INDIANA UNIVERSITY
SOUTH BEND

INFORMATICS PLASMA DISPLAY NEWS SERVER PROJECT AND ANALYSIS OF ITS SOFTWARE DEVELOPMENT

A thesis submitted in partial satisfaction of the
requirements for the degree of

MASTER OF SCIENCE

in

APPLIED MATHEMATICS AND COMPUTER SCIENCE

by

Khalid R. Al-asmari

December 2006

The Thesis of Khalid R. Al-asmari
is approved:

Professor Liguo Yu, Advisor

Professor Robert Batzinger

Professor Morteza Shafii-Mousavi

Professor Dana Vrajitoru

Professor David R. Surma
Graduate Director
Applied Mathematics and Computer Science

Abstract

Informatics Plasma Display News Server Project and Analysis of Its Software Development

by

Khalid R. Al-asmari

This project was an attempt to develop the Informatics Plasma Display News Server (IPDNS) used to drive the plasma display operated by the Informatics at Indiana University South Bend. The target software was designed to facilitate the posting of top Informatics and IT news stories from the department, across the campus and around the world.

This thesis describes the implementation and testing of the IPDNS, and analyzes the development processes used for this project.

IPDNS project development process was divided into two phases: 1) analyzing the requirement and designing the system; 2) implementing and testing the system.

In this thesis, the various techniques used for communication and project management are compared and their relative strengths and weaknesses are analyzed in regard to their influence and effect on the software development process.

Dedication

This thesis is dedicated in memory of my father and in honor of my mother. They have supported me all the way since the beginning of my studies.

Acknowledgments

I would like to express my gratitude to my advisor, Dr. Liguo Yu, for many insightful conversations during the development of the ideas in this project, and for helpful comments on the thesis.

I would also like to extend my appreciation to the review committee members: Dr. Robert Batzinger, Dr. Morteza Shafii-Mousavi, and Dr. Dana Vrajitoru for the assistance they provided at all levels of the research project.

I also would like to thank Dr. Robert Batzinger and Mr. Sardar for their contributions as members of the project design team.

Contents

List of Figures

List of Blocks

List of Tables

Chapter 1

Introduction

In March 2005, Indiana University South Bend (IUSB) Informatics acquired a plasma screen to display informatics and IT news to students in a hallway of Northside Hall. Because this display was strategically located to convey information to current and potential students of informatics. It was clear that repeating a single Power Point (PPT) presentation would not achieve this goal. Generating content in the form of PPT files required too much effort to gather, manipulate, and update the content. In addition, if the content was not varied, students would ignore the monitor because the content was considered stale.

In May 2005, a prototype news server was developed in Perl to display the news from a few news feeds on the Internet allowing it to display top world and IT news items as well as the departmental news. The students' response to this system was quite favorable and helped to develop an expectation that the monitor provides interesting news that is worth looking at.

Despite the favorable response to the initial prototype, this news server software required a re-write to overcome a number of limitations and make the system easier to maintain, operate and use.

This project re-engineered the news server to improve its performance, appearance, and maintainability. The update of IPDNS project was done in two phases:

1. **Analyzing the requirements and designing the system.** This work was undertaken by distributed team which used Wiki as the means to communicate, coordinate, and document the proposed project.

2. **Implementing and testing the system.** This work was carried out by the author under the direction of faculty advisors who consulted in frequent face to face meetings.

The difference in communication techniques used in this project provide some interesting insights into the value, weaknesses, and strengths of these techniques in particular programming environments. This thesis will discuss these findings and explore their relevance to software development projects.

The thesis is organized as follows: Chapter 2 contains literature review. Chapter 3 describes the IPDNS project and the various techniques employed to communicate with the various stakeholders of this project. Chapter 4 describes the project implementation and software testing. Chapter 5 summarizes the lessons learned from this project. Conclusions and possible future directions of work are given in Chapter 6.

Chapter 2

Literature Review

2.1 Software Development Environment

Traditionally, most software projects employed centralized software development which is generally performed in a single place. All the developers usually worked in the same city, or even in the same building. Face to face meetings were used to communicate ideas, coordinate efforts, and resolve issues. This form of centralized software development has been successfully used for decades.

However, the development of information and communication technologies has caused the IT industry to change its focus from national market to global markets [10]. This globalization of the software industry is seen as software companies not only sell their products globally, but also begin to develop their products globally. These trends are also seen in the rapid adoption of distributed software development where software development teams work in different geographical locations. In fact, global software development with developers located in different countries or hemispheres has become one of the most significant trends in the software industry during the past decade [4].

The software industry has come to expect the following benefits of distributed software development [3, 13, 15]:

1. Improve the product quality by using top rated international IT professionals.
2. Meet specific customer needs with market customization and localized code.
3. Lower development costs by using services from countries with cheaper cost of living.
4. Lower testing costs by contracting professional functional testing services.
5. Increase the product competitively by expanding the range of languages supported.
6. Standarize the product to allow it to be distributed and supported internationally without change or modification.

As a result, many companies have turned toward globalization to produce high quality software products without increasing labor costs. Approximately 40% of the Fortune 500 companies in USA have engaged in outsourcing to India [5]. Furthermore, more than 50 countries are currently participating in collaborative software development internationally [5].

2.2 Issues in Distributed Software Development

Because developers are located in different cities, time zones, countries, and/or hemispheres, projects using distributed software development experience significant issues in communication, coordination, culture differences, and knowledge management [6, 8, 11, 12, 13], some of which are shown in Table 2.1.

Table 2.1: Issues related to distributed software development

1	**Communication issues**	loss of development speed, lack of informal communication.
2	**Coordination issues**	physical distance issues, and time zone issues.
3	**Cultural issues**	cultural conflicts, incompatible work ethics.
4	**Technical issues**	lack of synchronization and standards.
5	**Knowledge management in distributed organizations**	ineffective information and knowledge-sharing mechanisms, poor documentation.

While many studies have shown that communication, coordination, and exchanges of documents are important for the success of software projects [14], it is hard to update and distribute software documents quickly enough to keep up with the conceptual changes that arise, especially during the design phase of software project. Due to inefficient coordinations between globally dispersed software developers, miscommunication between team members has been one of the major drawbacks of Global Software Development [6]. Of all the issues listed in Table 2.1, communication and coordination appear to be most important.

The following are popular methods used for communication and coordination of global development teams [2].

- **Travel.** Airplanes can bring the team members together, but it is expensive and time consuming. Nevertheless, there is certain information that is best conveyed face to face.

- **Phone.** Phone conversation is a traditional communication technique because it is quick and instantaneous. It has the drawback that the involved developers need to be available at the same time. Hence, it is limited by differences in time zones and work schedules of the various team members. However, a phone call may be more efficient than travel for conveying step-by-step instructions or resolving an urgent issue.

- **Email.** Email is one of the most widely used communication tools for distributed teams. But it is effective only when all team members are committed to reading and responding to email in a timely manner. Similar tools include voice mail and fax machines.

- **Newsgroup.** Threaded discussions in a newsgroup allow multiple developers to contribute to the body of ideas and principles that govern development projects. These discussions are useful for future reference as the project evolves.

- **On-line project management software (PMS).** Members of a team depend on each other but it is nearly impossible to achieve practical levels of accountability without some form of monitoring. The team needs to choose this critical software tool carefully. It is not optional. With an on-line project management software tool it is possible to organize, plan, schedule, and track work assignments. This formal communication method is most effective when it is maintained by the project manager and updated daily by each team member.

- **Content management software (CMS).** In a distributed environment, it is essential that all team members have access to most current copies of working documents. Web-based content management tools, such as Twiki, allow all members to freely create and edit working documents as web content that can be read and changed using a regular web browser. It is a good tool for distributed team members who concurrently work together on the same document, especially during system design specification. Software tools that comply with WebDAV, such as Jakarta Slide, can handle more complex documents, including PDF and Microsoft Word document.

- **Web (video) conference.** Visual contacts within a team help to enhance the feeling of working of comradery among team members. Web (video) conference tools can be used as a media for consultations, presentations, and demonstrations of prototypes and deliverables.

Regardless of methods used, the goal is to create an environment where working in a geographically dispersed team is nearly as effective as working in the same building. Global software development will require people to interact with each other frequently using different methods to overcome the challenges of working decentrally. However, this will require the developer to acquire new skills and become familiar with new tools designed to support software development activities that would usually take place through the direct interaction with people.

The open source software community has been facilitated by the growing number of software packages that address these needs. Table 2.2 is a list of different open source CASE tools that are commonly used.

Table 2.2: Open-source communication CASE tools

Category	**Name**	**Web site**
Project management	Achievo	http://www.achievo.org
	PHProjekt	http://www.phprojekt.com
	Double Choco Latte	http://sourceforge.net/projects/dcl
Content management	wiki	http://twiki.org
	Jakarta Slide	http://jakarta.apache.org/slide
	Plone	http://plone.org
Web conference	Ivisit	http://www.ivisit.com
	Vic	http://www-nrg.ee.lbl.gov/vic
	WebHuddle	https://www.webhuddle.com
Defect tracking	Bugzilla	http://www.bugzilla.org
	Bugzero	http://www.websina.com/bugzero
	Mantis	http://www.mantisbt.org
Version control	CVS	http://www.nongnu.org/cvs
	Subversion	http://subversion.tigis.org

There is a clear demand for easy to use and inexpensive tools and that can support both communication, coordination, and documentation in distributed software development. Wiki is a web-based technology that addresses many of these issues.

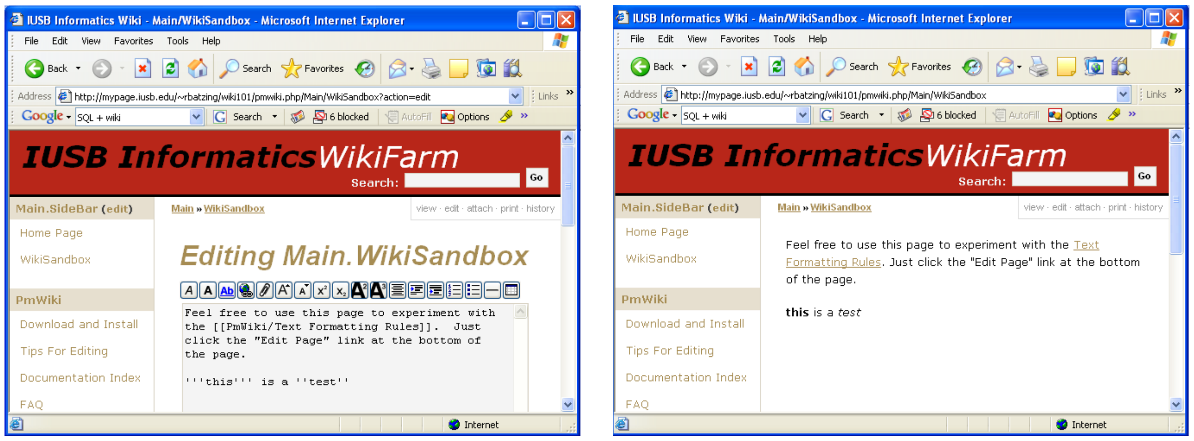
2.3 Wiki

Wiki is a software technology first implemented as WikiWikiWeb by Ward Cunningham[7] to allow users to freely create and edit Web page contents using any Web browser. It supports hyperlinks and has a simple text syntax for creating new pages and editing existing pages on the fly. As shown in Figure 2.1, authors type their ideas as marked plain text in a form window which others will view as formatted text. Each and every change is recorded in history file and the contents are fully searchable within the website as soon as revised pages are posted. These features provide a development team with a platform for posting ideas, concepts, documentation, and source code in a format that is easy to read and modify. Most importantly, Wiki supports open and easy access/modification, providing a great opportunity for collaboration and interaction.

Wiki has inherent mechanisms to protect the contents from malicious altering and to allow user to trace the changes made to individual document. These mechanisms include the following:

1. **Page history**. A record that contains the date and time of every edit, and the user who made it. It is possible to revert the contents of a page back to a

Figure 2.1: Two views of a Wiki page


IUSB Informatics Wiki - Main/WikiSandbox - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://mypage.iusb.edu/~rbatzing/wiki101/pmwiki.php/Main/WikiSandbox?action=edit
Address http://mypage.iusb.edu/~rbatzing/wiki101/pmwiki.php/Main/WikiSandbox
IUSB InformaticsWikiFarm
Search: Go
Main.SideBar (edit)
Home Page
WikiSandbox
PmWiki
Download and Install
Tips For Editing
Documentation Index
FAQ
Main » WikiSandbox
view · edit · attach · print · history
Internet

Editing Main.WikiSandbox

```
Feel free to use this page to experiment with
the [[PmWiki/Text Formatting Rules]].  Just
click the "Edit Page" link at the bottom of
the page.

'''this''' is a ''test''
```

Feel free to use this page to experiment with the Text Formatting Rules. Just click the "Edit Page" link at the bottom of the page.

this is a *test*

editor view formatted view

previous form.

2. **Email notification**. The site manager gets automatically notified when something has changed in a Wiki web site.

3. **Code of conduct**. Shared goals and responsibilities encourage contributions that are positive and unbiased.

4. **Human nature**. Malicious users are outnumbered especially when all changes are labeled with the time and author of the correction.

5. **Password protection**. User authentication can be implemented and used to administer specific rights to the content.

The markup language for hypertext defined by Wiki corresponds to HTML tags. The Wiki server transforms Wiki documents into HTML text strings that can be viewed on any web browser. In addition to the structured text rules, the Wiki software provides the user with links to a collection of tools to view, edit, and interrogate the resulting hypertext. It is also possible to use Wiki to maintain plain ASCII text as found in source code. Table 2.3 shows some of the applications of Wiki that would be useful to global software development projects.

Table 2.3: A few of the many uses for Wiki in software development

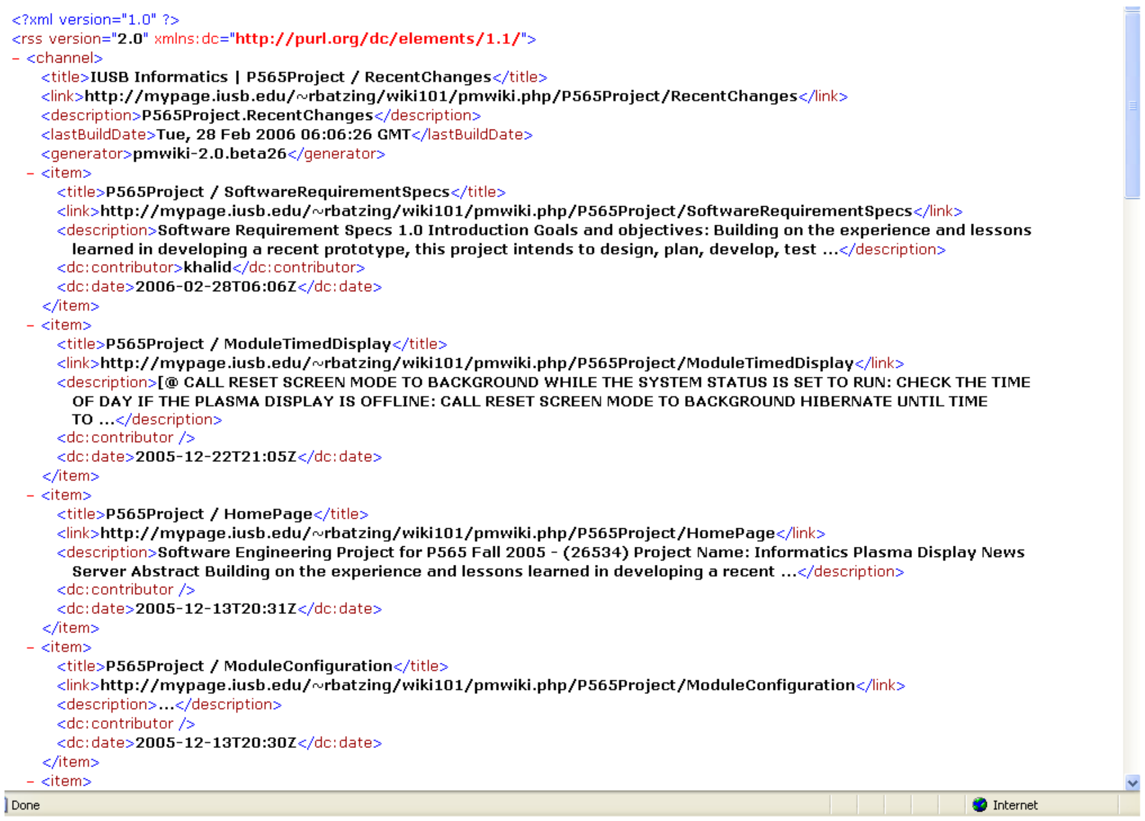
• Recording a dialog between various stakeholders	• To do lists
• Version control of documentation and software	• Task assignments
• Meeting agendas, notes, and handouts	• Document preparation
• Development of FAQ documentation	• Group announcements
• Project plan and related documents	• Interacting with clients
• Status reports	• Contact information

The Wiki technology has continued to develop and has been ported from Perl to other program languages. There are no less than 15 different Wiki packages available from sourceforge.net alone, each with a wide variety of options and supporting tools. In this project PmWiki was chosen because of its ability to alert developers of changes to the site via RSS feeds. This facility converts the Wiki platform from a passive repository to an active message center between developers.

2.4 RSS Technology

RSS (Really Simple Syndication) was developed to provide a method for indexing and consolidating news information. It is a very popular family of XML protocols that have been used by news services, Web blogs, and Podcasting services and the use of RSS continues to grow. The specification for RSS is currently maintained by the RSS Advisory Board [16]. As shown by Figure 2.2, this technology makes it easy to obtain highlights and news from various businesses and organizations. Any news server can easily update its content by downloading RSS files periodically from relevant sources. The content for each news items can be parsed from the XML file and converted into HTML which can be viewed in a web browser.

Figure 2.2: A sample RSS file from the IPDNS Project Wiki



```
<?xml version="1.0" ?>
<rss version="2.0" xmlns:dc="http://purl.org/dc/elements/1.1/">
- <channel>
    <title>IUSB Informatics | P565Project / RecentChanges</title>
    <link>http://mypage.iusb.edu/~rbatzing/wiki101/pmwiki.php/P565Project/RecentChanges</link>
    <description>P565Project.RecentChanges</description>
    <lastBuildDate>Tue, 28 Feb 2006 06:06:26 GMT</lastBuildDate>
    <generator>pmwiki-2.0.beta26</generator>
  - <item>
      <title>P565Project / SoftwareRequirementSpecs</title>
      <link>http://mypage.iusb.edu/~rbatzing/wiki101/pmwiki.php/P565Project/SoftwareRequirementSpecs</link>
      <description>Software Requirement Specs 1.0 Introduction Goals and objectives: Building on the experience and lessons
        learned in developing a recent prototype, this project intends to design, plan, develop, test ...</description>
      <dc:contributor>khalid</dc:contributor>
      <dc:date>2006-02-28T06:06Z</dc:date>
    </item>
  - <item>
      <title>P565Project / ModuleTimedDisplay</title>
      <link>http://mypage.iusb.edu/~rbatzing/wiki101/pmwiki.php/P565Project/ModuleTimedDisplay</link>
      <description>[@ CALL RESET SCREEN MODE TO BACKGROUND WHILE THE SYSTEM STATUS IS SET TO RUN: CHECK THE TIME
        OF DAY IF THE PLASMA DISPLAY IS OFFLINE: CALL RESET SCREEN MODE TO BACKGROUND HIBERNATE UNTIL TIME
        TO ...</description>
      <dc:contributor />
      <dc:date>2005-12-22T21:05Z</dc:date>
    </item>
  - <item>
      <title>P565Project / HomePage</title>
      <link>http://mypage.iusb.edu/~rbatzing/wiki101/pmwiki.php/P565Project/HomePage</link>
      <description>Software Engineering Project for P565 Fall 2005 - (26534) Project Name: Informatics Plasma Display News
        Server Abstract Building on the experience and lessons learned in developing a recent ...</description>
      <dc:contributor />
      <dc:date>2005-12-13T20:31Z</dc:date>
    </item>
  - <item>
      <title>P565Project / ModuleConfiguration</title>
      <link>http://mypage.iusb.edu/~rbatzing/wiki101/pmwiki.php/P565Project/ModuleConfiguration</link>
      <description>...</description>
      <dc:contributor />
      <dc:date>2005-12-13T20:30Z</dc:date>
    </item>
  - <item>
```

Chapter 3

Project Description

3.1 Overview and the Requirement

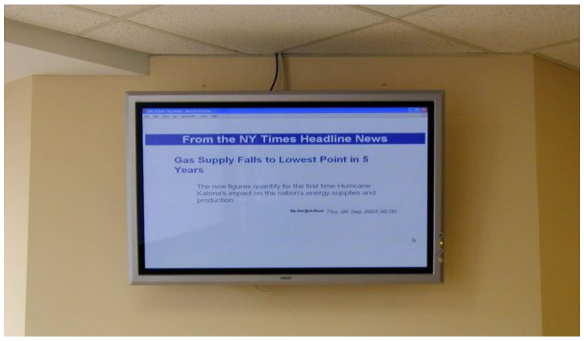
The Informatics Plasma Display News Server (IPDNS) was a project intended to develop a news server to drive the plasma display operated by Indiana University South Bend Informatics (Figure 3.1). The target software was expected to facilitate the posting of top Informatics and IT news stories from the department, across the campus and around the world.

The requirements of the project were developed during conversation with Dr. Ruth Schwartz, the head of IUSB Informatics. The main points are summarized below:

1. Improve the appearance and readability of the news screens:

    - Change color schemes according to news topics.

Figure 3.1: Plasma display


From the NY Times Headline News
Gas Supply Falls to Lowest Point in 5 Years

- Change the timing of displayed pages to better suit the ability of readers to read through the body of a message.
- Support accented characters that commonly occur in news feeds such as SlashDot and NIH.
- Include pictures that are referenced by certain news feeds.
- Remove the display of URL links from the contents of news pages.
- Support the display of long news bits so that they do not run off the screen.

2. Improve the installation and maintenance of the news service:

   - Automate the installation of the software on a new system.
   - Simplify the submission of PPT files and related information to the system.
   - Simplify the updating of the information in the RSS feed list.
   - Support for displaying PDF files in the fullscreen scrolling mode.
   - Provide extensive documentations of the news service.

3. Improve the effectiveness of the news service:
   - Shorten the number of news items shown in the main news program loop.
   - Make IUSB Informatics branding more prominent with various transition screens that identify the sponsor of this service.
   - Develop a news category that spotlights IUSB Informatics activities and programs.
4. Improve the organization of the news programming into specific categories:
   - Headline news: eg., NY Times, Washington Post, BBC.
   - Technology news: eg., NY Times, Washington Post, BBC, Harvard Business Review.
   - Technology highlights: eg., InfoWorld, NIH, NAS.
   - IT professional news: eg., SlashDot, ACM, InfoWorld, Linux forum.
   - Campus news: eg., UCET, Library, Campus webpage.
   - Departmental news: eg., Informatics/CIS department announcements.
   - Weather: eg., US weather bureau, Local weather site.
   - Special feature: eg., tutorials, spotlight on some aspects of the department.
5. Use Microsoft Agent to broadcast special announcements.

3.2 The Development Tools

The developers agreed to use a common set of software tools to simplify development. Most of these software were downloaded from the Internet and used under open source licenses. The software tools used are listed in Table 3.1.

Table 3.1: Softwares used in the IPDNS Project

Software Categories	**Tool**	**Description**
Software Design Tools	Graphviz	A graph drawing tool.
	Visual Paradigm	A UML tool.
Programming Environment	Active Perl	A Systems software development platform.
Datebase Manager	SQLite	A database engine.
Document Browser and Display	Adobe Acrobat Reader	A PDF file browser.
	Firefox Web Browser	A HTML browser.
	Microsoft PowerPoint Viewer	A PPT file viewer.
	MS Agent	A MS character viewer.
Software Documentation	PmWiki	An Interactive web authoring tool.
	AAA Logo	A logo design tool.
Project Management Software	Trac	A project management tool for monitoring tasks and issues.

3.3 The Development Process

This project was developed under two phases: Phase 1 focused on the analysis of requirements and software design; and Phase 2 was concerned with software implementation and testing.

Phase 1 was undertaken as part of a graduate software engineering group project. Three developers interviewed the primary client to establish the goals and parameters of the project. Feedback and suggestions were provided by the instructor mentor. The distributed software development model was used for this phase of the work because the developers were unable to meet face to face regularly. All documents were drafted and developed online. The final documentation was printed from the online material after all the developers had an opportunity to edit and revise the text. Final approval for each document was given in group meetings.

A Wiki site[1] was established for the project and was used as a platform for communication and coordination as concepts, documents and software were developed. Draft ideas and documents were posted for review and comment. Various indices and RSS feed services were activated on this site to make it easier to identify new materials to be reviewed.

As this was the first time this development team had worked in a distributed software development enviroment, it was important that other communication methods, such as email and phone, were used to overcome miscommunications that arose when issues in the Wiki text were not being addressed. In addition, the project used the Trac software[2] to monitor and manage various mission critical tasks in a timely manner.

[1]http://mypage.iusb.edu/∼ rbatzing/wiki101/pmwiki.php/P565Project /HomePage
[2]http://www.edgewall.com/trac/

Phase 2 was primarily carried out by the programmer under the direction of a coadvisor. The main task was to implement the design and test the final software product. Because main concerns for this phase were related to understanding the Perl language and resolving technical issues, face to face meetings were the major communication and coordination method used. Face to face meetings proved to be more effective than Wiki and other communication methods at this phase, because it was easier to discuss technical problems orally and demonstrate specific features than to describe them in writing.

Chapter 4

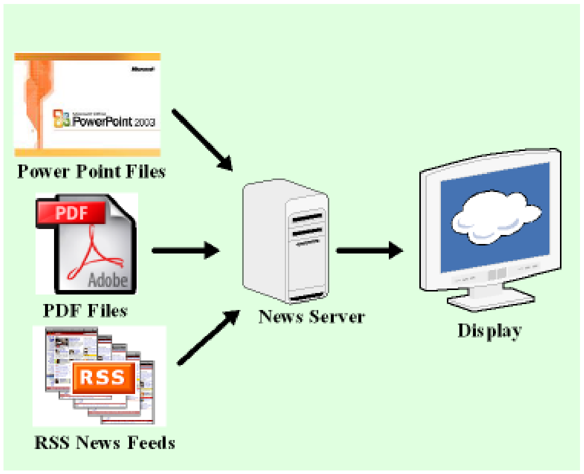
Project Implementation and Testing

4.1 Project Structure

As shown in Figure 4.1, the news server was designed to manage and display news resources regardless of their format. The output of the server monitor was mirrored to the plasma monitor using a video Y-cable connection.

The news server implemented by this project consists of three basic modules; Interface Module, Formatting Module, and Display Module. All of these modules were developed to run concurrently under the Windows XP environment. The functions of each module are given in Table 4.1

Figure 4.1: Basic operation



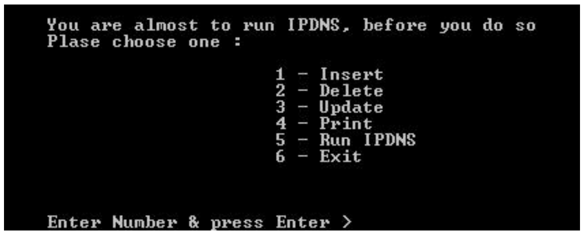
4.2 Implementation Details

Perl version 5.8.8 was used to implement the three modules described above. These modules shared a common database managed by SQLite version 3.3.5. The design and structure of this database was based on the IPDNS Project Design Specification [1]. The final database structure is shown in Appendix A. Subroutines common to two or more module were pooled into a source file that was included at run time(Table 4.2).

4.2.1 Interface Module

For this version of the news server, a prototype interface module was written as a command line application for the purpose of building the databases and testing the integrity of the database design. The basic options to the user of this menu-driven module are shown in Figure 4.2.

Figure 4.2: The main interface menu



```
You are almost to run IPDNS, before you do so
Plase choose one :

                        1 - Insert
                        2 - Delete
                        3 - Update
                        4 - Print
                        5 - Run IPDNS
                        6 - Exit


Enter Number & press Enter >
```

In order to quickly build the interface to the database needed by other modules, subroutines were written using Perl Database Interface (DBI) to support the four basic database operations: insert, delete, update, and retrieve. As the main requirement for this module was the speed of development, no attempt was made to make this a user-friendly application. The menus for these operations are given in Table 4.3. Each of these operations provided a simple but robust interface to the five tables within the database.

4.2.2 Formatting Module

Block 4.1 shows the source code for the formatting module. This module was designed to convert RSS feeds into a sequence of HTML pages that could be easily displayed. It was also used to update the status of each news resource to determine whether its content could be displayed depending on whether the resource exists and is scheduled for current display.

The extraction of content from RSS files required implementing a parser to identify the individual news items and a translator to render the text in HTML. Because the display module also interacted concurrently with the HTML files corresponding to an RSS feed, it was critical to implement a mechanism which would prevent any attempt by two or more modules to access these files concurrently in a variety of read, write, or delete modes. This was achieved by forcing the Formatting Module to use a separate working directory. The database was used to identify which working directory contained files for display and which one could be used to develop new material to be displayed later.

4.2.3 Display Module

The display Module was designed primarily as a time keeper. Blocks 4.2, and 4.3 show the subroutine **run_ipdns()** which contains the main function of Display Module. The first task of this module was to check whether the current time was within the range of normal hours of operation. Its main responsibility was to display each news item long enough to be read and then replace it with the next news item on the program. The news server was given the ability to display three types of files–PDF, HTML, and PPT, by calling the appropriate browser via the operating system. (The corresponding lines of codes for the specific version of the subroutine **display()** which show these display types can be seen in Blocks 4.4, 4.5, and 4.6, respectively.) The display module was also designed to request the operating system to kill the process related to the browser when the display time for a news item has expired.

In the case of RSS files this module was programmed to work directly with HTML rended version of the content. The code for this can be found in Block 4.7.

4.3 Testing

The following sections describe the three methods that were used to help identify and locate problems in the Perl code and to improve the performance of the modules.

4.3.1 Print Statement

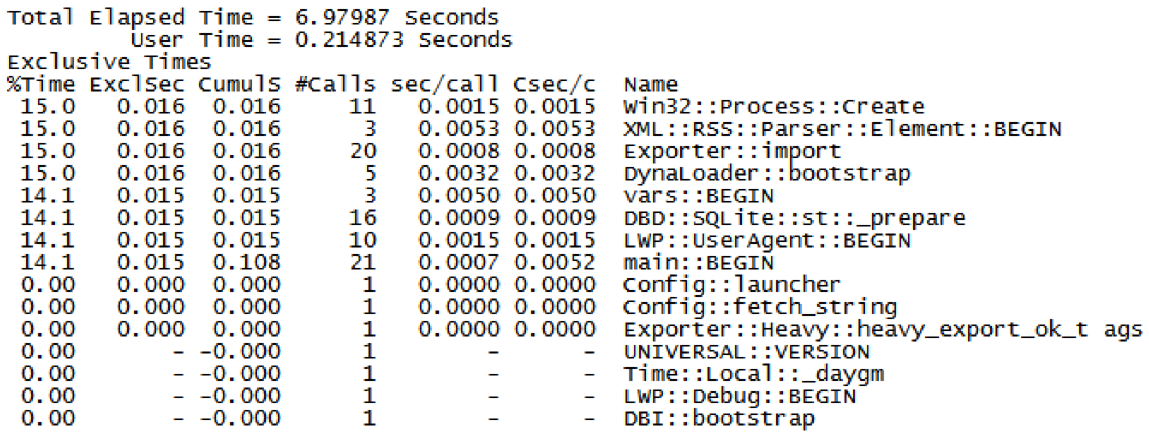
The operation of each module was verified by adding print statements to display the input and output of each subroutine. This debugging technique was used to monitor the effects of code execution and identify potential problems. At times, this was the easiest and fastest way to figure out problems and validate the algorithms used in the program. Most of the errors and bugs were detected by this technique.

4.3.2 Perl Debugger

Each module was tested in Perl Debugging Mode which identified common errors of programming with warning messages. This debugging technique was easy to use [9] and was effective in highlighting a number of bugs that escaped earlier attempts to test the software. For example, there were several incidences of variables used before they were assigned a value. These errors were caught by this debugging technique.

4.3.3 Profiling

Figure 4.3: Project profile data before optimizing

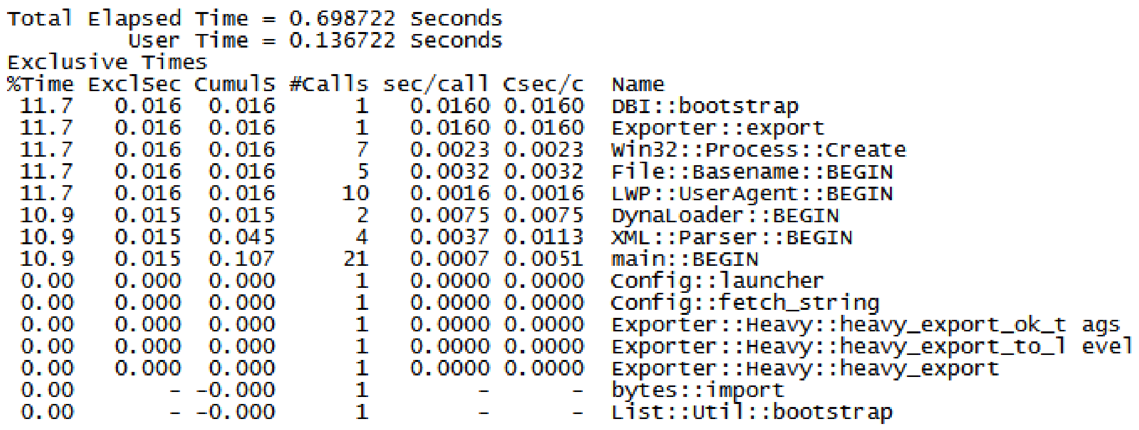


```
Total Elapsed Time = 6.97987 Seconds
         User Time = 0.214873 Seconds
Exclusive Times
%Time ExclSec CumulS #Calls sec/call Csec/c  Name
 15.0   0.016  0.016     11   0.0015 0.0015  Win32::Process::Create
 15.0   0.016  0.016      3   0.0053 0.0053  XML::RSS::Parser::Element::BEGIN
 15.0   0.016  0.016     20   0.0008 0.0008  Exporter::import
 15.0   0.016  0.016      5   0.0032 0.0032  DynaLoader::bootstrap
 14.1   0.015  0.015      3   0.0050 0.0050  vars::BEGIN
 14.1   0.015  0.015     16   0.0009 0.0009  DBD::SQLite::st::_prepare
 14.1   0.015  0.015     10   0.0015 0.0015  LWP::UserAgent::BEGIN
 14.1   0.015  0.108     21   0.0007 0.0052  main::BEGIN
 0.00   0.000  0.000      1   0.0000 0.0000  Config::launcher
 0.00   0.000  0.000      1   0.0000 0.0000  Config::fetch_string
 0.00   0.000  0.000      1   0.0000 0.0000  Exporter::Heavy::heavy_export_ok_t ags
 0.00       - -0.000      1        -      -  UNIVERSAL::VERSION
 0.00       - -0.000      1        -      -  Time::Local::_daygm
 0.00       - -0.000      1        -      -  LWP::Debug::BEGIN
 0.00       - -0.000      1        -      -  DBI::bootstrap
```

Profiling studies were carried out using the Perl Profiler. Figure 4.3 shows a sample output of a time profile for an early version of 5.6 Module.

The first line indicates that the program finished in 6.98 seconds after it was started and the second line estimates the time the code would have taken if it was the only process running on the machine, (0.21 seconds). For the remaining 0.56 seconds, the CPU was working on other tasks. By studying the profiling of the program (Figure 4.3), it was determined that the program calls a number of subroutines and it was possible to determine which subroutines were using the most time. By restructuring the program, the speed of the program increased to the point that the number of measured subroutine calls fell from 96 to 58 (Figure 4.4). Total time spent by the program is reduced to 0.14 sec. These results confirmed that the news server

Figure 4.4: Project profile data after optimizing



```
Total Elapsed Time = 0.698722 Seconds
         User Time = 0.136722 Seconds
Exclusive Times
%Time ExclSec CumulS #Calls sec/call Csec/c  Name
 11.7   0.016  0.016      1   0.0160 0.0160  DBI::bootstrap
 11.7   0.016  0.016      1   0.0160 0.0160  Exporter::export
 11.7   0.016  0.016      7   0.0023 0.0023  Win32::Process::Create
 11.7   0.016  0.016      5   0.0032 0.0032  File::Basename::BEGIN
 11.7   0.016  0.016     10   0.0016 0.0016  LWP::UserAgent::BEGIN
 10.9   0.015  0.015      2   0.0075 0.0075  DynaLoader::BEGIN
 10.9   0.015  0.045      4   0.0037 0.0113  XML::Parser::BEGIN
 10.9   0.015  0.107     21   0.0007 0.0051  main::BEGIN
 0.00   0.000  0.000      1   0.0000 0.0000  Config::launcher
 0.00   0.000  0.000      1   0.0000 0.0000  Config::fetch_string
 0.00   0.000  0.000      1   0.0000 0.0000  Exporter::Heavy::heavy_export_ok_t ags
 0.00   0.000  0.000      1   0.0000 0.0000  Exporter::Heavy::heavy_export_to_l evel
 0.00   0.000  0.000      1   0.0000 0.0000  Exporter::Heavy::heavy_export
 0.00       - -0.000      1        -      -  bytes::import
 0.00       - -0.000      1        -      -  List::Util::bootstrap
```

was suitably optimized for normal operation where screens are displayed for 10 to 15 sec.

The meaning of each column in Figure 4.3 and 4.4 is explained below:

`%Time` : Percentage of time of interrupts to the corresponding routine.

`#Calls` : Number of calls to this routine.

`sec/call` : Average number of seconds per call to this routine.

`Name` : Name of routine.

`CumulS` : Time (in seconds) spent in this routine and routines called from it.

`ExclSec` : Time (in seconds) spent in this routine (not including those called from it).

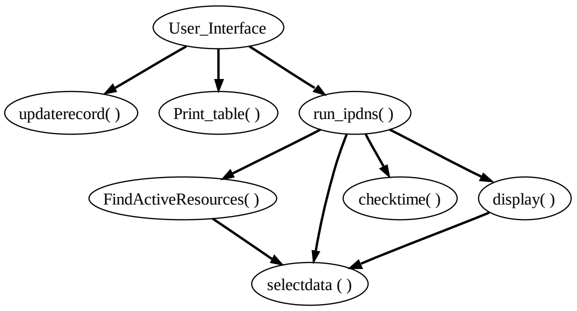
`Csec/c` : Average time (in seconds) spent in each call of this routine (including those called from it).

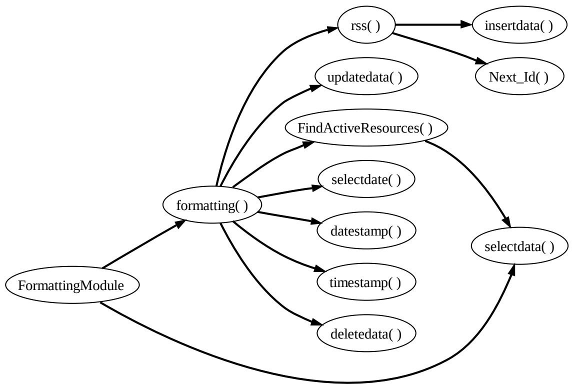
Table 4.1: The basic functions of each module

Module	**Functions**
Interface	• Insert records to the database. • Delete records from the database. • Update records to the database. • Print out the database tables. • Run the IPDNS Display Module. • Exit the program.
Formatting	• Reset the list of available resources. • Get a list of all news categories and their attributes. • Check for current resources files for each news category. • Update any expired RSS feeds for each news category. • Convert updated RSS feeds into HTML files. • Make a list of resource files to be displayed. • Calculate and annotate the required display time for each news resources to be displayed.
Display	• Hiberate if the current time represents an off-hour time period. • Pick the next news item from the lists of the resources to be displayed. • Determine the format of the news item. • Load and execute appropriate file viewer. • Wait until the required display time expired. • Remove the display process and its related window.

Table 4.2: Subroutines called by each Module

Module	Subroutines

Interface	 User_Interface
	updaterecord( )
	Print_table( )
	run_ipdns( )
	FindActiveResources( )
	checktime( )
	display( )
	selectdata ( )

Formatting	 rss( )
	insertdata( )
	updatedata( )
	Next_Id( )
	FindActiveResources( )
	selectdate( )
	formatting( )
	datestamp( )
	selectdata( )
	timestamp( )
	FormattingModule
	deletedata( )

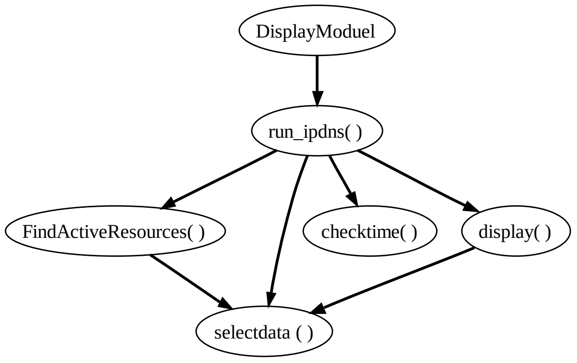
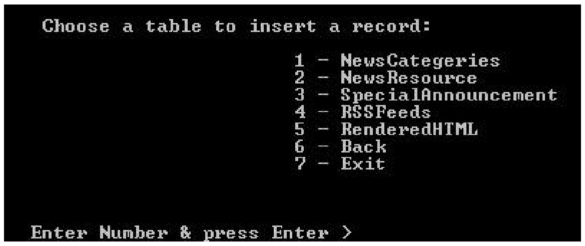
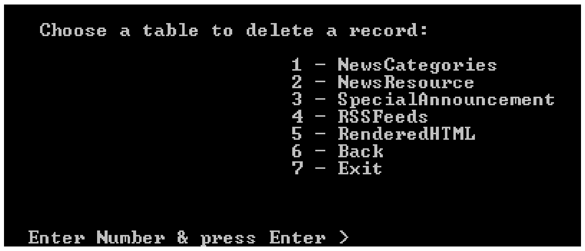
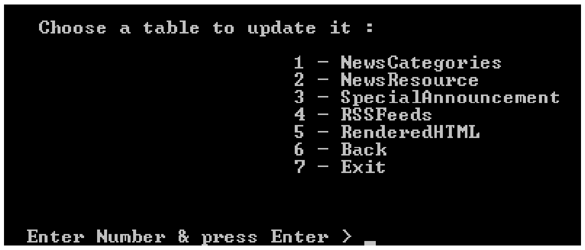
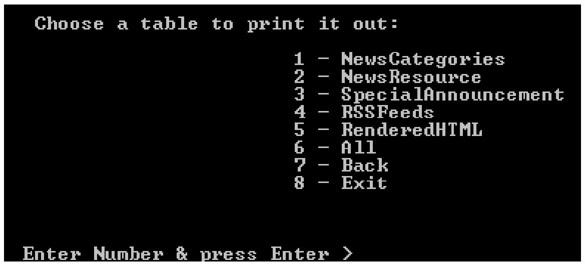
Display	 DisplayModuel
	run_ipdns( )
	FindActiveResources( )
	checktime( )
	display( )
	selectdata ( )

Table 4.3: All interface operations

Operation	Menu
Insert	 ```Choose a table to insert a record: 1 - NewsCategeries 2 - NewsResource 3 - SpecialAnnouncement 4 - RSSFeeds 5 - RenderedHTML 6 - Back 7 - Exit Enter Number & press Enter >```
Delete	 ```Choose a table to delete a record: 1 - NewsCategories 2 - NewsResource 3 - SpecialAnnouncement 4 - RSSFeeds 5 - RenderedHTML 6 - Back 7 - Exit Enter Number & press Enter >```
Update	 ```Choose a table to update it : 1 - NewsCategories 2 - NewsResource 3 - SpecialAnnouncement 4 - RSSFeeds 5 - RenderedHTML 6 - Back 7 - Exit Enter Number & press Enter > _```
Print	 ```Choose a table to print it out: 1 - NewsCategories 2 - NewsResource 3 - SpecialAnnouncement 4 - RSSFeeds 5 - RenderedHTML 6 - All 7 - Back 8 - Exit Enter Number & press Enter >```

Block 4.1 Formatting module

```
 sub formatting
{
  # Update status to unavailable for all expired and premature
  # resources which are marked available
  my $dateunit = datestamp();
  my $timeunit = timestamp();

  $status = updatedata("master_project_db","NewsResource",
            "DisplayExpiryDate < '$dateunit' OR '$dateunit' < DisplayStartDate","Status = 0");
  $status = updatedata("master_project_db","NewsResource",
             "DisplayExpiryDate > '$dateunit' AND DisplayStartDate <= '$dateunit'","Status = 1");

   # Create a list of resources that should be active
   my @activeresources = FindActiveResources;
   my @activeresource = FindActiveResources;

   while (@activeresources)
     {
      my $resource = shift(@activeresources);
      # QUERY FOR ALL CURRENT NEWS RESOURCES
      @attributes = selectdata('master_project_db','NewsResource',
                              "*","ResId == '" . $resource ."'");
      $alist = $attributes[0];

      # CASE NON RSS
      if ($alist->[7] != 1)
       {
         # check if the file exists
         if (!(-e $alist->[2] . "/" . $alist->[1]))
                   {
                   # reset the status to 0 if not
                   $status = updatedata('master_project_db', 'NewsResource',_
                                          "ResId == " . $alist->[0],"Status = 0");
                   }
       }
      else
       {
          $rlist = (selectdata('master_project_db','RSSFeeds',"*","RSSId == " . $alist->[0]))[0];

          #if RSS file is out of date
          if (($rlist->[7] <= datestamp()) || ($rlist->[8] + $rlist->[3]  < timestamp()))
             {
              #Attempt to download new copy of RSS feeds
               $content = fetchrss($rlist->[4],$rlist->[6]);

              #if failed reset the Status back to 0
                   if (length($content) < 10)
                      {
                          $status = updatedata("master_project_db", "NewsResource",
                                               "ResId == " . $alist->[0], "Status = 0");
                      }
                   else
                      {
                         if($rlist->[9] eq 'A')_
                              {
                                 $workingDirectory = 'B';
                              }
                               else
                              {
                                 $workingDirectory = 'A';
                              }

                        @results = deletedata('master_project_db','RenderedHTML',"*",
                                  "RSSFeed == '$alist->[0]' AND Directory == '$workingDirectory'");
                      #parse rss into HTML in the working directory
                      rss($alist->[2],$alist->[1],$workingDirectory,$rlist->[0],$rlist->[6],_
                            $rlist->[5],$alist->[0], $content);
                        #change the value of the working directory, Rss time/date stamps
                        $status = updatedata("master_project_db", "RSSFeeds","RSSId == " . $rlist->[0],
                                               "DateRefreshed = '" . datestamp() .
                                              "', TimeRefreshed = '" . timestamp() .
                                             "', ActiveDirectory = '" . $workingDirectory . "'");_
                     }#end else
           }#end if updata rss
      }#end else
   }#end While

   while (@activeresource)
    {
      my $update = shift(@activeresource);
      $updateDir = updatedata("master_project_db", "NewsResource","ResId == " . $update    ,_
                              "Dir = '" . $workingDirectory . "'");
    }
  }
```

Block 4.2 Timed-display module

```
sub run_ipdns
 {
  print "Current time:", checktime(), "\n";
  # update base time to check  file age correctly
  $^T = time();
  # running format module
  Win32::Process::Create( $ProcessObj,"C:\\Perl\\bin\\perl5.8.8.exe",
              "perl5.8.8 c:\\format_test.pl",
              0,NORMAL_PRIORITY_CLASS,
              ".")|| die ErrorReport();
  sub ErrorReport{print Win32::FormatMessage( Win32::GetLastError());}
  # to shotdown the system
  Win32::Process::Create( $ProcessObj,"C:\\Perl\\bin\\perl5.8.8.exe",
              "perl5.8.8 c:\\configration.pl",
              0,NORMAL_PRIORITY_CLASS,
              ".")|| die ErrorReport();

  while(1)
    {
      # go to sleep at the end of day
      if ((checktime() < 800 ) || (checktime() > 2135))
    {
          print "Go to sleep", checktime(), "\n";
    }
      # display the news during the day time
      else
    { # to select the news randomly
      my $range = 3;
      my $minimum = 1;
      my $random_number;
      my $next;
          # to find the active resources
      my @activeresources = FindActiveResources();
      my $size = @activeresources;
```

Block 4.3 Timed-display module (continued)

```
       while (@activeresources &&   $minimum < $size )
         {
           my $resource = shift(@activeresources);
               # generating random number
           $random_number = int(rand($range)) + $minimum;
               # to prevent from displaying the same news next time
               # in the same period
           $minimum =  $random_number + 1;
               # to select data
           @attributes = selectdata('master_project_db','NewsResource',
               "*","ResId == '" . $minimum ."'");
           # save it in an array
           $newslist =  $attributes[0];
           # call display subroutine to display the news
           display("$newslist->[0]","$newslist->[1]","$newslist->[2]",
               "$newslist->[3]","$newslist->[4]","$newslist->[5]",
               "$newslist->[6]","$newslist->[7]","$newslist->[8]",
               "$newslist->[9]");
         }#End While(@acti..)
     }# End Else
       # to shot down the system
       @shutdown = selectdata('master_project_db','Configuration',"*",
             "ID == 1");
       $shutlist = $shutdown[0];
       if ( $shutlist->[2] == 1 )
     { # killing any Firefox viewer still open
       kill 9, @kill_pid_html;
       exit 0;
     }
     } # End While(1)
```

Block 4.4 Displaying PDF files

```
if( $ResourceType == 2 )
   { # rand(number of pdf files)
     $random = int(rand(1))+1;
     # create a process to display pdf file
     Win32::Process::Create($ProcessObj,"C:\\Program Files\\Adobe\\
                Acrobat 7.0\\Reader\\AcroRd32.exe",
                "AcroRd32  $Directory\\" . $random . ".pdf",
                0,NORMAL_PRIORITY_CLASS,
                ".")|| die ErrorReport();
     # killing all html running file
     kill 9, @kill_pid_html;
     # if the display run time greater than 40 sec.
     # run the special announcment
     if($DisplayRunTime>=40)
       {# create a process to run specialannouncement.pl file
     Win32::Process::Create( $ProcessObj,"C:\\Perl\\bin\\perl5.8.8.exe",
                 "perl5.8.8 c:\\specialannouncement.pl",
                 0,NORMAL_PRIORITY_CLASS,
                 ".")|| die ErrorReport();
       }
     # wait for display
     sleep($DisplayRunTime) if $debug;
     # killing pdf running file
     $ProcessObj->Kill(0);
   }#End of pdf
```

Block 4.5 Displaying HTML files

```
if( $ResourceType == 3 )
  { # rand(number of html files)
    $random = int(rand(1))+1;
    # create a process to display html file
    Win32::Process::Create($ProcessObj,"C:\\Program Files\\
                Mozilla Firefox\\firefox.exe",
               "firefox   $Directory\\" . $random . ".htm",
               0,NORMAL_PRIORITY_CLASS,
               ".")|| die ErrorReport();
    # if the display run time greater than 40 sec. run the special
    # announcment
    if($DisplayRunTime>=40)
      {# create a process to run specialannouncement.pl file
    Win32::Process::Create( $ProcessObj,"C:\\Perl\\bin\\perl5.8.8.exe",
                "perl5.8.8 c:\\specialannouncement.pl",
                0,NORMAL_PRIORITY_CLASS,
                ".")|| die ErrorReport();
      }
    # wait for display
    sleep($DisplayRunTime)  if $debug;
    # find out the process ID
    $pid = $ProcessObj->GetProcessID();
    # pushing the process ID into an array
    push(@kill_pid_html,$pid);
  }#End of html
```

Block 4.6 Displaying PPT files

```
if($ResourceType == 4 )
  { # rand(number of ppt files)
    $random = int(rand(2))+1;
    # create a process to run ppt file
    Win32::Process::Create( $ProcessObj,"C:\\Program Files\\
                 Microsoft Office\\PowerPoint Viewer\\
                 PPTVIEW.EXE","PPTVIEW  $Directory\\"
                 . $random . ".ppt",0,NORMAL_PRIORITY_CLASS,
                ".")|| die ErrorReport();
    # killing all html running file
    kill 9, @kill_pid_html;
    # if the display run time greater than 40 sec. run the
    # special announcment
    if($DisplayRunTime>=40)
      {# create a process to run specialannouncement.pl file
    Win32::Process::Create( $ProcessObj,"C:\\Perl\\bin\\perl5.8.8.exe",
                "perl5.8.8 c:\\specialannouncement.pl",
                0,NORMAL_PRIORITY_CLASS,
                ".")|| die ErrorReport();
      }
    # wait for display
    sleep($DisplayRunTime) if $debug;
    # killing ppt running file
    $ProcessObj->Kill(0);
  }#End of ppt
```

Block 4.7 Displaying RSS files

```
sub display {
 local($ResId,$Dir,$Directory,$Category,$DisplayStartDate,
       $DisplayExpiryDate,$DisplayRunTime,$ResourceType,
       $Link,$Status) = @_;
    # to handel htm file from RSS folder
  if( $ResourceType == 1 )
   { # rand(number of rss files)
     $random = int(rand(2))+1;
     # to fetch the display run time
     $Hlist = (selectdata('master_project_db','RenderedHTML',
               "*","RSSFeed == '$ResId' AND  FileNumber ==
               '$random' AND Directory =='$Dir'"))[0];
     # create a process to display html file
     Win32::Process::Create($ProcessObj,"C:\\Program Files\\
                 Mozilla Firefox\\firefox.exe",
                "firefox C:\\$Dir\\$Directory\\" . $random . ".htm",
                0,NORMAL_PRIORITY_CLASS,
                ".")|| die ErrorReport();
     # if the display run time greater than 40 sec. run the special
     # announcement
     if($Hlist->[4]>=40)
       { # create a process to run specialannouncement.pl file
     Win32::Process::Create($ProcessObj,"C:\\Perl\\bin\\perl5.8.8.exe",
                "perl5.8.8 c:\\specialannouncement.pl",
                0,NORMAL_PRIORITY_CLASS,
                ".")|| die ErrorReport();
       }
     # wait for display
     sleep($Hlist->[4]) if $debug;
     # find out the process ID
     $pid = $ProcessObj->GetProcessID();
     # pushing the process ID into an array
     push(@kill_pid_html,$pid);
   }#End of RSS
```

Chapter 5

Experience and Knowledge

It is recognized that the IPDNS project was a small project and had too few participants to be able to draw statistically significant conclusions about the efficiency of the programming environment used. Wiki technology has become increasingly popular with open source projects in which dozens of programmers from around the world collaborate to develop applications. These developments are excellent sources of information about the behavior of programmers within Wiki environments. The section that follows will attempt to compare the experience of the IPDNS project against that of the open source developers of Wikimedia.

5.1 Wikimedia

Wikipedia was created in 2001 by Larry Sanger and Jimmy Wales and has become a popular multi-lingual, Web-based encyclopedia. It was launched as an English language project, using a Wiki technology to collect and serve information. The entire content of the Wikipedia is encoded and maintained as a Wiki database by volunteers, allowing most articles to be changed by almost anyone with access to the Wikipedia web site.

The Wiki technology used to support Wikipedia is actually a product of the Wikimedia Group which has grown to 56 volunteer developers. The activities and submissions of each and every Wikimedia developer are recorded online [18]. These data are useful for studying the performance of volunteer developers on an international open-source project of this size.

Figure 5.1: The line represents the number of lines of code corrected or added to the system [18]

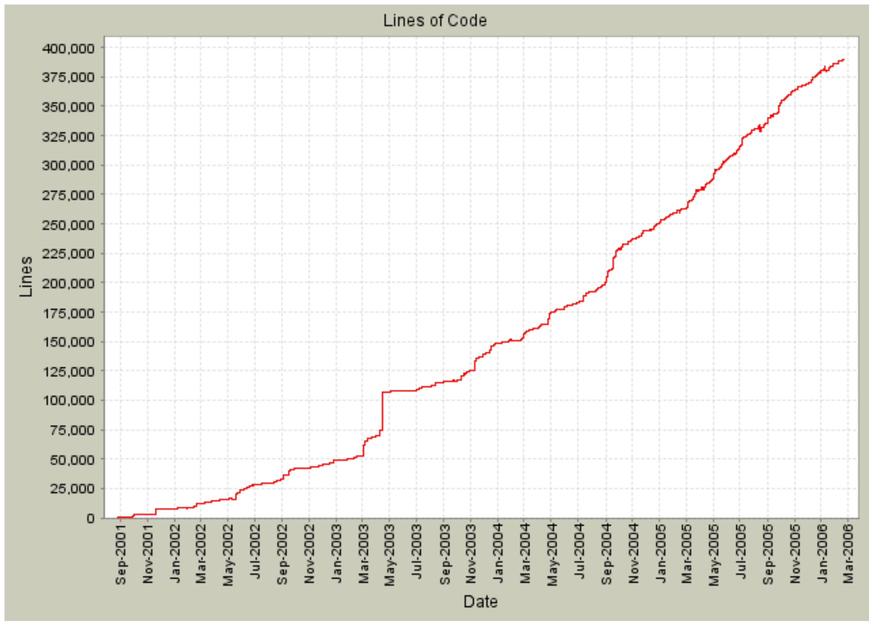


Figure 5.1 shows the rate of development of the Wikimedia. This software development program has been successful as seen by the fact that the software has

continued to grow an increasing rate ever since its launch in 2001.

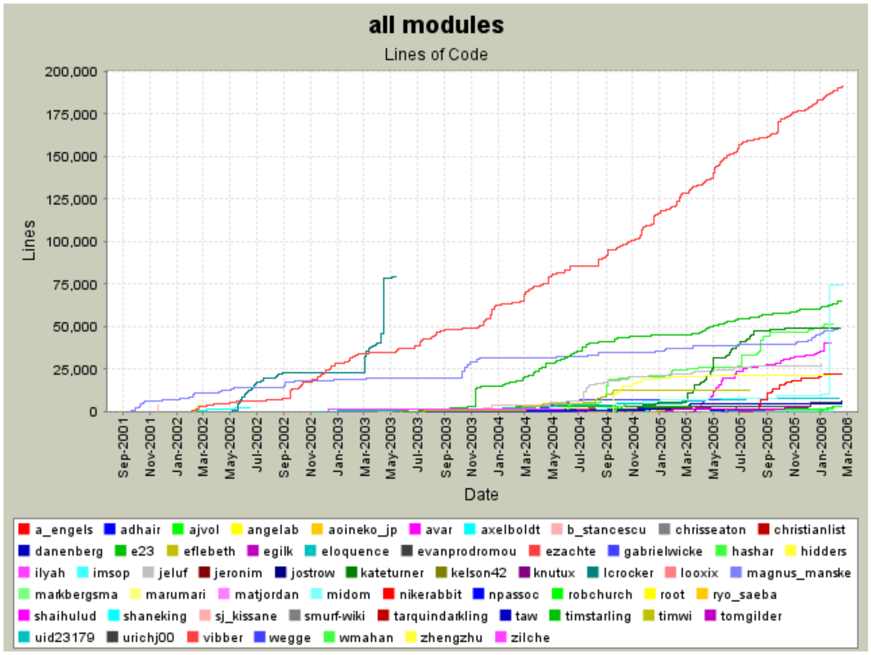
Figure 5.2 shows the effort of developers against the growth of the software. This chart illustrates major changes to large-scale project, i.e. the growing number of participants as the project matures. From this chart it can be seen that some programmers started early in the project but the majority have jointed the project later. Because a dedicated Wiki was used to track and maintain the documentation, source code and history of the project, newcomers to the project are able to orient themselves to the project and contribute to the software development effort fairly quickly. Most participants are able to attain their maximum rate of Lines Of Code (LOC) submissions within 2 months of joining the project. As Wiki technology has been instrumental in managing the increasing complexity of the Wikimedia project and providing support for the growing number of developers, data from their online log files provide useful insights to the effectiveness of this environment beyond what was seen in the IPDNS project.

Table 5.1: Effect of experience on the size of changes submitted

Total	LOC per change		
LOC	Less than 50	Greater than 50	total
Greater than 7000	12	2	14
236 - 7000	19	9	28
Less than 236	13	1	14
Total	44	12	56

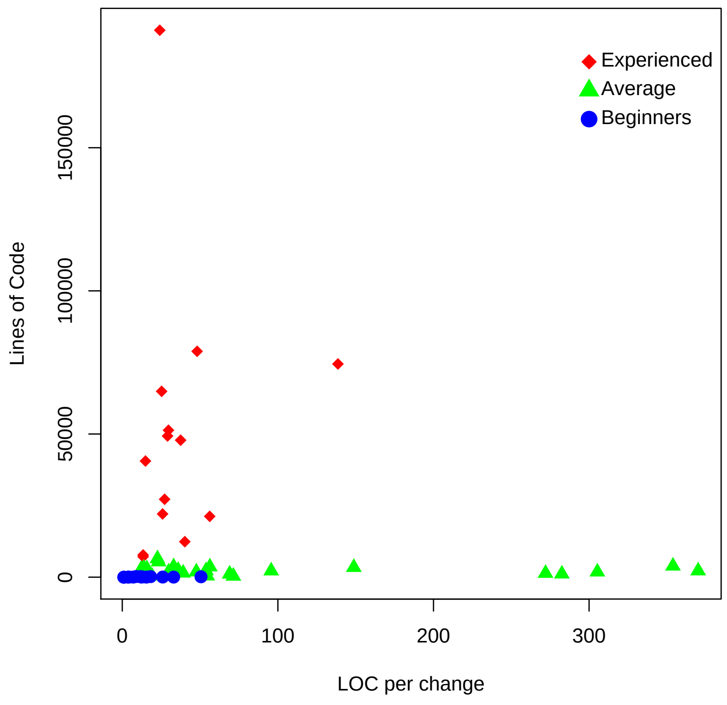
For example, the Wiki environment works best when developers submit small incremental changes that colleagues are able to understand and build on. However,

Figure 5.2: Developers against the growth of the software [18].



this requires new disciplines and skills that developers need to learn. This can clearly be seen in Figure 5.3 where the total number of LOC submitted by each programmer has been tallied against their average LOC per change submission. The corresponding statistics are shows in Table 5.1. These data would suggest that beginners tend to keep changes short while they are learning the system. However, once confidence has been built, the average programmer is more likely to submit larger patches than either beginners or experienced contributors to the system ($p < 0.99$). Experienced developers within this environment submit smaller patches more often which allows colleagues a chance to further develop the code during breaks (allowing development to continue on a 24 hr basis).

Figure 5.3: Effect of experience on submission size

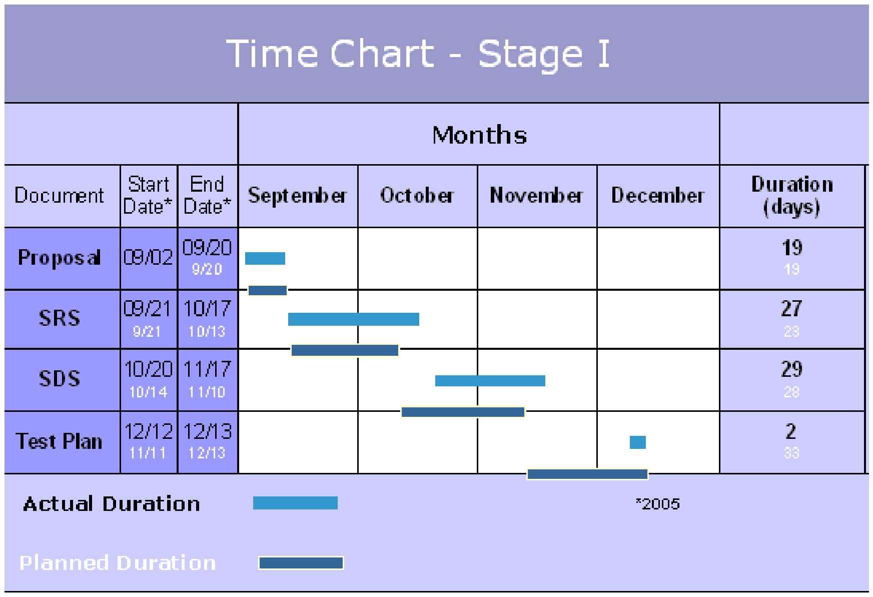


5.2 Distributed Development of IPDNS

As described in Chapter 3, the IPDNS project is developed under two phases: distributed development is used for requirement elicitation and software design, centralized development is used for implementation and testing. In Phase 1, Wiki was used as the infrastructure for communication and coordination. In Phase 2, face-to-face meeting is used as the major communication method.

In the initial stage, five stakeholders are involved in this project. Therefore, we utilized distributed development, in which Wiki is the major communication and coordination tool. The project Wiki was set up before the project started. In software development, Wiki was used to facilitate coordination among team members; in software maintenance, Wiki is used to monitor the operation of the software product. The time chart for this phase is shown in Figure 5.4

Figure 5.4: Time chart - Phase 1



Time Chart - Stage I

			Months				
Document	Start Date*	End Date*	September	October	November	December	Duration (days)
Proposal	09/02	09/20 9/20					19 19
SRS	09/21 9/21	10/17 10/13					27 23
SDS	10/20 10/14	11/17 11/10					29 28
Test Plan	12/12 11/11	12/13 12/13					2 33

Actual Duration

Planned Duration

*2005

5.2.1 Co-editing

In the IPDNS project, the most important application of Wiki was to support co-editing by multiple developers. Table 5.2 shows the summary of the work on four major documents produced in this project: Project Proposal, Software Requirement Specification (SRS), Software Design Specification (SDS), and Testing Plan. The size of the document was measured in the number of words. The rate of change was measured in number of words added, deleted or modified. The table also contains information about the amount of time spent in the co-editing of the document and the number of times the document was co-edited. The number of different IP addresses used to access the Wiki is a measure of the different physical locations where the team members were working from and reflected the amount of mobility of the group.

Table 5.2: Summary of the coediting of four documents

	Size (words)	Change (words)	Change (times)	Span (days)	IP
Proposal	2756	2811	291	19	7
SRS	1147	926	155	23	9
SDS	2542	2336	126	29	7
Test plan	1262	205	24	2	2

Table 5.2, shows that the number of changes made to Proposal, SRS, and SDS are about same size of their final documents respectively. This illustrates the volatile nature of co-editing. For example, in the case of the proposal, it was modified 291 times in 19 days. (This means, on average, the proposal was modified 15 times a day by different members). Without the support of Wiki, it would have been very difficult, even for professional developers, to use traditional sequential editing techniques, and updated to distribute files at that frequency. Because the developers were either full time students or full time staff, there were very limited opportunities for face to face meetings for this project, because of other duties and commitments. Using Wiki meant that the newest version of the documents were always available despite the hectic and asynchronous schedules of the developers.

Figure 5.5: The editing effort on four major documents

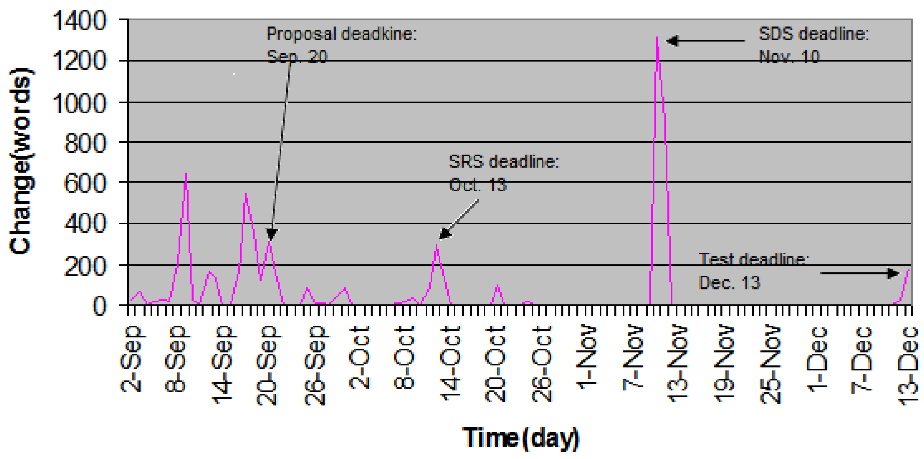


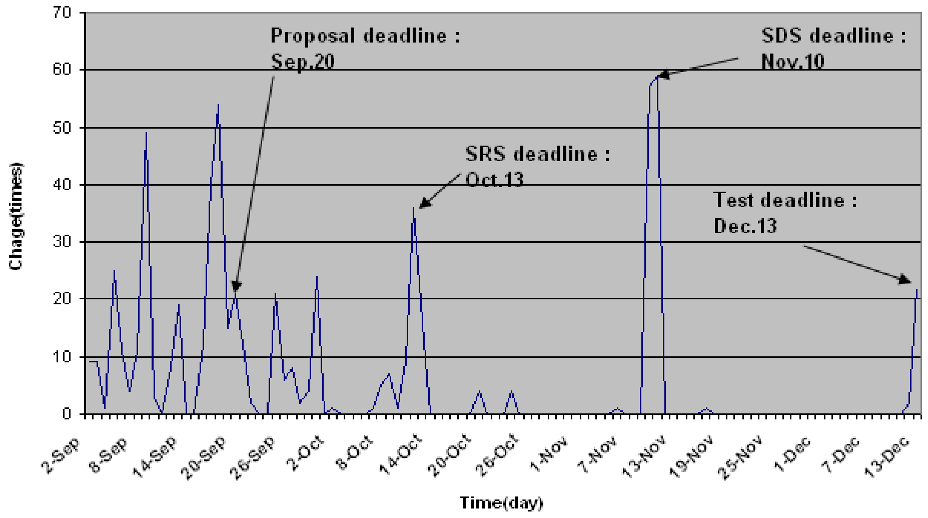
Table 5.3: The deadline of four documents

Document	Deadline
Proposal	September 20, 2005
SRS	October 13, 2005
SDS	November 10, 2005
Test plan	December 13, 2005

Figures 5.5 shows the co-editing effort on four documents: Proposal, SRS, SDS, and Testing Plan. When the number of words modified on a document is plotted against time, it is clear that the effort spent on co-editing each document increases as deadline draws near (Table 5.3). This stimulus caused by deadlines is analogous to the effect of site visits reported by Dr. Liguo [19]. Wiki allowed developers logging into work from remote location simultaneously on the same document. Without the support of Wiki, the deadline would not have been met.

Figure 5.6 illustrates the number of times a document was changed each day

Figure 5.6: All documents - time of changes per day

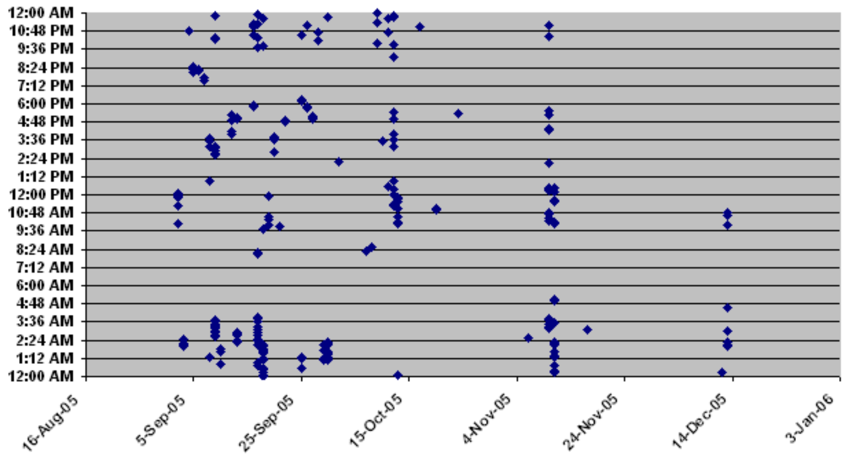


during the editing of Proposal, SRS, SDS, and Testing Plan. The frequency of changes increased dramatically before the due date. For example, SDS was modified about 60 times on the due date, November 10, 2005. The frequent changes facilitated synergy between the participants and the support of Wiki made it possible to be creative and to deliver documents on time.

From our experience, Wiki not only facilitated the co-editing, but also sped the documentation process. It reduced the time for file exchange and allowd multiple programmers to work on the same document at the same time.

The communication was not hampered by external pressure from spam. Another advantage we found about Wiki is the location and time of work. As long as a developer could access the internet, he could modify the document as needed no matter where he was and what time it was. For example, when one developer was on

Figure 5.7: Team activity in co-editing

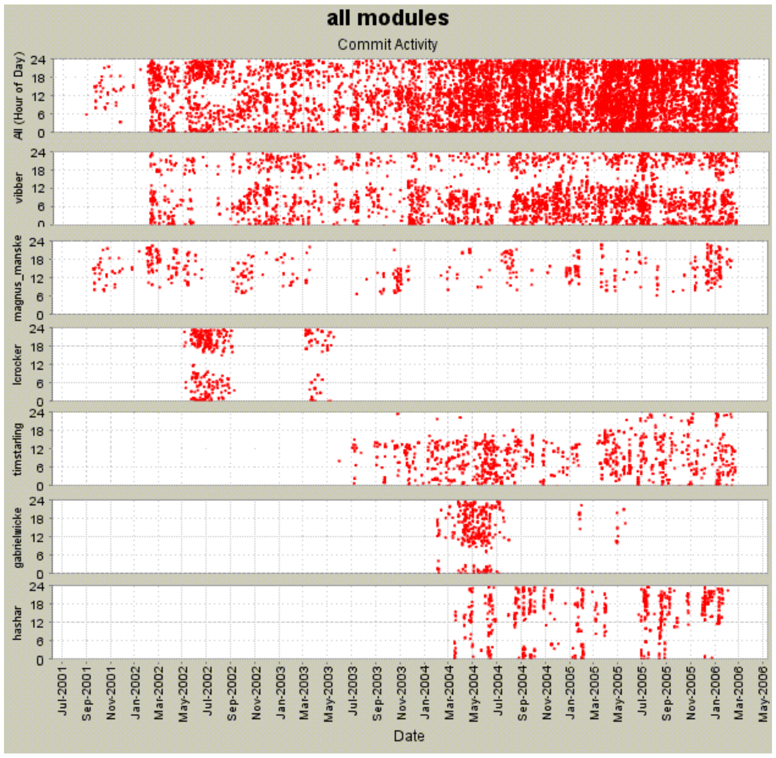


a trip to New York, no one expected that he could work this project. However, with the laptop and the internet access, he was able to contribute to this project from his hotel. Figure 5.7 shows the co-editing activities of the three developers. We can see, with the support of Wiki, the developers are not limited by time. They worked on the project whenever they wanted and/or were able to. The advantage was that the Wiki always provided the most current version of the document in a simple-to-use interface that did not stifle creativity.

This synergy from collective round-the-clock effort was also seen by MediaWiki [4]. Figure 5.8 shows a sample of the authors activities in Project Avar. The vertical axis indicates the activities of seven most prolific authors. It can be clearly seen that these members work on this project at different times of day. The top composite tracing in Figure 5.8 illustrates the transition of Wikimedia into 24×7 from a local

[4]http://tools.wikimedia.de/∼ avar/cvs/html/all/authors.html

Figure 5.8: Authors activities in Wikimedia project Avar[18]



operation service. This can be seen in the disappearance of the evening break in the development seen in the early years of the development. The example of contributors work hours show that development continues despite the fact that there is no shared timezone or work day among the chief developers. With the support of Wiki, it is easier for developers around the world to work together on time critical projects.

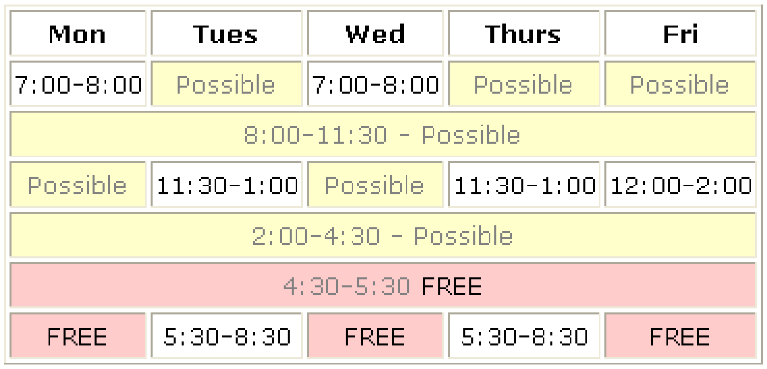
In addition, Wiki also helps to organize the co-editing. For example, after a team leader created a new page and wrote the headlines, the other developers could fill in the empty spaces to complete the document.

5.2.2 Communication and Project Management

In the IPDNS, Wiki was used most of the time for communication instead of a phone and an email. Two distinct advantages of Wiki were seen: (1) Compared to phone conversations, Wiki is not constrained by the time schedule of the developers; (2) Compared to email, messages posted to Wiki is easy to be noticed and more reliable. With the growing pressure of spam and increasing email traffic, developer's email account may be filled with junk mails or other unrelated mails. It was also possible for a developer to ignore or even accidently delete an important email regarding the project. These communication failures were avoided by using Wiki.

Wiki is also used in peer review and indirect communication. Simple databases were easy to implement and maintain. For example, Wiki was used to show the time schedules of the developers. The schedule was easily updated and available to all other team members. When there is a need for an appointment or a change of schedule, developers could refer to the corresponding Wiki web page and update the information as needed. All changes were instantly available to the other developers. Figure 5.9 shows the screen shot of one such schedule. At the same time the project also used Wiki in its communication between the developers and the instructor mentor. The final document and the presentation slides are made available through the Wiki web site. Wiki proved to be are ideal platform for the communication of creative ideas between the stakeholders. In addition, clients could know the progress of the project, download demos, and interact with developers by directly accessing the Wiki web

Figure 5.9: Wiki is used to show time schedule



Mon	Tues	Wed	Thurs	Fri
7:00-8:00	Possible	7:00-8:00	Possible	Possible
8:00-11:30 - Possible				
Possible	11:30-1:00	Possible	11:30-1:00	12:00-2:00
2:00-4:30 - Possible				
4:30-5:30 FREE				
FREE	5:30-8:30	FREE	5:30-8:30	FREE

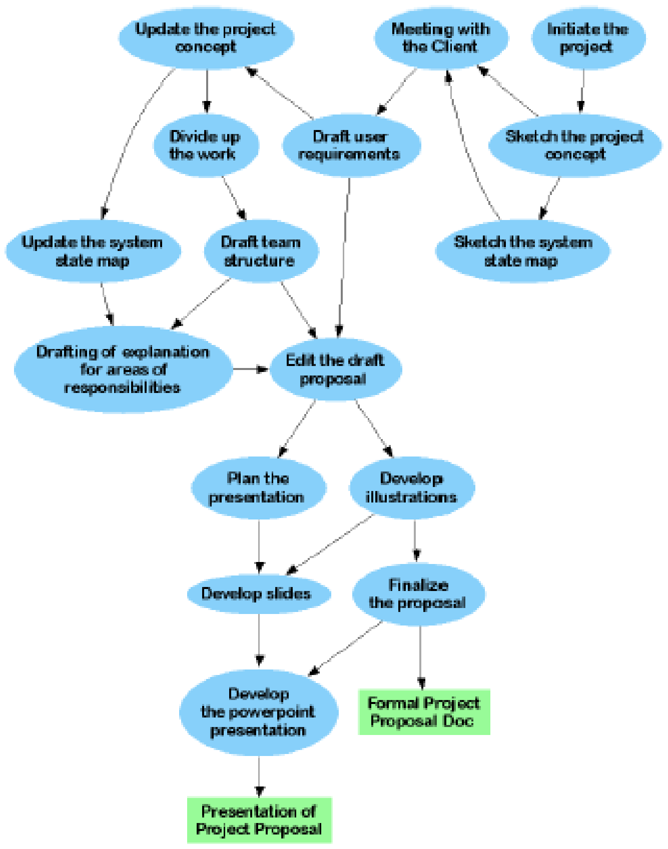
site.

In our project, Wiki was the only software used for the management of the entire project. The project manager posted the finalized document on the web site to make it available for all the developers. The manager could control the modification of these documents. Beside formal project document, other project information was also exchanged with Wiki. Figure 5.10 shows the dependency of all the tasks created in the planning phase by the project manager. It is available in Wiki and can help other developers to understand the project progress and the current development stage.

5.2.3 Other Tasks

Trac[17] is a system for managing tasks within software projects. It provides flexible web-based issue tracking services. In this project, we integrated Trac system with wiki and used it for various proposes. Figure 5.11 shows the progress of current

Figure 5.10: Wiki is used for project management


Update the project concept
Meeting with the Client
Initiate the project
Divide up the work
Draft user requirements
Sketch the project concept
Update the system state map
Draft team structure
Sketch the system state map
Drafting of explanation for areas of responsibilities
Edit the draft proposal
Plan the presentation
Develop illustrations
Develop slides
Finalize the proposal
Develop the powerpoint presentation
Formal Project Proposal Doc
Presentation of Project Proposal

tasks using Trac ticket system.

In the IPDNS project, tickets were used to report bugs, present problems, and review project status. A ticket contains the information about the problem or the issue, such as the reporter, the status, the type, the priority and the resolution of the issue. Most importantly, Trac integrated with Wiki forms to create a flexible issue management system within the Wiki enviroment where it was possible to change and comment on specific problem tickets at any time.

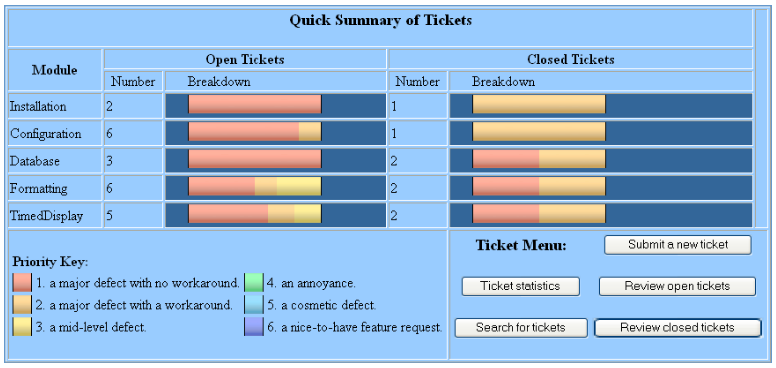


Figure 5.11: A screen shot of quick summery of tickets

Quick Summary of Tickets

Module	Open Tickets		Closed Tickets	
	Number	Breakdown	Number	Breakdown
Installation	2		1	
Configuration	6		1	
Database	3		2	
Formatting	6		2	
TimedDisplay	5		2	

Priority Key:

🟥 1. a major defect with no workaround.	🟩 4. an annoyance.
🟧 2. a major defect with a workaround.	🟦 5. a cosmetic defect.
🟨 3. a mid-level defect.	🟪 6. a nice-to-have feature request.

Ticket Menu:

In the IPDNS project, the Trac system was used to help developers manage various tasks such as problem assessment and project status review. For example, meeting notes and tickets were used to review the project status. They provided summary information about the status of the project. Figure 5.11 and 5.12 illustrate the web interface for the ticket creator and meeting notes respectively.

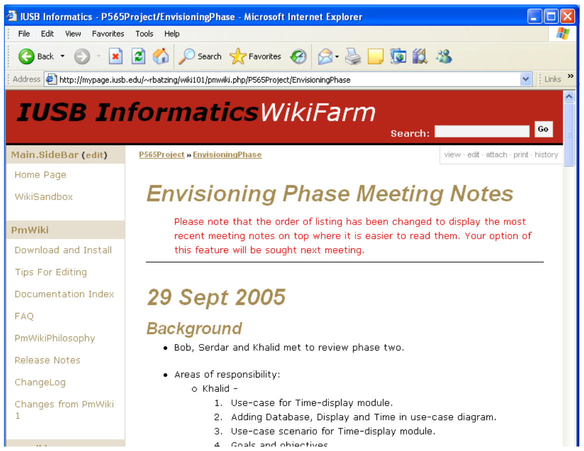


Figure 5.12: A screen shot of meeting note

IUSB Informatics - P565Project/EnvisioningPhase - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://mypage.iusb.edu/~rbatzing/wiki101/pmwiki.php/P565Project/EnvisioningPhase
Links
IUSB InformaticsWikiFarm
Search: Go
Main.SideBar (edit)
Home Page
WikiSandbox
PmWiki
Download and Install
Tips For Editing
Documentation Index
FAQ
PmWikiPhilosophy
Release Notes
ChangeLog
Changes from PmWiki 1
P565Project » EnvisioningPhase
view · edit · attach · print · history

Envisioning Phase Meeting Notes

Please note that the order of listing has been changed to display the most recent meeting notes on top where it is easier to read them. Your option of this feature will be sought next meeting.

29 Sept 2005

Background

- Bob, Serdar and Khalid met to review phase two.
- Areas of responsibility:
  - Khalid -
    1. Use-case for Time-display module.
    2. Adding Database, Display and Time in use-case diagram.
    3. Use-case scenario for Time-display module.
    4. Goals and objectives

We also found tickets to be helpful in reviewing the project status. The graphical notation made progress monitoring user friendly. Tickets can also facilitate project review after it is finished. We could extract data about a particular task or issue. The data can then be analyzed using integrated statistical tools. The data and knowledge gained from the ticket system were valuable assets to improve the software process.

5.2.4 Limitations of Wiki

As with any other software tools, Wiki has its limitations.

1. Currently, Wiki does not support coediting of complex document type, such as Word, and Excel. Because these document types are ubiquitous, development of these documents online in a cooperative manner would be useful. While MS SharePoint attempts to deliver some of these facilities using the MS Office Suite, it is currently too expensive and unreliable for the average consumer.

2. Visual and audio contact is important for software development, especially when collaboration depends on close teamwork. If Wiki could be integrated with other web conference software, it would be more helpful for some specific tasks, especially when relationships within developers or between developers and client are tense or require repair.

3. While Wiki is best for software development and maintenance, it is not as suitable for requirement elicitation. Brain storming appears to require an element of trust that is hard to build remotely. A face-to-face interview or an on-site

observation is more efficient than Wiki to obtain the software requirement from the client, particularly when the client is not clear as to what the application needs to do.

4. Most current implementations of Wiki do not support grammar and word-checking which are needed to provide a flexible and robust editing environment. With these capabilities, it would be easier to generate publishable documents from the contents developed.

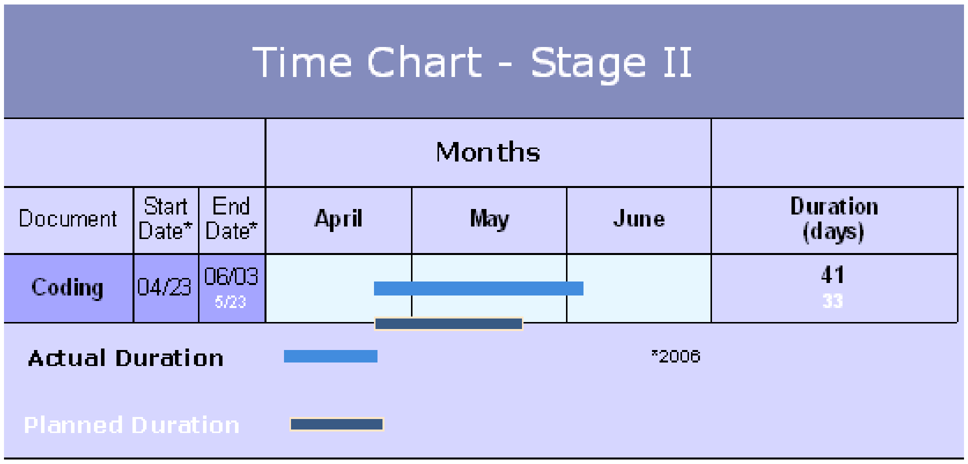
5.3 Centralized Development

In the second phase of the project, centralized development is used. Because only two stakeholders (the programmer and the coadvisor) were involved in implementation and testing, face to face meetings became the major means for communication and coordination.

The time chart for implementation and testing is shown in Figure 5.13. Because of the number of technical issues that were discussed and resolved, face-to-face meetings appeared easier than written communication. In particular, the developer and the coadvisor worked together to debug and test the program, and the coadvisor taught the programmer how to use certain software tools, centralized development was more efficient then distributed development. According to our experience, centralized face-to-face meetings are more efficient than distributed development in the following cases:

debugging a program, testing a program, tutoring, code implementation, and difficult issue resolution. However, they required more careful scheduling and preplanning than Wiki communications.

Figure 5.13: Time chart - Phase 2



Time Chart - Stage II

			Months			
Document	Start Date*	End Date*	April	May	June	Duration (days)
Coding	04/23	06/03 5/23				41 33

Actual Duration

Planned Duration

*2006

Here, we show an example of using face-to-face meeting for issue resolution. The main function of display model is to create a process that can run different types of viewers, PPT, PDF and HTML. During the implementation, we found a problem with HTMLs' viewer. The procedure is shown in Block 5.1. It creates a process to run the FireFox viewer which displays the HTML file. Every time a process was created, it had to be tracked in order to be able to kill that process after it finished its task.

For example, when two or more HTML files are displayed in order, there was a problem caused by the few seconds of delay between killing the first viewer and starting the second one. This problem was found to be intrinsic to the interaction between the Firefox viewer and the WinXP operating system. In addition, the extra

Block 5.1 Creates a new process

```
Win32::Process::Create($ProcessObj,
                       "C:\\Program Files\\Mozilla Firefox\\firefox.exe",
                       "firefox filename",
                        0,NORMAL_PRIORITY_CLASS,
                        ".")|| die ErrorReport();
sleep($DisplayRunTime);
$ProcessObj->Kill(0);
```

seconds of waiting would make the display inefficient.

To solve this problem, the process ID was pushed on a stack (Block 5.2), when displaying HTML files. If the running file is PDF or PPT format, all the processes on the stack were killed(Block 5.3).

Block 5.2 Pushing processes ID

```
Win32::Process::Create($ProcessObj,"C:\\Program Files\\Mozilla Firefox\\firefox.exe",
                       "firefox filename",
                        0,NORMAL_PRIORITY_CLASS,
                        ".")|| die ErrorReport();

sleep($DisplayRunTime);
$pid = $ProcessObj->GetProcessID();
push(@kill_pid_html,$pid);
```

Block 5.3 Killing processes

```
Win32::Process::Create($ProcessObj,
                      "C:\\Program Files\\Microsoft Office\\PowerPoint Viewer\\PPTVIEW.EXE",
                      "PPTVIEW  filename",
                      0,NORMAL_PRIORITY_CLASS,
                      ".")|| die ErrorReport();

kill 9, @kill_pid_html;
sleep($DisplayRunTime);
$ProcessObj->Kill(0);
```

This tracking of HTML files opened by FireFox became a major issue that required special care during the implementation of the system. The developer and the coadvisor attempted to use e-mail and Wiki to communicate and discuss this issue.

However, after several days, progress was stalled. Finally, we met face to face and worked together and tried many different ways to solve this problem. After several hours of debating, and searching the internet for solutions, the problem was solved. Without that face-to-face meeting, it would have taken even longer to resolve this problem.

5.4 Discussions

The IPDNS project used both distributed and centralized development. Both of which have their strengths and weaknesses. In practice, the development style should be carefully selected according to different working environment, different levels of problem solving, and different skills of the developers. We also learned in this project that distributed development requires active participation of all members in the development. If some members are not prompt in responding to online messages or document updates, the benefit of distributed development is lost. Under this situation, centralized development is more efficient because developers are more likely to feel the pressure and the urgency of the issues and more traditional management pressures can be brought to bear. If participants are not disciplined, face-to-face meetings and working in a common location would result in faster responses and greater progress.

Chapter 6

Conclusions and Future Work

The Informatics Plasma Display News Server (IPDNS) was a project intended to develop a news server to drive the plasma display operated by the Informatics Department at Indiana University South Bend. In this thesis, the implementation and testing of the Informatics Plasma Display News Server (IPDNS) was described. The project was able to develop, build and test a news server which has enhancements to the server in current use. In the future, the user interface of the IPDNS project should be improved. This can be done by building interactive Web application using the Common Gateway Interface (CGI) which is the predominant platform for deploying Web applications today.

The IPDNS project also served as a means for testing the effectiveness of different software development environments. We found both distributed and centralized development to have their strength and weakness. In practice, an appropriate development style should be carefully selected according to the demands of the development environment, problem domain, and skill of the developers.

The Wiki environment proved useful enough to recommend that serious thought be given to employ Wikis for use in all computer science courses which have group projects. This would give students valuable experience training in remote software development that would build useful shills that are increasing in importance especially with the growing trend to employ development services of off-shore developers. Wikis would also give instructors and mentors the ability to track the progress and the division of effort among the members of a student work group.

As distributed software development continues to grow, new skills will be required. Clear and concise technical communications will play an important role. In the course of this thesis, Wiki has proven itself to be a low cost platform where the communication between remote team members can be learned and studied.

Chapter 7

Appendices

7.1 Appendix A

A Database Table Definition News Categories

```
create table NewsCategories
(
  CatId INTEGER PRIMARY KEY, -- Record Identifier; autoincrementing field
  Description VARCHAR(30), -- Short description of the news category
  ViewSequence INTEGER, -- Viewing order of the news category: 0 = Adver-
                           tisement,Other Numbers = sequence
  Importance INTEGER -- Relative importance of this news category
);
```

Special Announcements

```
create table SpecialAnnouncement
(
  AnnouceId : INTEGER PRIMARY KEY, -- Record Identifier; autoincrementing
                                      field
  DisplayStartDate : DATESTAMP, -- Starting date of the announcement
  DisplayExpiryDate : DATESTAMP, -- Last date of the announcment
  MessageText : VARTEXT(240), -- Text of the scrolling message
  MessageTitle : TEXT(60), -- Short title of the message
  RelatedGraphic : TEXT(60) -- Relevent graphic for HTML version
);
```

News Resource

```
create table NewsResource
(
  ResId INTEGER PRIMARY KEY, -- Record Identifier; autoincrementing field
  FileName VARCHAR(60), -- Filename of the resource
  Directory VARCHAR(60), -- Directory where resource can be found
  Category INTEGER, -- Associated CatID from NewsCategories table
  DisplayStartDate DATESTAMP, -- First date to display the news (yyyy-mm-dd)
  DisplayExpiryDate DATESTAMP, -- Last day to display the news (yyyy-mm-dd)
                                  or the word NEVER
  DisplayRunTime INTEGER, -- Duration of the display on screen in seconds
  ResourceType CHAR(3), -- Type of news resource: RSS, PDF, HTM, PPT
  Link INTEGER, -- Record number of RSS Feed
  Status INTEGER -- Status of availability: 0 = Unavailable, 1 = Available
);
```

RSS Feeds

```
create table RSSFeeds
(
  RSSId INTEGER PRIMARY KEY, -- Record Identifier; autoincrementing field
  Layout CHAR(8), -- Layout generator be used
  HighlightColor CHAR(8), -- Highlight color in HEXDEC RGB
  RefreshFrequency INTEGER, -- News feed refresh frequency: 0 = never;
                               > 0 represents hours
  URL CHAR(80), -- Location of the news feed online
  TradeMark CHAR(40), -- Filename of the trademark of the news source in
                         the image directory
  Source CHAR(60), -- Name of the news feed
  DateRefreshed DATESTAMP, -- The date of the last refresh YYYY-MM-DD
  TimeRefreshed TIMESTAMP, -- The time of the last refresh HH:MM:SS
  ActiveDirectory CHAR(1) -- Current temporary directory of active news items
);
```

HTML Rendering of RSS feeds

```
create table RenderedHTML
(
  HTMLId INTEGER PRIMARY KEY, -- Record Identifier; autoincrementing field
  RSSFeed INTEGER, -- Associated RSS Feed
  Directory CHAR(1), -- Working directory
  FileName CHAR(20), -- File name of the converted HTML file
  DisplayTime INTEGER, -- Calculated display time of the HTML files
  CreationDate DATESTAMP, -- Date created
  CreationTime TIMESTAMP -- Time created
);
```

Bibliography

[1] Robert Batzinger, Khalid R. Al-asmari, and Serder Demir. Software design specification. Available online `http://mypage.iusb.edu/~ rbatzing/wiki /index.php/P565Project`, Accessed 2 August 2006.

[2] Scott Berkun. *The art of Project Management.* Oreilly, 2005.

[3] Michael Bittner. Global product development seen as a boon for product lifecycle management vendors. In *Technology Evaluation, December 12.*

[4] Erran Carmel. *Global Software Teams.* Prentice Hall, 1999.

[5] Erran Carmel and Ritu Agarwal. Tactical approaches for alleviating distance in global software development. In *IEEE Software, vol. 18, Issue 2*, 2001.

[6] Cherry, S., Robillard, and P.N. Communication problems in global software development: Spotlight on a new field of investigation. In *The 3 rd International Workshop on Global Software Development, ICSE04 , May 24, Edinburgh*, 2004.

[7] Ward Cunningham. Wikiwikiweb. Available online from Protland Pattern Repository `http://c2.com/cgi/wiki`, Accessed 2 August 2006.

[8] Daniela Damian. Global software development: growing opportunities, ongoing challenges. In *Software Process: Improvement and Practice. Volume 8, Issue 4, Pages 179-182*, 2003.

[9] Goldenink. Perl debug. Available online `http://goldenink.com/perl/perldebug.html`, Accessed 6 July 2006.

[10] James D. Herbsleb and Deependra Moitra. Guest editors' introduction: Global software development. In *IEEE Software, Vol. 18, No. 2, pp. 16-20.*

[11] J.D. Herbsleb and A. Mockus. An empirical study of speed and communication in globally-distributed software development. In *IEEE Transactions on Software Engineering. Vol. 29, NO. 6. pp. 481-494*, 2003.

[12] Filippo Lanubile, Daniela Damian, and Heather L. Oppenheimer. Global software development: technical, organizational, and social challenges. In *ACM SIGSOFT Software Engineering Notes. Volume 28 , Issue 6*, 2003.

[13] Microsoft. Top 10 benefits of developing globally. Available online `http://www.microsoft.com/globaldev/getWR/10benefits.mspx`, Accessed 3 January 2006.

[14] The Wharton School of the University of Pennsylvania. Why

global software development unleashes innovation. Available online `http://knowledge.wharton.upenn.edu/index.cfm`, Accessed 3 March 2006.

[15] Rafael Prikladnicki, Jorge L. N. Audy, and Roberto Evaristo. An empirical study on global software development: Offshore insourcing of it projects. In *In Proc. of the Int'l Workshop on Global Software Development, International Conference on Software Engineering (ICSE 2004), Edinburgh, Scotland, IEE, pp. 53–58.*, 2004.

[16] RSS Advisory Board. Really simple syndication specification 2.0.1. Available online `http://www.rssboard.org`, Accessed 2 August 2006.

[17] Edgewall Software. Welcome to the trac project. Available online `http://projects.edgewall.com/trac`, Accessed 28 March 2006.

[18] WikiMedia. Authors. Available online `http://tools.wikimedia.de/~avar/cvs/html/all/authors.html`, Accessed 20 January 2006.

[19] Liguo Yu, Robert P. Batzinger, and Srini Ramaswamy. A comparison of the efficiencies of code inspections in software development and maintenance. In *Proceeding of 2006 International Conference on Software Engineering Research and Practice, Las Vegas, Nevada, June 26-29, pp. 460-465.*, 2006.