Using Ajax to integrate institutional electronic theses and dissertation repository with corporate systems at University of São Paulo

Rogerio Toshiaki Kondo (rogerio@sc.usp.br)[*], Maria de Lourdes Rebucci Lirani (lurdinha@sc.usp.br)[*], Vinicius Augusto Tagliatti Zani (Vinicius.Zani@gmail.com)[*], Caetano Traina Junior (caetano@icmc.usp.br)[**], Paulo Cesar Masiero (masiero@icmc.usp.br)[**], Fabio Kon (kon@ime.usp.br)[†]

[*]*Centro de Informática de São Carlos, University of São Paulo (USP)*
CP 438 – 13560-970 – São Carlos – SP – Brazil
[**]*Instituto de Ciências Matemáticas e de Computação, University of São Paulo (USP)*
CP 668 – 13560-970 – São Carlos – SP – Brazil
[†]*Instituto de Matemática e Estatística, University of São Paulo (USP)*
Rua do Matão, 1010 – Cidade Universitária – 05508-090 – São Paulo – SP - Brazil

Abstract

The University of São Paulo (USP) has the largest public graduate system in Brazil with more than 25.000 enrolled students and produces more than 5.000 doctoral theses and master dissertations every year. These students are distributed in 236 graduate programs in all human knowledge fields, which are grouped in 53 different academic units located in 6 cities. Theses and dissertations in paper are archived in 39 libraries. Since 2001, the Digital Library of Thesis and Dissertations of USP (DLTD/USP) is being used as repository of this work.This paper aims at describing and discussing the Ajax implementation used to integrate the DLTD/USP with corporate systems taking into account the university's spatial and institutional diversity. Ajax is one of the key components of Web 2.0. The university graduate studies information system, which is used by all programs throughout the academic life of the students, is used as input for the DLTD/USP system for uploading their thesis or dissertation. Institutional information, such as author's name, date of defense and advisor's name, are retrieved from this corporate database. Students use the account of the graduate studies system to do this activity. Members of the graduate registrar office access the DLTD/USP system to review and approve the submitted electronic thesis or dissertation. During the catalog process executed by members of the staff of each library, a MARC file is generated as output to feed the University library catalog, where theses and dissertations available on the DLTD/USP can also be retrieved.

Introduction

The University of São Paulo (USP) was the first Brazilian institution to implement a corporate digital library of theses and dissertations. Until then, there had been some departamental or research groups of digital libraries in Brazil. The Digital Library of Theses and Dissertations of USP (DLTD/USP) [1] was inaugurated in June 2001. Although USP has the largest public graduate system in the country, producing about 2,300 doctoral theses and 3,200 master dissertations each year in its 236 graduate programs, this annual production was not totally available at the Internet until the middle of 2006. Due to an administrative edict of Coordenação de Aperfeiçoamento de Pessoal do Ensino Superior (CAPES) [2], a federal agency responsible for evaluation of all graduate programs in Brazil, all defended theses and dissertation at USP are now obligatorily deposited in the DLTD/USP. Currently, the DLTD/USP has more than seven thousand documents (Figure 1) and accounts for more than twenty and two million accesses in 2006 from more than 130 countries.

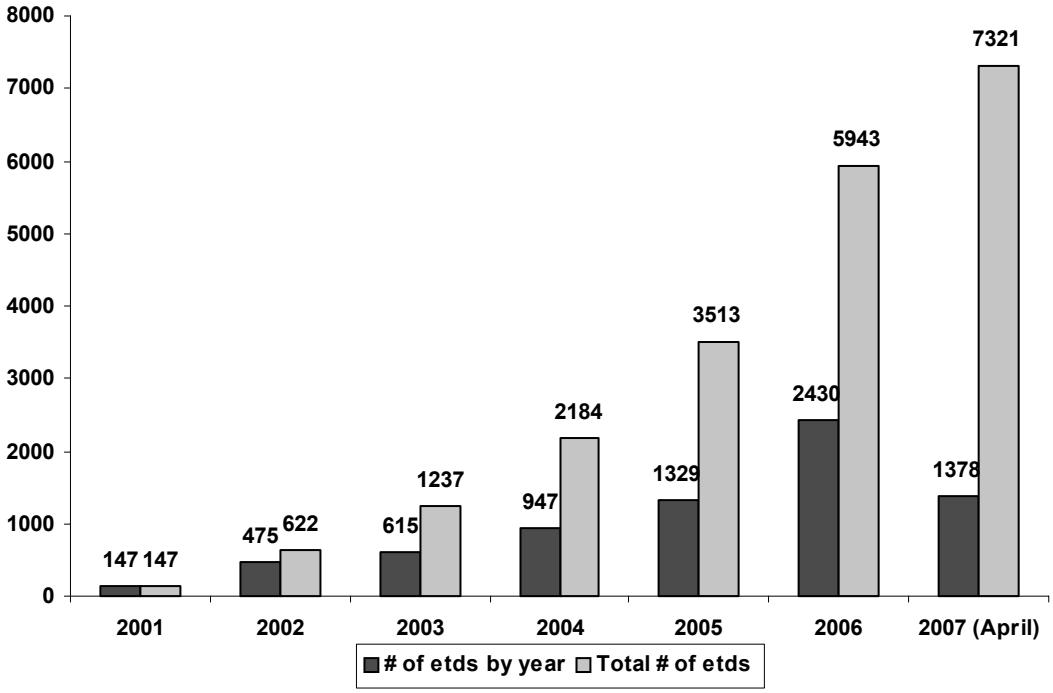

Figure 1: Evolution of available electronic theses and dissertation at the DLTD/USP.

The DLTD/USP adopted the digital content generation system of the Networked Digital Library of Theses and Dissertations (NDLTD) [3], which was adapted to USP's requirements [4]. Since its implementation, there has been a great concern about how to integrate and keep data consistent between the DLTD/USP system and the corporate systems: the Graduate Studies Information System (Janus/USP) [5], which provides thesis or dissertation data, and the Library Catalog (Dedalus/USP) [6], which offers a public way to retrieve data for all theses and dissertations defended at the University (Figure 2).

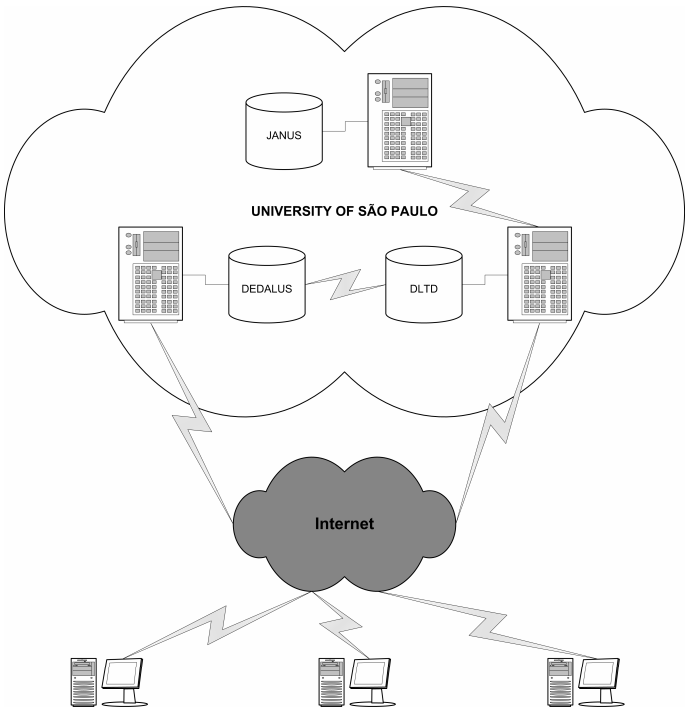

Figure 2: Integrating corporate databases with the DLTD/USP: the corporate graduate system Janus/USP and the institutional library catalog Dedalus/USP.

In August 2006, the adapted NDLTD system written in the Perl language was replaced by a new version written in the PHP language [7]. New concepts of Web application's development and Ajax technology have been used. This innovation in the DLTD/USP system was able to absorb the increasing demand due to mandatory deposit – Figure 1 shows that available documents in the first four months of 2007 is greater than half of the total available documents in 2006. This work describes and discusses the use of this new technology to integrate the electronic theses and dissertation (etd) repository with the corporate systems of the University of São Paulo.

Understanding the academic institution

Nowadays, USP has more than 25.000 enrolled students distributed in its 236 graduate programs in all human knowledge fields. These programs are grouped in 53 academic units located in 6 cities in the State of São Paulo. Although each academic unit is subordinated to the University's academic regulation, they have their own specificities and autonomy to work. All these units use Janus/USP throughout the academic life of their students.

When defended, the theses and dissertations are obligatory archived in paper in one of the 39 libraries also located in those 6 cities. These libraries have to record all theses and dissertations defended at the University in their library catalog to allow bibliographic references to be recovered by anyone as well as to keep them available on the shelves for public access.

The main challenge to the digital library's development was to define how to integrate existent activities in the academic units and libraries with the new tasks introduced by the DLTD/USP. Thus, a digital content generation workflow was defined to attend the University's academic diversity and complexity [4]. This workflow was evaluated and modified afterwards to speed up the demands of deposit obligation as well as to improve the integration between the corporate systems.

The new digital content generation workflow

The new digital content generation workflow has three main stages carried out by three different actors as Figure 3 shows: submission stage (students), reviewing stage (graduate registrar offices) and cataloging stage (library staffs). This new workflow is different from the NDLTD's original one which has the accounting submission stage. All graduate students have an account to access the Janus/USP system, where they can accomplish their registrations and follow their academic evolution. Using the same account, the student does his submission in the DLTD/USP system during the thesis deposit date and the degree homologation date. Some metadata are retrieved from the Janus/USP database. Other metadata are provided by students, besides the inclusion of their PDF files. At the end of this stage, the system generates a new etd.

Using its own Janus/USP account, the graduate office accesses the digital library system to review and validated the submitted etds. This can only be done after the degree homologation of the defense. In this case, other metadata, like defense date and advisor's name, are retrieved from the Janus/USP database. The graduate office then approves the etd to be registered into the digital library.

The etd registration is executed by librarians working in the library where the thesis or dissertation in paper will be archived. They use the NDLTD/USP accounts to access the

system. The cataloging stage allows the library staff to change some etd metadata, normalizing them according to Brazilian norms related to bibliographic standards. After this changing, etd metadata are converted to Machine-Readable Cataloging (MARC) data, which are used to supply to the Dedalus/USP database. Finally, this stage generates a title page for the etd to become publicy available. As MARC data also contain the etd URL, a search in the Library Catalog Dedalus/USP can also retrieves the digital content available in the DLTD/USP.

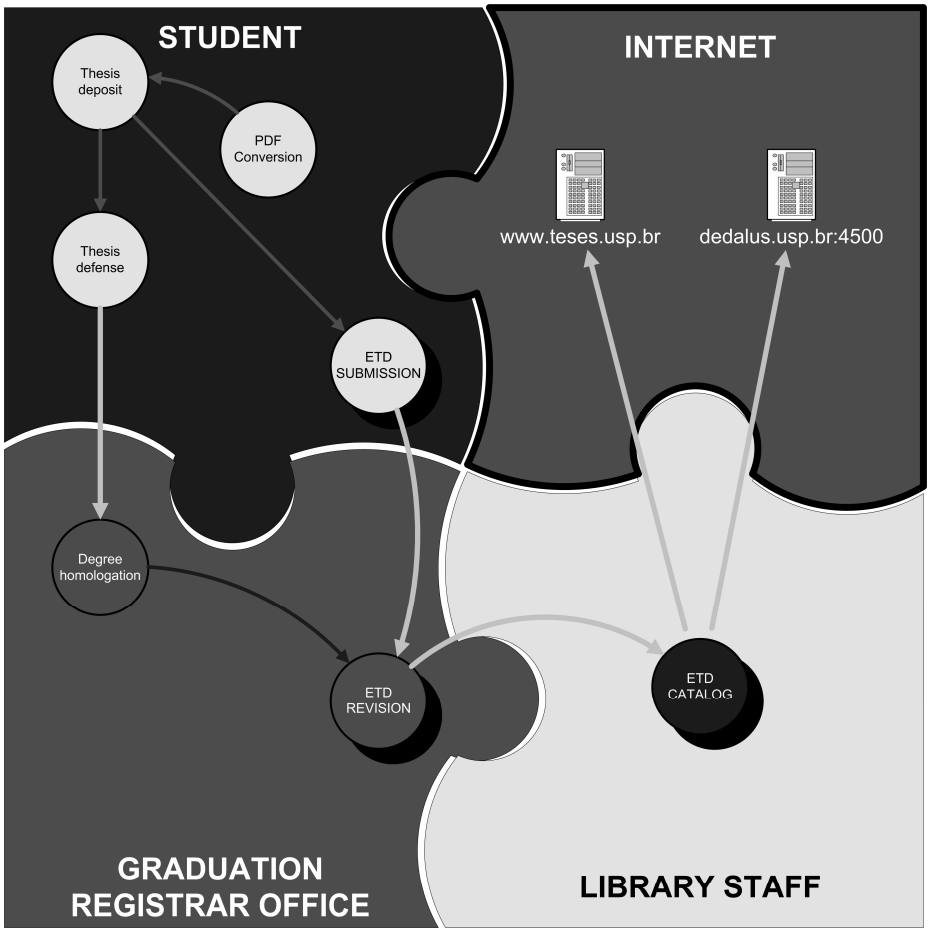

Figure 3: Digital content generation workflow and its actors at the University of São Paulo: students submit their etds to the DLTD/USP, members of the graduate registrar office validate them, and librarian staffs register the etds to be accessible from two sites in the Internet.

Integrating with corporate databases

A challenge during the DLTD/USP system's development was to ensure the data integrity and consistency during the digital content generation workflow aiming avoiding errors between the available data in the digital library and the corporate data used by the graduate offices and libraries. So, a strategy to integrate the existent databases with the DLTD/USP database was developed.

The Janus/USP database is implemented in Sybase and is managed by a central office. This database contains all information related to the graduate student's academic life, such as registration dates, enrolled subjects and other data like defense date and advisors' names. Some of these data are considered private and confidential such as grades. Therefore, for security reasons, a specific database view of Janus/USP is used and it is accessible only by the DLTD/USP system (Figure 4).

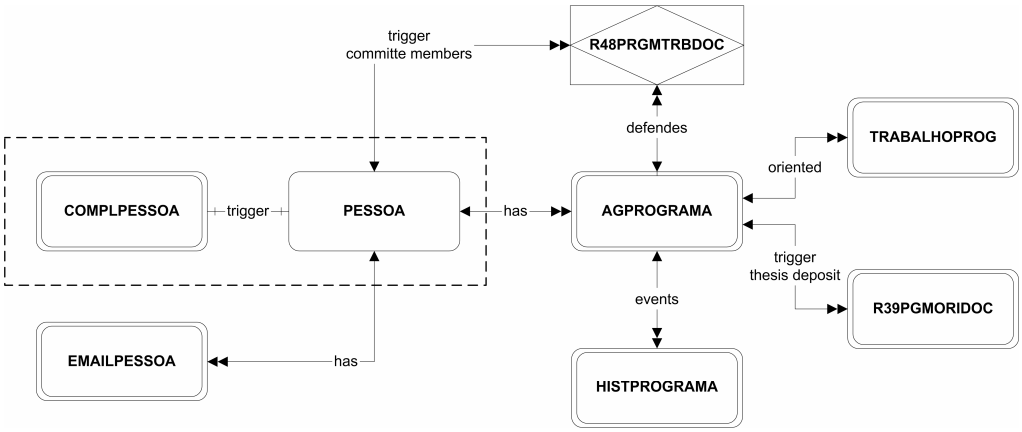

Figure 4: Database view from the complex database of Janus/USP: only the needed data is available to the DLTD/USP.

This view is not implemented in the central Janus/USP database server but in a replication server that belongs to the DLTD/USP intranet. In a Linux server, the Sybase Adaptative Server Enterprise 12.5 (Expression Edition) [8] was installed and the tables of the view were created and populated. Using the Sybase Replication Server, the central database updates the replication server as soon as data are changed.

Table 1: Student's metadata retrieved from corporate database during submission stage.

Metadata	**Database table**
Name	PESSOA (PERSON)
USP ID	AGPROGRAMA (PROGRAM)
Email	EMAILPESSOA (EMAIL)
Academic unit	AGPROGRAMA (PROGRAM)
Graduation program	AGPROGRAMA (PROGRAM)
Document title (if available)	TRABALHOPROG (WORK)
Program level	AGPROGRAMA (PROGRAM)

When the students submit their work, the metadata related to them are retrieved from this replicated database and then stored into the DLTD/USP MySQL database to optimize the access (Table 1). Students cannot change these metadata but they are responsible to provide other metadata necessary for the etd, such as the abstract and keywords. They are also required to upload the PDF files.

In the reviewing stage, after degree homologation, the DLTD/USP system retrieves other metadata to complement the submitted etd (Table 2). The graduate office must then verify all metadata. It also does not have permission to change them. Any error detected in these metadata must be corrected in the original system (Janus/USP), which will be replicated to the local server and, finally, updated in the DLTD/USP database.

Table 2: Student's metadata retrieved from corporate database during reviewing stage.

Metadata	**Database table**
Defense date	AGPROGRAMA (PROGRAM)
Homologation date	AGPROGRAMA (PROGRAM)
Advisor's name	PESSOA (PERSON)
Advisor's USP ID	R39PGMORIDOC (ADVISOR)
Advisor's title	R39PGMORIDOC (ADVISOR)
Advisor's email	EMAILPESSOA (EMAIL)
Committee member's name	PESSOA (PERSON)
Committee member's USP ID	R48PGMTRBDOC (COMMITTEE)
Committee member's title	R48PGMTRBDOC (COMMITTEE)

In the cataloging stage, the library staff has the permission to change some metadata to normalize them according to Brazilian standards. After this correction, a MARC file is generated to feed the Dedalus/USP database, implemented in ALEPH.

Implementing the new process with Ajax

The NDLTD original system has characteristics of a traditional Web application, where user's actions generate HTTP requests to the application' server. Then, the server executes all needed instructions and returns a HTML page to the user's browser. During this server's processing time, the user just waits for the next step. With the new development technologies for Web applications such as Ajax, the user's interaction has been improved, avoiding this inactivity while wating for the next step. Users now do not notice that the application is interacting with the remote server asynchronously (Figure 5).

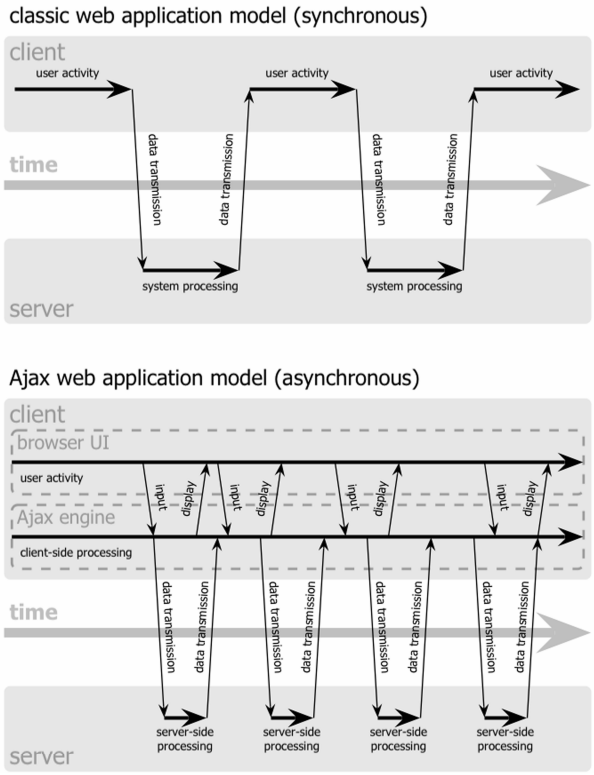

Figure 5: The synchronous interaction of traditional Web application (top) compared to the asynchronous pattern of an Ajax application [9].

Ajax is an acronym for Asynchronous JavaScript and XML and it is one of the key components of Web 2.0 [10]. It is a set of tools which allows more interaction activity between the server and the browser, incorporating XHMTL and Cascade Style Sheet (CSS) for standards-based presentation, Document Object Model (DOM) for dynamic display and interaction, XML and XSLT for data interchanging and manipulation, XMLHttpRequest for asynchronous data retrieval and JavaScript to binding them all [9].

Instead of loading a web page, an Ajax engine is loaded into the user's browser. This engine is responsible to present the application interface as well as to communicate with the server, according to browser's events. Using this engine, we can load web pages or pieces of HTML code, style sheets code, JavaScript code or whatever information is needed for the application (Figure 6).

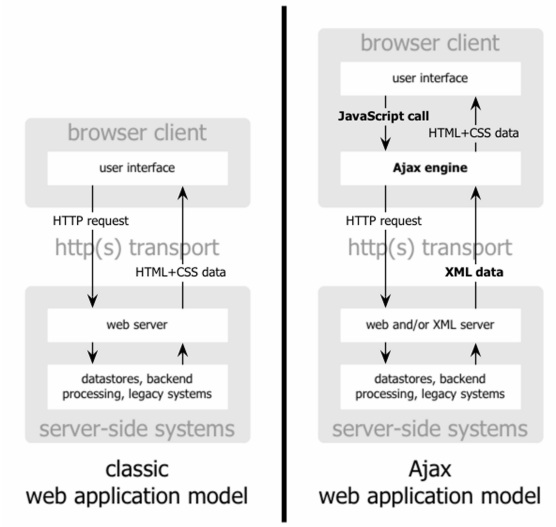

Figure 6: Traditional model of Web application (left) compared with the Ajax model (right) [9].

PHP xajax library

There are many Ajax implementations nowadays [11]. As the PHP language was chosen to program the new digital content generation system for the DLTD/USP, the xajax library was adopted [12]. It is an open source class library that generates and maps JavaScript functions into PHP functions, which can be called asynchronously in the remote server. It is comprised by three components:

- **JavaScript engine**: Residing in the user's browser, it is responsible for generating requests asynchronously to the server and for interpreting the XML results in browser's commands.

- **Xajax PHP class**: It resides in the server and is responsible to register and execute the mapped PHP functions requested by the JavaScript engine.

- **XajaxResponse PHP class**: It also resides in the server and it is used by PHP functions called in the xajax PHP class. These functions instantiate an xajaxResponse object that generates a XML response at the end of execution of these functions.

To illustrate the using of xajax in the DLTD/USP system, we choose the simplest case in the reviewing module: showing a list of submitted etds.

Xajax library example of use

The first step is to log in the system. The graduate office types its Janus/USP account. The connection mode is synchronous without using xajax to send user's data (ID and password) in a secure connection. As soon as the authorization is allowed, the connection becomes an ordinary web connection and the browser is redirected to the only page of the module: *index.php*.

The xajax PHP class, which is in *index.php* code, checks if the call is an asynchronous connection using the method *processRequests()* by verifying the existence of the *xajax* variable (lines 1 to 8 of Table 3). This variable contains the PHP function name to call if the connection is asynchronous. As this is the first call to *index.php* page, the variable *xajax* does not exist and the whole content of the page is sent to the browser.

Table 3: PHP class methods responsible for verifying asynchronous connections (server side) [12].

```
 1: function getRequestMode()
 2: {
 3:     if (!empty($_GET["xajax"]))
 4:             return XAJAX_GET;
 5:     if (!empty($_POST["xajax"]))
 6:             return XAJAX_POST;
 7:     return -1;
 8: }
 9:
10: function processRequests()
11: {
12:     $requestMode = -1;
13:
14:     $requestMode = $this->getRequestMode();
15:     if ($requestMode == -1) return;
16:
16:     ...
17:
18:     if ($requestMode == XAJAX_POST)
19:     {
20:             $sFunctionName = $_POST["xajax"];
21:
22:             if (!empty($_POST["xajaxargs"]))
23:                     $aArgs = $_POST["xajaxargs"];
24:     }
25:     else
26:     {
27:             header ("Expires: Mon, 26 Jul 1997 05:00:00 GMT");
28:             header ("Last-Modified: ".gmdate("D, d M Y H:i:s")." GMT");
29:             header ("Cache-Control: no-cache, must-revalidate");
30:             header ("Pragma: no-cache");
31:
32:             $sFunctionName = $_GET["xajax"];
32:
33:             if (!empty($_GET["xajaxargs"]))
34:                     $aArgs = $_GET["xajaxargs"];
35:     }
36:
37:     ...
38:
39:     $sContentHeader = "Content-type: text/xml;";
40:     if ($this->sEncoding && strlen(trim($this->sEncoding)) > 0)
41:             $sContentHeader .= " charset=".$this->sEncoding;
42:     header($sContentHeader);
43:
44:     ...
45:
46:     print $sResponse;
47:
48:     ...
49:
50:     exit();
51: }
```

Attached to the initial code page, the JavaScript engine is also loaded into the user's browser. After this loading, asynchronous connections can be made. When the page is completely loaded, an asynchronous connection to show the list of submitted etds is called in the HTML tag *body* (Table 4).

Table 4: HMTL code responsible for the first asynchronous call (browser side).

```
...
<body onLoad='JavaScript: Loading(); xajax_List_etds(1);'>
...
```

Then, the asynchronous connection is made by XMLHttpRequest and, in the server it is identified by *processRequests()* using the *xajax* variable. In this case, the value is *List_etds*

(line 32 in Table 3, because the method is GET). If there are any arguments for the function, these arguments can be accessed using the *xajaxargs* variable. In this case, the value is *1* (lines 33 and 34 in Table 3).

The PHP functions that can be called in asynchronous connections must be registered in the page *index.php*. The registration is done using the xajax PHP class' method *registerFunction()* (line 4 in Table 5).

Table 5: PHP code to register the function *List_etds* to be called in asynchronous connections (server side).

```
1: function RevisaoXajax() {
2:
3:     $this->xajax = new xajax();
3:
4:     $this->xajax->registerFunction(array("List_etds", $this, "List_etds"));
5:     ...
6:     $this->xajax->processRequests();
7: }
```

Each one of these PHP functions must instantiate an *xajaxResponse* object to return data to the user's browser (line 5 in Table 6).

Table 6: PHP code for *List_etds* function (server side).

```
 1: function List_etds($pagina) {
 2:
 3: ...
 4:
 5:     $objResponse = new xajaxResponse();
 6:
 7: ...// Retrieves data from DLTD/USP database and generates new HTML code
 8:
 9:     $objResponse->addAssign("core_content", "innerHTML", $this->tpl->get());
10: ...
11:
12:     return $objResponse->getXML();
13:}
```

Before returning data using the method *getXML()* (line 12 in Table 6), it is necessary assign an action to inform the browser to change the present content with the new generated HTML code (line 9 in Table 6). At the end of *List_etds* execution, an XML object is returned to *processRequests()* function, which then pass to the browser (lines 39 to 46 in Table 3). Table 7 illustrates an XML generated by *xajaxResponse()*.

Table 7: XML code generated by *xajaxResponse* object. The command (`n="as"`) assigns the target *core_content* (`t="core_content"`) to modify the *innerHTML* property with the content of CDATA parameter.

```
<cmd n="as" t="core_content" p="innerHTML">
<![CDATA[
<!-- Data generated from List_etds function -->
]>
</cmd>
```

The JavaScript engine always checks the status of asynchronous connections. If it is modified, it means that an answer was received from the server. When this happens, the engine interprets the received data and executes the commands using the JavaScript *processResponse()* method (lines 6 to 13 in Table 8).

Table 8: Xajax engine JavaScript extracted code used to process the XML response (browser side) [12].

```
 1: this.processResponse = function(xml)
 2: {
 3:
 4: ...
 5:
 6:     else if (cmd=="as")
 7:     {
 8:             cmdFullname = "addAssign/addClear";
 9:             if (this.willChange(id,property,data))
10:             {
11:                     eval("objElement."+property+"=data;");
12:             }
13:     }
14:
15: ...
16:
17: }
```

After all this processing, the list of submitted etds is showed to the graduate office.

All these tasks are done using just only one page of the module (*index.php*). For browsers, this page has three distinct areas, which are mapped to specific templates that are loaded according to xajax requests: the *title area*, which is instantiated when the page is first loaded, the *menu area*, where different menus are loaded depending on the module status, and the *data area*, where data are presented and also edited. Figure 7 shows the first screen of the reviewing module with a list of submitted etds.

Figure 7: Template areas of the DLTD/USP system: only one page is loaded into the user's browser after logging in the system. The menu and data area are interchangeable with other HTML code according to user's events.

Other considerations

Besides Ajax, other Web applications' development techniques are used in the new digital content generation system to make the code maintenance easy. One of the problems with the NDLTD original system is the mixture of Perl code with HTML code. Thus, templates are used to separate the PHP code from the system layout. The PEAR Sigma library is used [13] for this task.

To access all involved database (MySQL and Sybase), the PEAR DataObject library is used to connect, retrieve and store data [13]. Using this library, programmers do not need to know where data are stored nor in which database they reside.

System modules

Every stage of the digital content generation workfow is implemented modularly in PHP, using its own object classes and methods. Common methods are available to all modules in a shared external library. Five modules have been developed:

- **Submission module**: This module allows the students to submit their work using a wizard in six steps. Each step uses a particular template to collect data from the student, always guiding them to the next step to avoid incomplete submissions.

- **Reviewing module**: This module is accessible by the graduate office to review submitted contents sent by students. The review only happens after degree homologation and the graduate office must check all data and protect all PDF files. This protection is made automatically by the DLTD/USP system adding a password to avoid content copy and changing the document.

- **Cataloging module**: It is accessible only by library staff to normalize etd's data to Brazilian bibliographic reference standards, to generate a MARC file to feed the institutional library catalog, and to make them accessible in the DLTD/USP public site.

- **Administration module**: This module accesses all workflow stages by the system administrators who can interfere in each stage if needed.

- **Status module**: This module shows information to students about the status of their submitted etds (in the revision or in the cataloging stage).

The Ajax technology has allowed a new innovation in the interaction between the digital content generation system and their users, improving the response time. These tasks are largely executed by users who are not familiar with the digital content generation workflow. Thus, this new system's improvement has brought significant reduction of time in the work made by students, graduate offices and library staffs. Also, there are network traffic and server load reductions because client machines can now run codes previously executed only by servers.

Future work

For a complete integration between the DLTD/USP system and all University's corporate systems, it is necessary to implement a single sign-on (SSO) interface. University of São Paulo has planned to implement SSO this year for all the corporate systems. With this new functionality, we believe that accessing the DLTD/USP system will be easier for all users and the digital content generation workflow will become just a routine in their academic life.

Concluding remarks

Due to the diversity and dimension of the University of São Paulo, finding a content generation workflow to a digital library that minimizes its influence in the existent academic workflow was a hard task. The strong integration and the strict fufillment of responsibilities' policies for information correctness between all involved parts have been suitable in these six years of existence. Training was done with all involved parts to make them aware where the digital library is inserted in their daily routine.

Although all employees were trained, the initial participation was modest because there was not any obligation to deposit an electronic thesis or dissertation in the DLTD/USP. Therefore, until the middle of 2006, the students were invited to participate. Some of them were not sure to make their works available at the Internet due to plagiarism. This fear was also found amongst academic members. However, the reception amongst librarian staffs was big. They understood immediately that the DLTD/USP is a powerful tool to make available all intellectual production of a university.

The integration between the corporate systems with the digital library repository was fundamental because all data are kept consistent and integrated. Thus, the same data retrieved in the Librarian Catalog can also been retrieved in the DLTD/USP system.

The obligation forced by CAPES and followed by the University has increased the number of submissions and consequently the available etds. The Ajax technology used in the new DLTD/USP system has permitted to accelerate this availability improving the interaction between the digital content generation system and its users. The new version of DLTD/USP is ready to attend this demand and also available for any interested institution.

References

[1] DLTD [online]. Digital Library of Theses and Dissertations of University of São Paulo. [2007-05-09]. Available from: <http://www.teses.usp.br>.

[2] BRAZIL. Coordenação de Aperfeiçoamento de Pessoal do Ensino Superior. Institui a divulgação digital das teses e dissertações produzidas pelos programas de doutorado e mestrado reconhecidos. Administrative edict n. 13, February 24th, 2006. [2007-05-09]. Available from: <http://www.capes.gov.br/servicos/legislacao/portarias.html>.

[3] NDLTD [online]. Networked Digital Library of Theses and Dissertations. [2007-05-09]. Available from: <http://www.ndltd.org/>.

[4] MASIERO, P. C., BREMER, C. F., COLETTA, T. das G *et al*. *The University of São Paulo Digital Library o Theses and Dissertations*. Ci. Inf. [online]. 2001, vol. 30, n. 3, pp. 34-41. [2007-05-09]. Available from: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0100-19652001000300005&lng=en&nrm=iso>. ISSN 0100-1965.

[5] JANUS [online]. Graduate Studies Information System of University of São Paulo. [2007-05-09]. Available from: <https://sistemas.usp.br/janus/comum/entrada.jsf>.

[6] DEDALUS [online]. Library Catalog of University of São Paulo. [2007-05-09]. Available from: <http://dedalus.usp.br:4500/ALEPH/POR/USP/USP/USP>.

[7] KONDO, R. T., LIRANI, M. de L. R., ZANI, V. A. T. *et al*. Accelerating with Ajax the Digital Library of Theses and Dissertations of USP. In: II Workshop de Bibliotecas Digitais (XXI Simpósio Brasileiro de Banco de Dados, XX Simpóstio Brasileiro de Engenharia de Software). October 20th, 2006. Florianópolis: Sociedade Brasileira de Computação, 2006.

[8] SYBASE ASE [online]. Adaptative Server Enterprise. [2007-05-09]. Available from: <http://www.sybase.com/products/databasemanagement/adaptiveserverenterprise>.

[9] GARRET, J. J. [online]. *Ajax: A New Approach to Web Application*. Feb. 2005. [2007-05-09]. Available from: <http://www.adaptivepath.com/publications/essays/archives/000385.php>.

[10] O'REILLY, T. *What is the Web 2.o: design patterns and business models for the next generation of software*. Sep. 2005. [2007-05-09]. Available from: <http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html>.

[11] AJAXPATTERNS. The publicly editable repository of all things Ajax. [2007-05-09]. Available from: <http://ajaxpatterns.org>.

[12] XAJAX [online]. Xajax PHP and JavaScript library. [2007-05-09]. Available from: <http://www.xajaxproject.org/>.

[13] PEAR [online]. PHP Extension and Application Repository. [2007-05-09]. Available from: <http://pear.php.net/>.