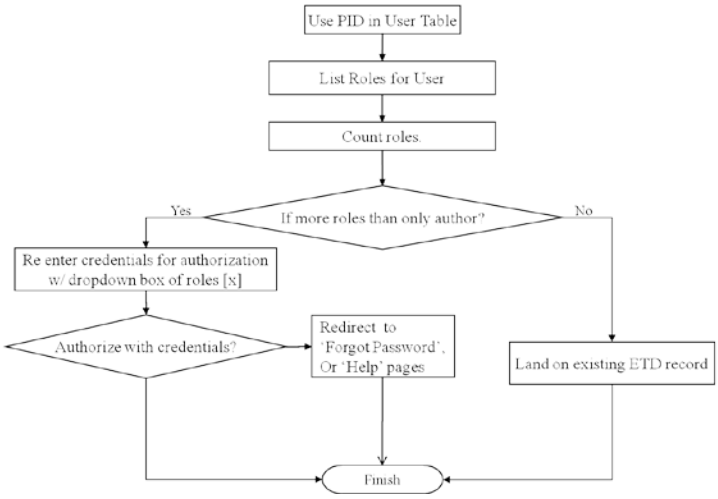

Figure 4. Flowchart for Authorization

Submission

Strictly speaking, a submission process is different from a login process as an author (see Figure 5). Once authentication as a valid user and authorization as an author role has succeeded, the user can see a *new ETD submission* link and a list of incomplete (pending) ETDs, if any. To make it easier for an author to submit a new ETD and to improve metadata consistency, integrating ETD-db with the Banner administrative system is designed as a new function for Authors. Basic Information related to a user (e.g., department, degree, document type, etc.) is provided by the Banner system and injected into the submission process. Finally, the author can enter his or her information through a Web interface.

Interaction between Objects in 'Show ETDs by author' View

We are implementing each view (e.g., show, new, edit, delete, etc.) for a specific controller (e.g., submit, review, manage, catalog, admin, search, browse). For example, Figure 6 illustrates a class diagram for 'show ETDs by author' view.

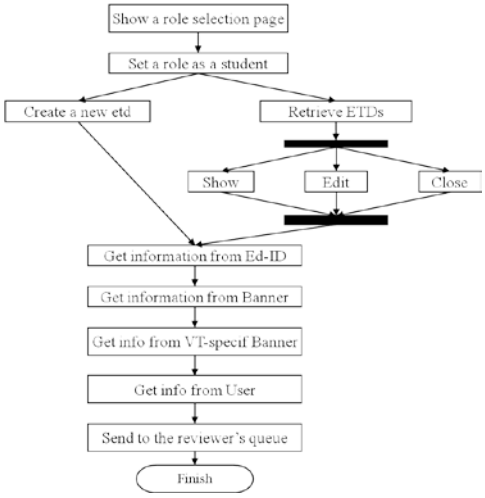

Figure 5. Activity Diagram for Submission Process

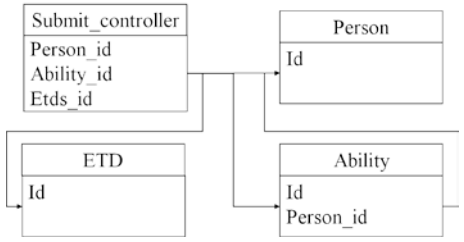

Figure 6. A Class Diagram for 'Show ETDs by Author' View

IMPLEMENTATIONS

Role management

Admin functionalities

We have implemented four admin functions: (a) Register Digital Objects, (b) Register Actions, (c) Register Roles, and d) Assignment of Permissions to each Role. These admin functions were designed to improve the security of the ETD-db system by providing fine-grained access controls.

Authorization for Multiple Roles

We also have implemented several authorization functions for multiple roles of a single user such as (a) different users with the same role, and (b) same user with multiple roles.

Submission process

ETD metadata submission

The current ETD-db system has accrued some inconsistency between metadata as well as between metadata and actual stored files. To solve this inconsistency, we are implementing a submission module that includes the following possibilities: (a) gets information from ED-ID, (b) gets information from Banner, and c) gets information from VT-Specific Banner. We have already completed a submission module that gets all information from users via a Web interface.

ETD file submission

We implemented multiple file uploads. A file can be of various types such as image, audio, video or document. These file types are designed as child classes with their identical single, parent class *Content* (see the *Content* class in Figure 2). This implementation technique is based on object oriented programming, which is more intuitive to humans, so that source code maintenance would be easier for system administrators and application developers. These multiple file uploads implement a compound object concept in digital libraries.

Committee member submission

We implemented a committee member submission process. Committee members consist of committee chair, co-chair, and members according to their roles. All committee members have the same class *Person* but different roles. Thus, the committee member submission process is associated with the *Person* model and the *Role* model.

TESTS

Test Driven Development (TDD)

One of the characteristics of the Ruby on Rails framework is test-driven development (TDD) (Meyer, 1997). Ruby on Rails has three types of tests we are using: (1) unit tests, (2) functional tests, and (3) integration tests. Unit tests are to test our models. Models are ones in the Model-View-Controller (MVC) software architecture. Models are implemented with *class* objects in the object oriented programming approach. Functional tests are to test if users' requests receive the response we expect. In the Ruby on Rails application, the users' requests are mapped to *actions* in a single *controller*. Integration tests are to test users' workflow/scenarios/usages. Thus, integration tests are concerned with testing interaction of actions, which were tested successfully, among controllers. Those step-by-step tests have been improving the quality of our application. As well as testing for current correctness, these tests are invaluable in detecting new errors introduced by future code updates.

SCENARIOS

This section describes some scenarios for the *submission process*, *role management*, and *import/export*.

Submission process

For the submission process, users must pass authentication and authorization. If the user authenticates successfully and has multiple roles, an authorization page will be displayed.

Authentication

Authentication leads to the first page displayed to a user who wants to submit an ETD. Figure 7 shows the authorization login page. There are two cases. If authentication succeeds, the system will check if the user has multiple roles or not. If the authentication fails, the system displays a feedback page.

In response to a wrong username and a password the user will be directed to a page with one link to re-authenticate, and another link to a customizable "Help" page. Since our university uses a centralized authentication system (e.g., ED-ID), the

centralized authentication authority maintains the "Help with PIDs" page.

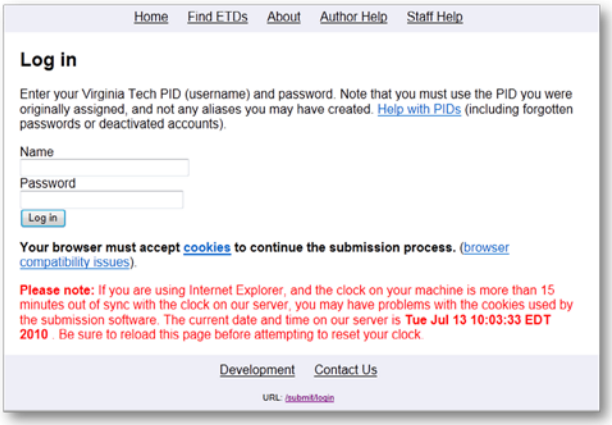

Figure 7. Login Page for Authentication

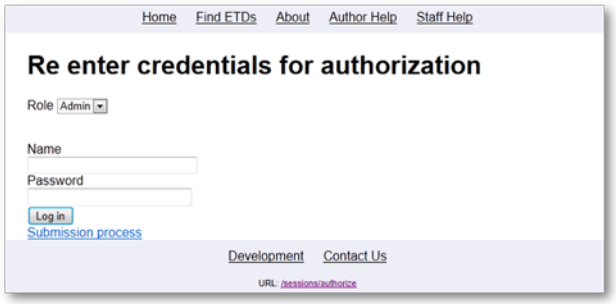

Figure 8. A Page for Handling a Multiple Role Case

Authorization for Multiple role case

Once a user who has only an author role has typed in his or her correct user name and password, the user will be directed to a personal submission page that helps him or her manage multiple or pending submissions. If a user has multiple roles, however, he or she will be redirected to the *re-enter credentials for authorization* page (Figure 8). Note that the 'role' below the page title has a list box. It shows the multiple roles of that user. He or she can choose any role desired to be authorized. Unfortunately, the current ETD-db 1.0 shares a single password for each major role, which is insecure. In the case of a reviewer, for example, because some student workers might be hired temporarily in that role, the password should be changed for the sake of security when such students are no longer being employed. To increase security and auditing capacity, authorization for the multiple-role case has been improved.

This time, in response to a wrong username and a password we display one link to reauthorize, and another link to a default help page that is maintained by the implementers/managers of ETD-db 2.0.

Author role: Author is the default role of all users that have a student position in the university. If a user clicks a link to the submission process, the *Pending ETDs* page (Figure 9) will show incomplete ETDs so that he or she can complete submission by entering missing metadata and files.

To add a new ETD, the user clicks a link to *New etd* (Figure 10). The page displays a form for recording metadata about the new submission that incorporates the existing ETD-MS standards.

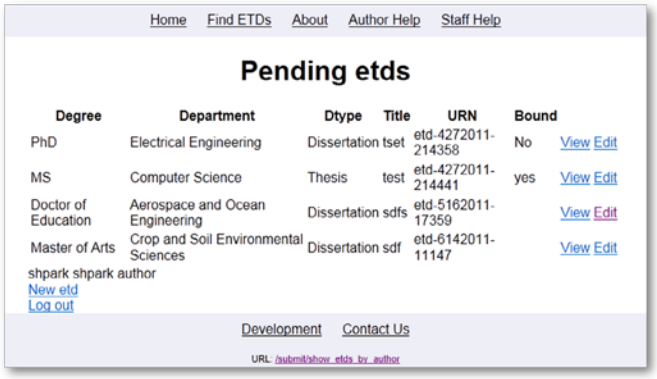

Figure 9. 'Show Incomplete ETDs by Author' Page

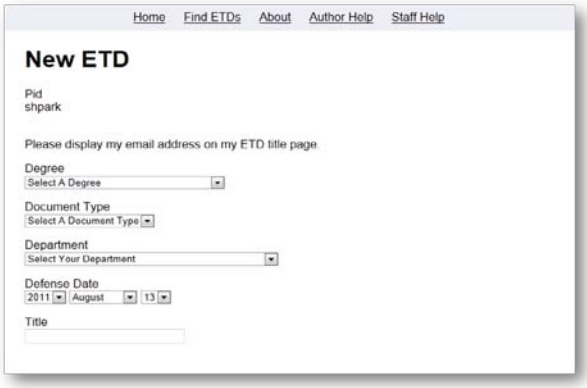

Figure 10. 'New ETD metadata submission' Page

Admin role: If an admin successfully authorizes on *Re-enter credentials for authorization*, the Web application displays the *Admin main* page (Figure 13). The *Admin main* page has many useful functions such as add/show user, add/show role, and add/show permission.

Role management (Admin functions)

Add roles: The first admin function is to add roles (see Figure 11). Before an admin assigns multiple, arbitrary roles to a person, the admin should be able to add a new role.

Add users: Next, the administrator will be able to add a new person to the system and assign roles to that user. (see Figure 12). It is possible to choose multiple roles.

Edit permission per role: We are implementing the *edit permissions per role function* (see Figure 13). For finer-grained access control the administrator can edit permissions according to his or her institution's workflow. The administrator can assign privileges to each role that dictates each action that can be preformed on each object.

Figure 11. 'Add new role' Page

Figure 12. 'Add new user' Page

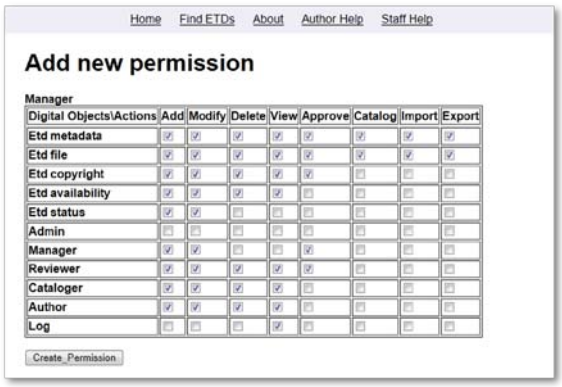

Figure 13. Manager's Permission Example in 'Add new permission' Page

Import/Export

A high-priority addition to ETD-db 2.0 is the ability to import ETDs into the repository or export ETDs out of it. We believe that having a programmatic interface to the ETD repository will make several operations easier to accomplish that are currently difficult to do in ETD-db 1.0. One example is the loading of Bound Theses and Dissertations (BTDs), which are ETDs derived from scanned copies of extant printed theses and dissertations. These currently largely use the existing Web submission workflow, which is a poor fit for this legacy data. A general batch-loading framework would be a much better solution for ingesting such content, as well as for ingesting content from other ETD repositories.

We are investigating the Simple Web-service Offering Repository Deposit (SWORD) protocol as one way to support the import and export of content. SWORD (Allinson, François, & Lewis, 2008) is a lightweight protocol for repository deposit. It is supported by several repositories, such as DSpace, ePrints, and Fedora, and so supporting SWORD should facilitate better interoperability between ETD-db and those SWORD-compliant repositories.

DISCUSSION

To articulate the improvements of ETD-db 2.0, we compare the current ETD-db and ETD-db 2.0 in Table 1. ETD db 1.7c and DSpace v1.1.1 have been already compared by Jones (Jones, 2004).

Stakeholder	ETD-db 2.0	ETD-db 1.0
System Administr-ators	– More maintainable administrator functions like user role management – New import & export function for easier migration from existing repositories – Exploiting the state of the art agile web development paradigm – improved maintenance – Eliminates inconsistencies between the file structure and database – Supports explicit UTF-8 character set encoding – Record log and audit files – Incorporated features for BTDs (scanned bound theses/dissertations)	– Share a single password per each role (e.g., reviewer and system administrator) – Does not support import & export function – Written in Perl scripts which provide software libraries depending on back end database – Does not support transaction processing – Character set encoding is not strongly enforced, which leads the inconsistent output – Does not support log and audit files – External PHP scripts support batch BTD loading into ETD-db 1.0
Managers	– Safer, more reliable 'change_availability' process through concurrency control and transactional processing support – Better approval and release notification	– Does not support transaction processing and relevant rollback.
Authors & Managers	– Connection to Banner/HR system – Provides authors with better feedback about progress and submission status – Supports explicit UTF-8 character set encoding	– Does not connect to Banner/HR system
Cataloger &Reviewer	– User role management and authentication (LDAP) – Better queue management and notification – Set date for automatic release notification – Turn off or extend date for automatic release	– Share a single username and password per each role – Hardcoded release date and release by requests from staffs or authors

Table 1. Comparision of ETD-db 2.0 and ETD-db 1.0

CONCLUSIONS & FUTURE WORKS

ETD-db 2.0 seeks to improve the current ETD-db system in terms of reliability and security giving many benefits to all stakeholders (e.g., libraries, graduate schools, and contributors) and users (e.g., system administrators, managers, reviewers, catalogers, authors, etc). For our development process, a state of the art Web development framework, Ruby on Rails, has been used to support requirements clarified by requirement analysis. Security issues have been improved by fine-grained access control, increased audit logging, and maintenance eliminating inconsistencies between the file structure and database. In addition, more reliable content management has been accomplished by increasing the consistency between contents and their metadata, ensuring content integrity, and implementing version control for each ETD. One of many future works is to make the new system more stable by means of repetitive tests, debugging, and troubleshooting.

We still have plans and future work for implementing and testing ETD-db 2.0. We plan the following: (a) get permission to access the VT Banner system for integration (in discussion), (b) interview reviewers (Graduate School), and catalogers to refine their workflow requirements (c) implement audit logging and provenance for seamless communication between author and reviewer (in design and to be implemented), and (d) add an import and export function (to be implemented in the near future).

ACKNOWLEDGMENTS

This project is supported by Digital Library and Archives, University Libraries, Virginia Tech.

REFERENCES

1. Allinson, J., François, S., & Lewis, S. (2008). SWORD: Simple Web-service Offering Repository Deposit. *Ariadne, 2008*(54). Retrieved from http://www.ariadne.ac.uk/issue54/allinson-et-al/
2. DSpace home page. from http://www.dspace.org/
3. ETD-db home page. from http://scholar.lib.vt.edu/ETD-db
4. Jones, R. (2004). DSpace vs. ETD-db: Choosing software to manage electronic theses and dissertations. *Ariadne, 2004*(38). Retrieved from http://www.ariadne.ac.uk/issue38/jones/
5. Meyer, B. (1997, May). Practice to Perfect: The Quality First Model. *Computer, 30*, 102-106.
6. Ruby, S., Tomas, D., & Hansson, D. H. (2011). *Agile Web Development with Rails* (Fourth Edition ed.): The Pragmatic Bookshelf.
7. Virginia Tech's ETD repository. from http://etd.vt.edu