A Knowledge Analytics Portal for Agile Programming

by

Shijun Ding

A Thesis Submitted to
Saint Mary's University, Halifax, Nova Scotia,
in Partial Fulfillment of the Requirements for
the Degree of Master of Science in Applied Science

December 12, 2016, Halifax, Nova Scotia

Approved:	Dr. Hai Wang Supervisor Department of Finance, Information Systems and Management Science
Approved:	Dr. Yinglei Wang External Examiner School of Business Acadia University
Approved:	Dr. Zoey Wan Supervisory Committee Member Department of Finance, Information Systems and Management Science
Approved:	Dr. Genlou Sun Supervisory Committee Member Department of Biology
Approved:	Dr. Roby Austin Graduate Studies Representative
Date:	December 12, 2016

Abstract

A Knowledge Analytics Portal for Agile Programming

by Shijun Ding

Abstract: Agile programming has been widely adopted for software development. One criticism on agile programming is the lack of documentation and knowledge sharing in the software development process. This thesis proposes a novel knowledge management approach for creating, managing, and sharing various types of documents for agile software development. The proposed approach divides software developers into two groups, *masters* and *apprentices*. Masters are senior software developers who are primarily responsible for agile software development. Masters avoid document writing as advocated in the agile manifesto for software development. Apprentices are junior software developers who are mainly responsible for creating, managing and sharing various types of documents for the agile software development process. The effectiveness of the proposed approach is investigated and validated through a prototype of a knowledge analytics portal. This thesis concludes that the proposed knowledge analytics portal increases the knowledge sharing for the agile software development process.

December 12, 2016

Table of Contents

Chapter 1
Introduction

At the beginning of 2000's, 17 software programmers met together and published a new group of lightweight software development methods called *Agile Software Development*. Before the term "Agile Software Development" was laid out, some important components of the agile software development process like *Scrum* (Schwaber 1997), *Extreme Programming* (Beck 2000) and *Dynamic Systems Development* (Stapleton 1997) had already been introduced, implemented and extensively studied. Nowadays, the growth of the IT industry is in an incredible speed. In the last decades, the adoption of the agile software development process has increased dramatically. It has achieved success in different perspectives, including banking systems (Bossi 2003), universities systems (Muller 2001) and government systems (Fruhling *et al*. 2008).

The agile software development process can delivery early version of software, support frequent communication, and respond rapidly to the changes from the customer by changing the software in next iteration. This process has overcome the limits of traditional methodologies for software development. As a result, more and more programmers show interest in the agile software development process.

Among all agile methodologies, Extreme Programming is absolutely the most famous one in last years (Mannaro 2008). The benefits of Extreme Programming compared to traditional software development are in "extreme" level.

- Continuously code review: Intensive on-time code review can avoid a lot of time waste for finding and fixing mistakes in code.

- Simplicity and clarity in code: The goal of extreme programming is writing simple and reusable code for achieving basic function. This will make fewer mistakes.
- Frequent communication with customer: According to the feedback from customers, with the time passes, the entire project is better understood.
- Less paperwork: Extreme Programming suggests face-to-face communications to produce smaller amount of high-value documentation.

Extreme programming means faster development and less comprehensive documentation. There is an ideal level for documentation in the agile software development process called "Just Barely Good Enough" (Ambler 2008). There is no exact template for the documentation. The best amount of documentation is just enough for the next iteration. Because too comprehensive documentation is a waste of precious time, and developers don't rely too much on detail documentation which might out of sync with the latest version of code (Ambler 2011). However, the lack of documentation may cost big loss for companies using Extreme Programming.

With the rapid changing environment of the IT industry, programmers turn to be a resource. As employees, turnover is also possible among programmers. Especially for experienced programmers, they have unique skills and are strongly demand in market. Retaining programmers is always a very important thing for an organization. In addition, every organization pays attention to the importance of training employees - both new and experienced.

While for extreme programming, less paperwork will make turnover much more expensive. Writing documents spends precious development time. In order to achieve rapid release, documentation is often left to the last. The early releases are without documentation. Once a programmer is gone, he might leave multi-million lines of codes and nobody knows what exactly he has already done. It is very likely that new programmers will rewrite a part of or the whole program. As a result, using Extreme Programming, it is likely to cost the organization much more with high employee turnovers and to supply less documented knowledge for new programmers.

Like Extreme Programming, other agile software development methods also suffer similar problems with the lack of documentation and knowledge management during software development.

1.1 Thesis Contributions

As agile software development projects generally suffer problems with the lack of documentation and knowledge sharing, this thesis proposes a novel knowledge management approach for creating, managing, and sharing various types of documents during the agile software development process. The proposed approach divides software developers involved in an agile software development project into two groups, *masters* and *apprentices*. Masters are senior software developers who are primarily responsible for agile software development. Masters avoid document writing as advocated in the agile manifesto for software development. Apprentices are junior software developers who are mainly responsible for creating, managing and sharing various types of documents for the agile software development process. With the existence of documentation, knowledge management and sharing become plausible for agile software development projects. Thus, our proposed approach enables knowledge management for agile software development. The effectiveness of our proposed approach is also investigated and validated through the implementation of the prototype of a knowledge analytics portal for agile software development. Our web portal has two layers, the front end and the back end. The front end is actually a presentation layer, which includes all User Interfaces and a Kanban Display system. The back end is a data access layer. It includes a server-side XML transformation system and a database which is under version control.

1.2 Thesis Organization

The remainder of this thesis is organized as follows. Chapter 2 reviews the related work and research literature. Chapter 3 presents the proposed knowledge management approach for the agile software development process. Chapter 4 describes the features of the prototype of a knowledge analytics portal for the proposed knowledge management approach. This prototype demonstrates the effectiveness the proposed knowledge management approach for the agile software development process. Chapter 5 summarizes the conclusions and future work.

Chapter 2
Background and Literature Review

2.1 Agile Software Development

Agile Methodologies(AMs), which were proposed in 2001, now become a widely adopted software development approach (Silva *et al*. 2008). These time-efficient methodologies respond faster to changing requirements comparing to the traditional ways. AM connects programmers and customers closely and allows early releases of product (Mannaro 2008). As very efficient and effective processes, the adoption rate for AMs is more than 60% (Ambler 2008). The “Manifesto for Agile Software Development” declared a common guideline for using Agile Methodologies. The following twelve principles were specified to support the manifesto (Beck *et al*. 2001):

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.

8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity – the art of maximizing the amount of work not done – is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

The most important part for agile programming is iteration. Iteration is actually a time period during which programmers finish the development. Usually the duration of iterations vary in different projects, from two weeks to four weeks (Ge *et al*. 2010).
The projects' source codes may change largely in order to meet customer's review result between two iterations. At the end of each iteration, there will be an available release (Beck 2000). Multiple iterations might be required to release a product or new features. As shown in Figure 2.1, agile software development projects can minimize overall risk of project failure.

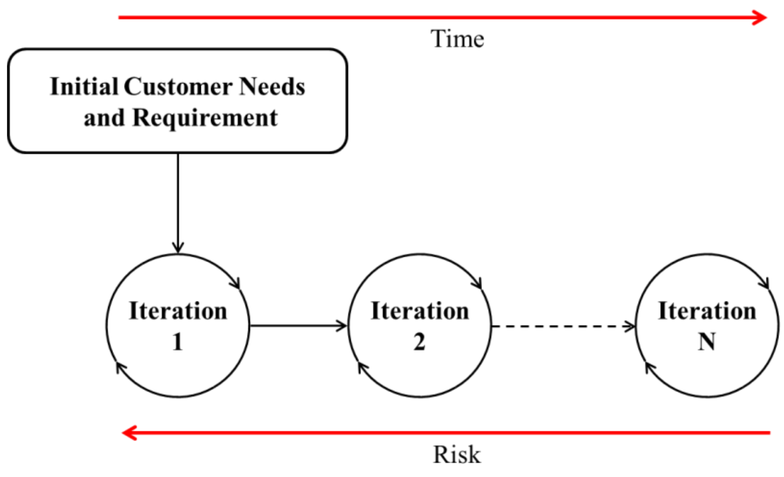


Figure 2.1 Time-Risk for Agile Life Time

Compare to the traditional plan-driven model for software development, the iterative development model shown in Figure 2.1 is generally the main component of software development projects involving extreme programming and other agile methodologies. In every iteration, it includes planning, implementation, evaluation and testing. After each iteration, it releases a code version for customer to examine during the part Implementation.

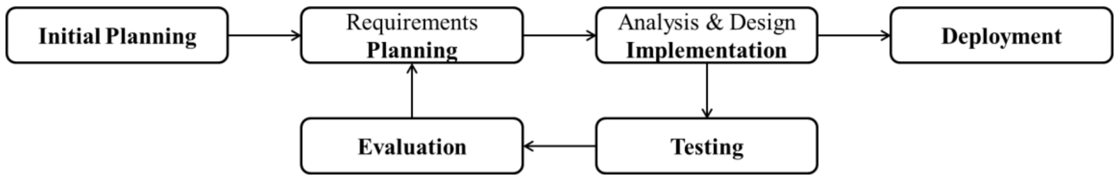


Figure 2.2 Iterative Development Model

The goal of using iterative development model is to have a working release to stakeholders after each iteration. The output of one iteration might not be a fully functioned market release. However, this function minimizes total risks and gets customers' feedback in a short period. As for the traditional plan-driven development method, even though the developers did all their best for gathering all requirements, the well-designed plan is always need to go back and make a change. The most importance value of accepting AMs is that can get rid of the cost of regret.

2.1.1 The Scrum Framework

Scrum is one of the most popular Agile Methodologies due to its favorable results. Scrum is actually a framework for managing software development projects. According to its iterative and incremental features, individuals can address all kinds of complex issues for delivering the most noteworthy conceivable quality of results (Schwaber & Sutherland 2013).

Scrum encourages a development team to collaborate as a single unit to fulfill its goal from a project management point of view (Schwaber & Sutherland 2013). In Scrum, the entire project is broken down into mini tasks referred to as “Sprints”. A Sprint is a period box of one month or less amid which a finished, useable, and conceivably software component is developed. The Scrum team adopts an iterative and incremental way to achieve streamline consistency and control hazard. The Scrum team aims to deliver the highest customer value and develop most flexible working software.

The followings are the key features of the Scrum framework:

The Scrum team: The Scrum team consists of a product owner, a development team, and a scrum master:

1) *Product Owner*: The product owner a single person who is in charge of amplifying the estimation of the software product for the development team. The product owner represents the customer and communicates the needs of the customer to the scrum master and the development team. It is the product owner

who guarantees the development team complete customer requirements to the level required.

2) *Scrum Master*: The scrum master works with development team, whose duties are supervising, motivating, and training the development team. The scrum master also acts as an intermediary to interpret and guide the development team to completely fulfill the product owner's requirements.

3) *Development Team*: The development team comprises of programmers responsible for the actual development of a conceivably, releasable software component called "increment" toward the end of every Sprint. Team members are organized and sort out to deal with their own particular work. The subsequent collaboration enhances the development team's general efficiency and transparency. Ideal development team size is usually five to nine (Schwaber & Sutherland 2013).

Scrum Artifacts: There are four main artifacts of Scrum: product backlog, Sprint backlog burndown chart and increment.

1) *Product Backlog*: The product backlog is a requested rundown of everything that required for completing the software product, and is the single source of prerequisites for any progressions to be made to the software product. It is produced by the product owner. The product backlog can be continually changed and updated.

2) *Sprint Backlog*: A Sprint backlog consists of a filtered version of the product backlog. Items inside of the Sprint backlog are chosen by the scrum master and

development team in order to set the goal of a Sprint. The Sprint backlog represents all functionalities that will be achieved in the next increment. Only the development team can modify the Sprint backlog during a Sprint.

3) *Burndown Chart*: The burndown chart is used to show the progress of the project, what has been accomplished accompanying with the time frame.
4) *Increment*: The increment is a functional, usable and releasable version of the software product at the end of each Sprint. Each Increment is completely tested to work additively with all past increments.

The Sprint: The heart of the Scrum framework is a Sprint, which is an iteration of software development life cycle to deliver a functional, usable and releasable version of the software product. The goal of a Sprint is determined by the consensus between the product owner and development team before the Sprint. The duration of each Sprint is determined by the scrum master. Typically, a Sprint lasts about 30 days (Layton 2015).

One criticism of the Scrum framework is that continuous changes of system requirements and instability with respect to the exact way of the completed product make for a somewhat extreme venture to life cycle for the project (Schwaber & Sutherland 2013). Also, the assumption that development team can reasonably estimate the time for finishing tasks is a risk. The Scrum project may suffer a risk of scope creep that unless there is a clear end date of the current project, otherwise, the project management stakeholders will be enticed to continue requesting new useful functions after each Sprint.

However, the advantages of Scrum are also obvious:

1. Easy to respond to changes,

2. Early identification of problems,
3. Easy to deliver quality product in a predictable time.

2.1.2 KANBAN

Kanban is a Japanese word that literally means signboard or billboard. It was originally designed as a solution for improving producing management, which was adopted by Toyota for streamlining its car manufacturing plants; for manufacturing, its main strengths are reductions in inventory with a focus on Just-In-Time (JIT) inventory control (Monden 1983). The strengths of Kanban made it a good candidate for adaption as an IT development model, where it helps reduce work-in-progress (WIP) and increases collaboration and efficiency. It is then applied to software development and becomes a framework of Agile Methodologies (Hurtado 2013).

Kanban illustrates workflow to team members by using visual representations, such as Kanban Boards, dashboards, and other graphics. Most of the works done on a project (such as designing or programming) are visible to all project participants. Team members receive new tasks from a queue. Development teams focus on a strictly limited number of tasks (i.e., work-in-progress) at a time. In a Kanban, several vertical lanes can be set on the Kanban board to represents different working phrases such as: To-Do, In Progress, Done, or extra steps like Plan, Develop, Test, etc. Development teams are encouraged to customize the board in order to fulfill different needs (Scotland 2010, Ahmad *et al*. 2013).

Currently, Kanban broads have been integrated with many other Agile methodologies, such as Scrum and XP (Kniberg & Skarin 2010).

2.1.3 Scrumban

Scrumban is a management technology and also a framework for development progress. As the name “Scrumban” represents, it is a hybrid of “Scrum” and “Kanban”. It encourages companies adopting principles of Agile Methodologies and simple innovations can help enhance productivity and eliminate waste.

Development teams using Scrumban looking for simple solutions on comprehended project. Scrum helps development team work “in flow”, and Kanban inspires changes and improvement to the work in progress. Scrumban is not a simple combination of Scrum and Kanban, but a new method applying Kanban for Scrum technology (Reddy 2015).

The essential idea of Scrumban is relatively simple. The basic element is a board with three columns “To Do”, “Work In Progress”, and “Done” (Ladas 2008). Movable notes through columns within board representing different tasks indicate the work progress of the software development. All tasks of the project are placed in the “To Do” column at first. Then development team members pull tasks according to its priorities to work. Developers focus on numbered tasks at the same time according to their abilities. The limited number of tasks working at one time allows development team avoids distraction of multitasking. Furthermore, as the benefit from Kanban, the noticing board makes the working progress transparent for every team member (Ladas 2008).

Following the strategy of Scrum, planning meeting can be used to discuss tasks prioritization and pull the most urgent tasks to the “Work In Progress” column. In addition, a certain number of tasks left in the “To Do” column stimulate the holding of planning meeting.

Another feature coming from Scrum is encouraging feedback and self-improvement.

It includes creating empty slots in “Work In Progress” column. Given a development team having productivity of ten tasks per work cycle, three empty slots are emerged if only seven tasks were pulled into “Work In Progress” column. Three new tasks with higher priority should be arranged to progressing. In this way, works finished in a project are finished by cycles of short iterative terms, and within each iteration development teams work on a limited amount of tasks. Without setting different due date for each individual task, tasks selected in an iteration share a same deadline, which means they have equal priority. If the length of iterations and team capability within an iteration are certain, development team can finish project at an estimated rate. Team members can work effectively with reduced stress (Ladas 2008).

For a detailed solution, “Work In Progress” column in Scrumban board can be divided to sub-columns like, “Analyzing”, “Developing”, “Testing” or “Ready to release”. With sub-columns function, team can minimize the influence of facing “bottleneck”. Development team members can dynamically focus more on “Testing” if some circumstances slowing a task from “Developing”. Until the problem solved, development team members are still fully functional without stopping the whole project.

Scrumban has several advantages over Scrum and Kanban techniques. First, it adopts planning on demand which save time by neither often meetings nor complexed estimation procedures. The saved time enables development teams pay more attention on completing tasks on time. Second, it uses continuous workflow with short cycles for planning. If some circumstances happened to a task, it could be returned to queue and to be finished in a new iteration without interrupt the working progress. Third, Scrumban allows development teams remove all wastes which contribute no value to the final product.

2.1.4 Extreme Programming

Extreme Programming (XP) is a popular agile methodology. One of the most important characters of XP is fast and efficient communication. XP development includes Pair Programming, continuous integration and unit testing. Continuous Integration means frequent release of software versions within fixed duration. Instead of testing after the development, unit testing during coding can let customer involved to satisfy the changing requirement. It can also guarantee the high quality of the source code. In addition, face-to-face communication is actually the most efficient and informational method (Wright & Webb 2011). As a more adaptive, iterative and evolutionary method, XP allows early and frequent feedback from customers and discovers high risks earlier. Several important elements called XP values demonstrate how to do a successful extreme programming:

- **Communication:** XP encourages face-to-face interaction. This is the most important value to creating best solution to customers' requirements. XP is mainly based on oral communication, and lack of documented knowledge sharing.
- **Simplicity:** The code for the software must be the simplest thing that could match all the requirements. The simplicity of the code can make it easily to understand and change in the future.
- **Feedback:** XP is designed to meet the customers' frequently changed requirements. The requirements for different iterations could be different based on customers' ideas. Both the customers and unit tests can supply useful feedback.
- **Courage:** The team is fearless because no one works alone. No need to document excuses for failures and the team is capable for any change to the project.

- **Respect:** Development team members respect each other and their work.

These values are only fundamental values used by XP teams. Teams can add their own values to these if making changes is necessary for developing efficiently. The features that differ Extreme Programming from traditional software development methodologies are as follows.

- Continuously code review: As using pair programming, one of the programmers is doing the code review all the time. Intensive on-time code review can avoid a lot of time waste for finding and fixing mistakes in code.
- Simplicity and clarity in code: All of the codes are finish by two programmers. The goal of extreme programming is to achieve the basic function with simple code. It can make code reusable and reduce the risk of facing mistake.
- Frequent communication with customer: Extreme Programming is designed for frequent changing requirements from customers. With time passes the entire project will be better understood.
- Pair programming: All codes for a project finished by XP teams are completed by two programmers working together. Pair programming increases development speed. Developers focus on coding instead of unnecessary documentation and meetings. The goal is to create faster software with fewer software bugs. Pair programming will be described in details in the next section.

The disadvantages of XP are as follows.

- Detailed planning: XP is a plan-driven development method. It requires a detailed planning since the beginning because of the potential changing requirements and project scope. Time spends on detailed planning affects the efficiency of XP.

- High cost developer turnover: Extreme Programming encourages developers focus on coding instead of redundant documents. However, lack of paperwork will make turnover much more expensive.
- Code based: As a code based technique, XP don't have perfect performance on complex project as design based method do. It can be frustrated on larger project as requiring too much time on refactoring. Also, codes testing are difficult due to neither detailed structure nor well documented defects.

2.1.5 Pair Programming

Pair programming means two software developers who work together in a same workstation. This method has been demonstrated to improve software quality and minimize the time to market (Jensen 2003, Williams & Kessler 2002). Pair programming increases teams' collaboration by sharing joint responsibilities to the developers.

Sitting side by side is the best way of pair programming. Both programmers concentrating on the same task and using face-to-face communication can make code quality much higher.

The relationship between these two programmers is similar to the relationship between driver and navigator. The programmer worked as driver focuses on typing codes and system design. The other programmer reviews the code concurrently to fix any defects that might affect the project like syntax errors, mistyping and class malfunctions (Lui & Chan 2008). They constantly communicate on best approaches and switch roles frequently during developing the project.

Remote pair programming is becoming a more popular mothed. It allows programmers writing codes from anywhere have Internet access, which even enables programmers from different countries working together. However the unique feature of face-to-face communication occurs side by side must remain, in order to keeping efficiency of remote pair programming. With the rapid development speed of Internet, remote pair programming is a natural innovation of traditional pair programming. Techniques supporting remote pair programming like desktop sharing, collaborative code editing tools and web-based terminal are adopted to guarantee the efficiency.

The basic idea of pair programming is two programmers make better decisions in comparison to working individually. For general consideration, it’s hard to believe that Pair Programming is more effective than Solo Programming. Different opinions come from both sides, as: (Beck 2000)

PROS	**CONS**
Programmers as a scarce resource, is wasteful by letting two people develop on a piece of code at the same time.	Experienced pair programmers support working together in pairs as "more than twice as fast"
Programming is traditionally considered as a private activity. Many programmers think their code is "personal". Working with a partner will face trouble on deadlines or code versions.[Subversion]	Several famous programmers prefer working in pairs. The resulting design is better. Simpler code is suggested, which is easier to extend.
Many programmers feel reluctant to work with another person.	According to some surveys, even junior developers contribute to a senior's programming.

Table 2.1 Pros and Cons of Pair Programming

From an economical point of view, adding the second person only increase a small development cost. However, pair programming will decrease the probability of software bugs. Also, according to the feedback from the programmers after using Pair Programming, they found it is more enjoyable to work in pairs than working alone using traditional approaches (Williams *et al*. 2000, Denny *et al*. 2009). Moreover, pair programming suggests side-by-side programming and rapid oral communications, which save time for code review. Pair programming also provides a channel for knowledge sharing. Even junior developers contribute to a senior's programming (Cockburn 2000). Actually, pair programming can accelerate the speed of learning from each other.

Pair programming does not always work and may not be effective in all cases. Two programmers have different personalities could cause conflicts. If one programmer is lack of experience, another programmer may spend all of the time tutoring. Such situation could be very effective is learning is the main purpose. However, this could cause time waste and frustrated programmers who cannot focus on development only. Research (Dybå *et al*. 2007) suggested senior programmers should work alone unless the project is too complex to be solved completed by an individual senior programmer.

2.2 Knowledge management

Knowledge is stored in human's mind as a precious asset. As it is in human memory, when facing knowledge sharing, knowledge must be presented by some information. Programmers communicating with each other which is a knowledge-sharing example. In order to transfer the data, there should be a media like programming language. The knowledge provider must use the media to represent the knowledge he wants to send. If the knowledge receiver knows the same media, this communication is success, otherwise, this may result misunderstanding. During communication, the source and target information storage are people.

Knowledge can be shared in a network is divided by two types: tacit knowledge and explicit knowledge (Nonaka & Takeuchi 1995). Tacit knowledge is personal knowledge embedded in individual (Salis & Williams 2010). Tacit knowledge is too deeply rooted in mind to code and transfer. Explicit knowledge is tacit knowledge that has been documented in a particular form that can be easily shared and learned (Salis & Williams 2010). The process of transforming tacit knowledge into explicit knowledge is called *externalization*, and the process of learning explicit knowledge and transforming it into tacit knowledge is called *internalization* (Palmieri 2002). The externalization process involves codifying tacit knowledge into the form of documents, databases, tools, etc. The internalization process consists of training, processing, and practicing to learn explicit knowledge.

The knowledge embedded in employees is known as *knowledge stocks* (Jackson *et al*. 2003). Knowledge stocks are the basement of an organization for knowledge sharing. Effective management of the *knowledge flow* (Collins & Clark 2003) determines the key advantage of an organization. This relies on a social network including strongly related individuals who trust each other (Lepak & Snell 2007). In pair programming, the programmers are willing to share knowledge with the rest of the organization. The knowledge is constantly sharing among employees. In addition, two experienced knowledge workers combining in dyadic ties are more likely to achieve career success (Fliaster & Schloderer 2010).

Tacit knowledge sharing is more difficult than explicit knowledge. First is the willing of employees. Some employees resist sharing information with other people. Second, the most efficient way for sharing tacit knowledge is face-to-face communication. And communication becomes a key successful element to software industry (Stapel &

Schneider 2012). The core benefit of tacit knowledge sharing can't be replaced by other method. Face-to-face communication is actually the most efficient and informational method (Wright & Webb 2011). Furthermore, the mutual responsiveness of dyadic partners is a critical success factor to workers using pair programming for creative performance (Fliaster & Schloderer 2010).

Explicit knowledge sharing is easily achieved because the knowledge provider can describe the knowledge properly. The storage container which contains explicit knowledge can be called a document. To capture and reuse knowledge is a problem to any organizations. Explicit knowledge is re-usable and much easier to manage.

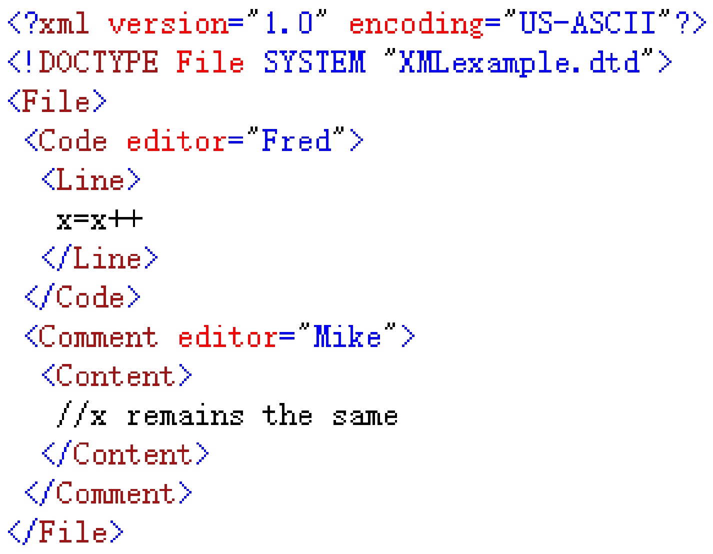
Knowledge management involves people and technology. It focuses on how an organization identifies, creates, captures, values, shares and applies knowledge. The objective of knowledge management is to improve the organization's performance by delivering needed knowledge to the right place at the perfect time (Levy & Hazzan 2009). Knowledge in a software company comprise of project documentation, software development strategies, employees' working experience, communication between employees and current market need. With well knowledge management, the organization is able to take the benefits of using new developing technology, and reusing existing knowledge (Palmieri 2002, Desouza *et al*. 2006, Willem & Scarbrough 2006).

Knowledge management systems are computerized tools for sharing and managing knowledge in organizations (Kankanhalli & Tan 2004). The main role of a knowledge management system is to help employees share knowledge. There are two different approaches for designing and utilizing knowledge management systems. Each model adopts a different knowledge management approach, *codification* for explicit knowledge and *personalization* for tacit knowledge. The codification approach focuses on the codification and reuse of explicit knowledge. Alavi & Leidner examined the use of Electronic Knowledge Repositories (EKR) for coding and sharing stored information (Alavi & Leidner 1999). A recent study shows the adoption of EKR depends on the users' motivation to knowledge sharing (He & Wei 2009). Another example is the Organizational Memory Information System (OMIS) that relates past explicit knowledge to the present in order to increase the efficiency of the organization (DeLone & McLean 1992). The personalization approach enhances the link between people to accelerate the transfer and sharing of tacit knowledge. An example is to provide contact information of expertise in an organization. Another example is to provide electronic forms for people who are dealing with similar projects or with similar interests to share knowledge with each other.

2.3 Extensible Markup Language (XML)

The eXtensible Markup Language (XML) is a formal and systematic rule for describing structured data. It is a standard framework used for data interchange over Internet (Abiteboul *et al.* 2000). Like HTML, this language is a subset of the Standard Generalized Markup Language (SGML). On 1998, XML (1.0) started to become a World Wide Web Consortium (W3C) recommendation (Yen *et al.* 2002). Nowadays a lot of open source tools combined with XML are available. The implementation of XML can even help verifying software and testing source code (Friedman-Hill 2001).

XML can annotate text in a distinguishable way. It uses character '<' in the beginning and '>' at the end to separate the markup from contents, which is also called a tag. For each data element, there is usually a pair of tags to consist it, start-tag and end-tag. Besides the empty element, the pair of tags with the content they enclose constitute an element. An empty element can be represented either by A data element can have attributes and corresponding values stored in the start-tag. XML tags can be customized by programmers and all elements in XML files are required using end-tag. Furthermore, quotation marks are necessary for attributes values in XML files. Figure 2.3 shows an example of a simple XML file.

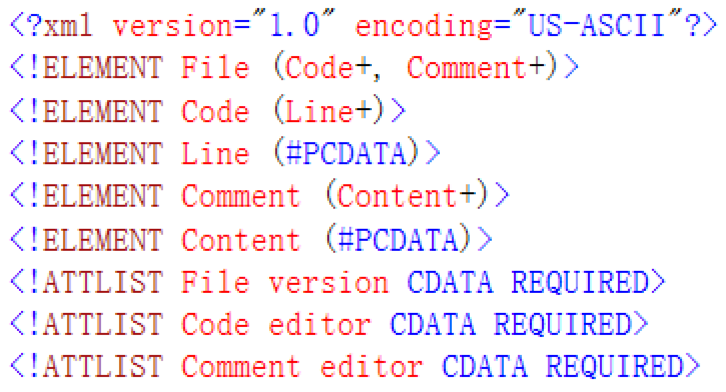


```
<?xml version="1.0" encoding="US-ASCII"?>
<!DOCTYPE File SYSTEM "XMLexample.dtd">
<File>
 <Code editor="Fred">
  <Line>
   x=x++
  </Line>
 </Code>
 <Comment editor="Mike">
  <Content>
   //x remains the same
  </Content>
 </Comment>
</File>
```

Figure 2.3 XML File Sample

By using XML, programmers can document information efficiently. As illustrated in Figure 2.3, both Code and Comment elements are nested in File element. The editors of two elements are shown in the attributes part of these two elements. With this format, XML can provide clearly descriptive markup to make the information more readable even for human user. On the other hand, compare to HTML, rendering the document content from a XML file will take more time, because the program must convert the descriptive markup at first. XML is good at data independence and data description. The stored data with procreate attribute can easily extract by programmers.

XML elements are nested in a hierarchical format following a Document Type Definition (DTD). All allowed tags and the relationships between tags of a XML file are listed in the corresponding DTD file. Figure 2.4 is an example of DTD file to XML file in Figure 2.3.



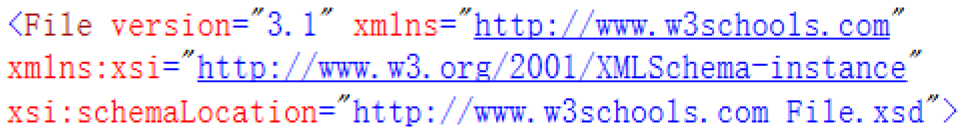
```
<?xml version="1.0" encoding="US-ASCII"?>
<!ELEMENT File (Code+, Comment+)>
<!ELEMENT Code (Line+)>
<!ELEMENT Line (#PCDATA)>
<!ELEMENT Comment (Content+)>
<!ELEMENT Content (#PCDATA)>
<!ATTLIST File version CDATA REQUIRED>
<!ATTLIST Code editor CDATA REQUIRED>
<!ATTLIST Comment editor CDATA REQUIRED>
```

Figure 2.4 DTD File Sample

Although the tags of XML files are not limited, all the tags are controlled by rules illustrated in DTD files. The most significant function of DTD is leading programmers creating proper new elements in XML. There are four kinds of markup declarations in DTD, element declaration, attribute declaration, entity and notation declaration. The element and attribute are important. In Figure 2.4, the elements correspond to all tags in Figure 2.3. This DTD defines the structure of allowed element name. In addition, DTD defines the contents of these elements. The root element is *File*, and only the element names *File*, *Code*, *Line*, *Comment* and *Content* are allowed in the XML file. In Figure 2.4, the attribute declarations (ATTLIST) define the data type of values correspond to attribute names. The value for attribute *version* can only be character string (CDATA). The notation #PCDTA stands for *Parsed Character Data*, which indicate the content enclosed in this tag is simply text. The value to attribute *version* is required for element *File*, which can add necessary illustration to this whole XML file. DTD files are created mainly for validating declarations of element type, attribute, entity and notation. DTD can declare an attribute name and the attribute value to it, but DTD files can't do specific constraint. Sometimes if the length of a value is limited, the syntax #PCDATA is too ambiguous to define that data type.

An alternative technique to DTD is XML Schema. XML schema is more expressive than DTD. Basically, DTD can define the relationship between elements, and XML Schema extents it to allow the detailed definition of the attribute types. Moreover, XML processor can also use XML Schema to validate XML document. Compare to DTD, XML Schema allows the programmer define the attribute values and element contents. It has a rich type system, which allows programmers to define restricted values for each of the elements and attributes in the XML file. It allows regular expression for constrain the format of a single value.

When to connect a schema file to the original XML file, the <!DOCTYPE> element in Figure 2.3 should be removed, and the <File> element should be substituted to:



```
<File version="3.1" xmlns="http://www.w3schools.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.w3schools.com File.xsd">
```

Figure 2.5 Changed XML Header

The extension to an XML schema file is "xsd". At the end of the element<File> indicate the name of the schema file used for constrain this XML file. Figure 2.6 is a schema file to the XML file in Figure 2.3. As shown in Figure 2.6, XML Schema can specify detailed type for attribute.

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <xsd:element name="File">
  <xsd:complexType>
   <xsd:sequence>
    <xsd:element name="Code">
     <xsd:complexType>
      <xsd:sequence>
       <xsd:element name="Line" type="xsd:string"/>
      </xsd:sequence>
      <xsd:attribute name="editor" type="xsd:string" />
     </xsd:complexType>
    </xsd:element>
    <xsd:element name="Comment">
     <xsd:complexType>
      <xsd:sequence>
       <xsd:element name="Content" type="xsd:string"/>
      </xsd:sequence>
      <xsd:attribute name="editor" type="xsd:string" />
     </xsd:complexType>
    </xsd:element>
   </xsd:sequence>
   <xsd:attribute name="version" type="xsd:string" />
  </xsd:complexType>
 </xsd:element>
```

Figure 2.6 An Example of the XML Schema

XML files are a useful tool for storing and illustrating data. To display the data properly, XML transducers is needed (Salminen & Tompa 2012). One of the technologies called XSLT can help programmer extract and render only the needed elements. XML Stylesheet Language (XSL) was developed for rendering XML files by W3C. The XSL Transformation language (XSLT) established in 1999 is used to transform XML documents into other format (Wadler 1999). XML files are well-formed tree structured files. The source file can be considered as a tree. An XSLT program is capable of pushing fragments from source tree to any location in the target tree or extracting it back. In this way, XSLT can even convert the XML syntax to a new HTML file, which can be seen on the webpage directly and have different structure.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <html>
      <body>
        <h1>File Presentation</h1>
        <table border="1">
          <tr>
            <td>File Version</td>
            <td>
              <xsl:value-of select="/File/@version"/>
            </td>
          </tr>
          <tr>
            <td>Code Editor</td>
            <td>
              <xsl:value-of select="/File/Code/@editor"/>
            </td>
          </tr>
          <tr>
            <td>Comment Editor</td>
            <td>
              <xsl:value-of select="/File/Comment/@editor"/>
            </td>
          </tr>
        </table>
        <p color="Blue">
          <xsl:value-of select="/File/Code/Line"/>
        </p>
        <p color="Green">
          <xsl:value-of select="/File/Comment/Content"/>
        </p>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

Figure 2.7 An Example of the XSLT File

Figure 2.7 is an XSLT file for displaying XML file in Figure 2.3. After rendering the original file, the file version, code editor and comment editor are displayed in a table. And the content of code is displayed in a blue font while the comment is green. As data stored in XML can be easily extracted, using XML for documentation can let the encoded file be easily examined by applications.

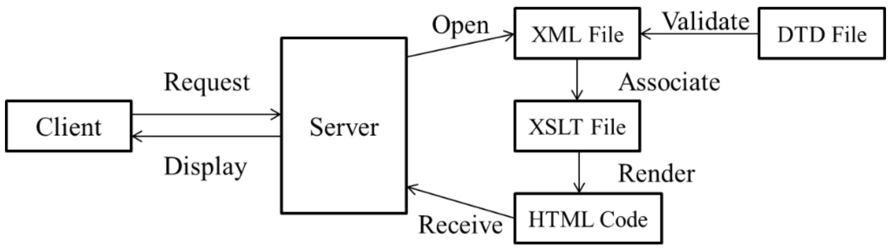


Figure 2.8 Web-based XML Editing System

2.4 Version Control for Software Source Code

Version control refers to technologies that help a software development team manage changes to source code over time. Version control system keeps track of every modification to the source code in a special kind of database. Programmers can compare earlier versions of the source code to help fix the mistake while minimizing disruption to all team members. For software development projects, the source code is a repository of the invaluable knowledge and understanding about the problem domain that the programmers have collected and refined through careful effort. Version control is able to protect source code from both catastrophe and the casual degradation of human error and unintended consequences. Moreover, version control is also able to help software development teams prevent concurrent work from conflicting. Changes made in one part of the software can be incompatible with those made by another programmer working at the same time. A good version control system facilitates a smooth and continuous flow of changes to the source code rather than locking and preventing changes to the source code at the expense of blocking the progress of others. Software development teams that do not use version control often run into problems of not knowing which changes that have been made or the creation of incompatible changes between two unrelated pieces of code. Hence, programming without using version control is risky, like not having backups.

The use of version control for source code enables the traceability of a complete long-term change history of every program. The history record often includes the author, date and written documents on the purpose of each change. Having the complete history enables going back to previous versions for debugging purposes. Having the annotated history of the source code can enable programmers to make correct and harmonious changes that are in accord with the intended long-term design of the system.

There are two different approaches for version control, the *centralized* model and *distributed* model.

In a *centralized* model, a central repository is used for users reading and writing files. The only central repository is actually a data server stored all tracked files and their history (Collins-Sussman *et al*. 2004). The newest revision of central repository is often called HEAD and is the most convenient file to read (Otte 2009). User can retrieve a working copy to their local computers range from single files to the entail repository. They commit changes to and update working copies from an online repository, without manually keeping too many copies of files locally.

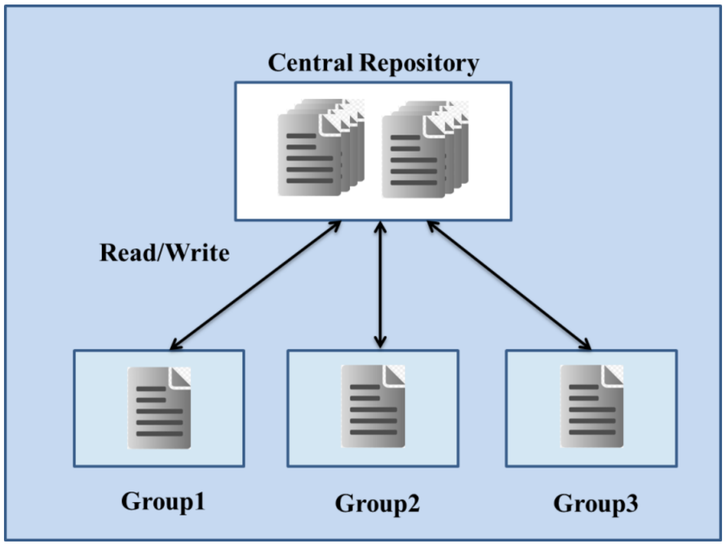


Figure 2.9 Centralized Version Control System Model

As the example of Figure 2.9, by using LAN and WAN, the central repository can be accessed from anywhere of the world by different groups of developers.

In a *distributed* model, a central sever contains repository to programmers are not necessary. As a new approach to Version Control Systems, every user has a complete repository on their own development terminal. A local repository reduces the development teams' reliance on Internet. Users are able to work more flexible and completely offline (Otte 2009). However, sharing between different workstations still needs communication through Internet.

Figure 2.10 shows an example of the distributed model. Every local repository on programmers' computer is independent from all other local repositories. As a result, communicating between local and remote repositories is much more complicated than centralized version control system. By systematic recording the entire editing history and branches information, distributed version control systems have better merge capacity. According to its flexibility, it is suitable for many development methods (Otte 2009).

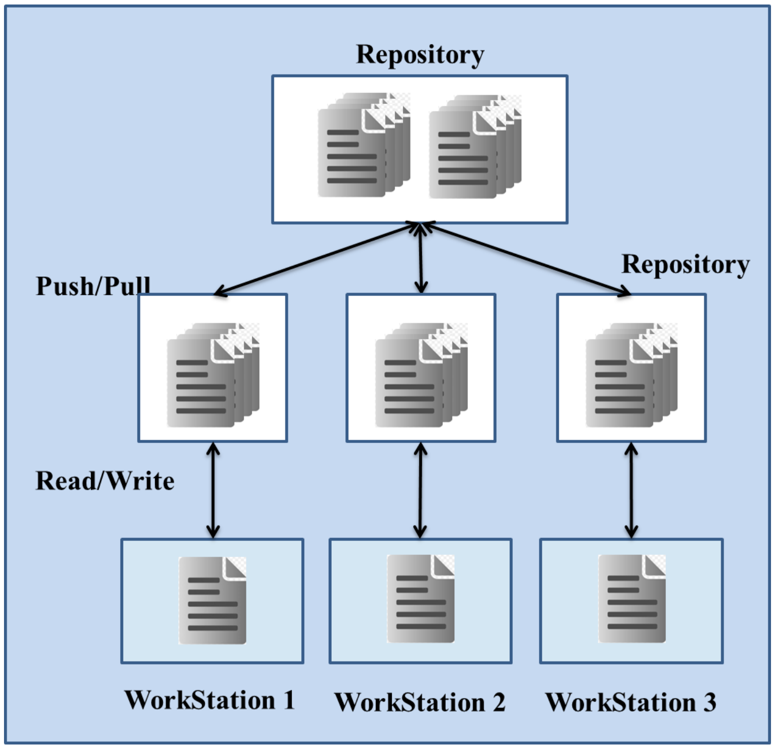


Figure 2.10 Distributed Version Control System Model

Chapter 3
The System Architecture and System Requirements

3.1 A Knowledge Management Framework for Agile Software Development

As Agile software development lacks of documentation and knowledge sharing in the software development process, we propose a knowledge management framework for creating, managing and sharing various types of documents for agile software development. In our knowledge management approach, programmers are divided into two groups, *masters* and *apprentices*. Masters are senior software developers who are primarily responsible for programming. Masters avoid document writing as advocated in the agile manifesto for software development. Apprentices are junior software developers who are mainly responsible for creating, managing and sharing various types of documents for the agile software development process.

The advantages of our proposed knowledge management are obvious. First, various types of documents will be produced and sharing in the agile software development process. In case of an employee turn-over, the new employee will be able to learn the existing work through the documents. Moreover, valuable knowledge will be able to easily searched and shared among all team members of the software development team. Furthermore, as apprentices are normally expected to acquire more experience, code reviews and documentation can improve the learning experience. Document sharing also help apprentices and other junior programmers get familiar with the software development process. We strongly believe that the costs associated with apprentices can be well justified by the benefits associated with knowledge sharing in the agile software development process.

Using the aforementioned framework, we developed a system called “Unite Portal” is the name for knowledge management for agile software development. It integrates the Kanban and Subversion technologies for software development, uses XML and object-relation databases to store and share data. As designed as a web portal, this system is easily accessed through web browsers and is helpful to user management.

3.2 System Overview

The main idea of this system is to build a user-friendly, management-conveniently and access-easily web portal that could help managing version and support collaboration for development teams adopting Agile Software Development. The hardware and software system architectures of Unite Portal are illustrated in Figure 3.1 and Figure 3.2 respectively. The system includes a user management module, a Kanban system module, a version control module and an XML file management module.

The users of this system are supposed to be programmers involved in agile programming projects. This system emphasizes communications and collaborations between programmers (i.e., users). It allows the programmers in a same team access to and edit same project easily and simultaneously. With the file management module, the system is able to display all available versions of programs and documents to different users.

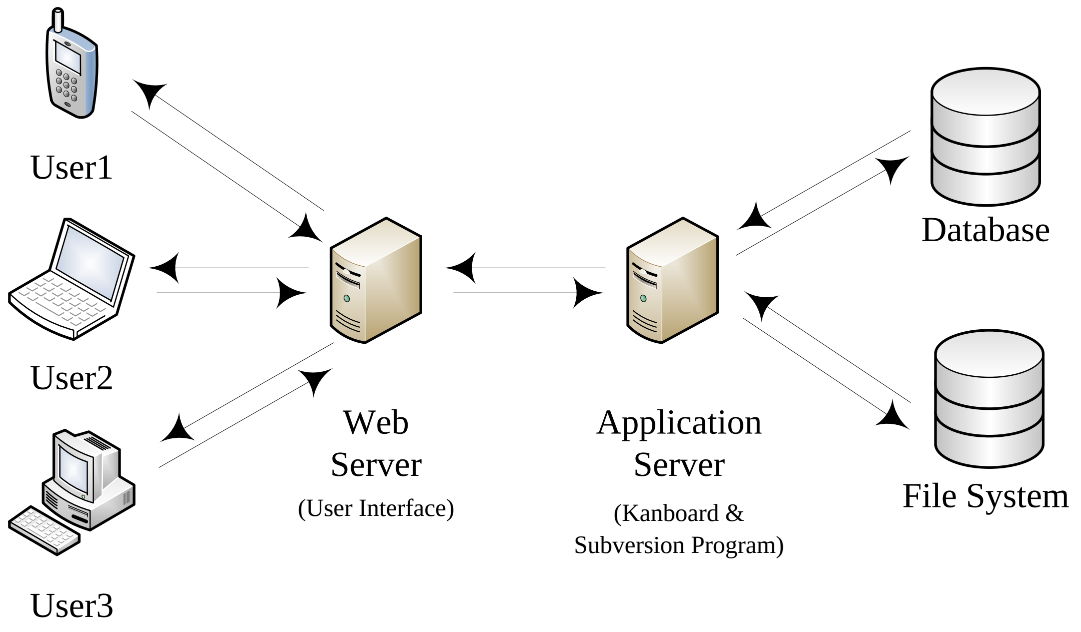


Figure 3.1 Hardware System Architecture of Unite Portal

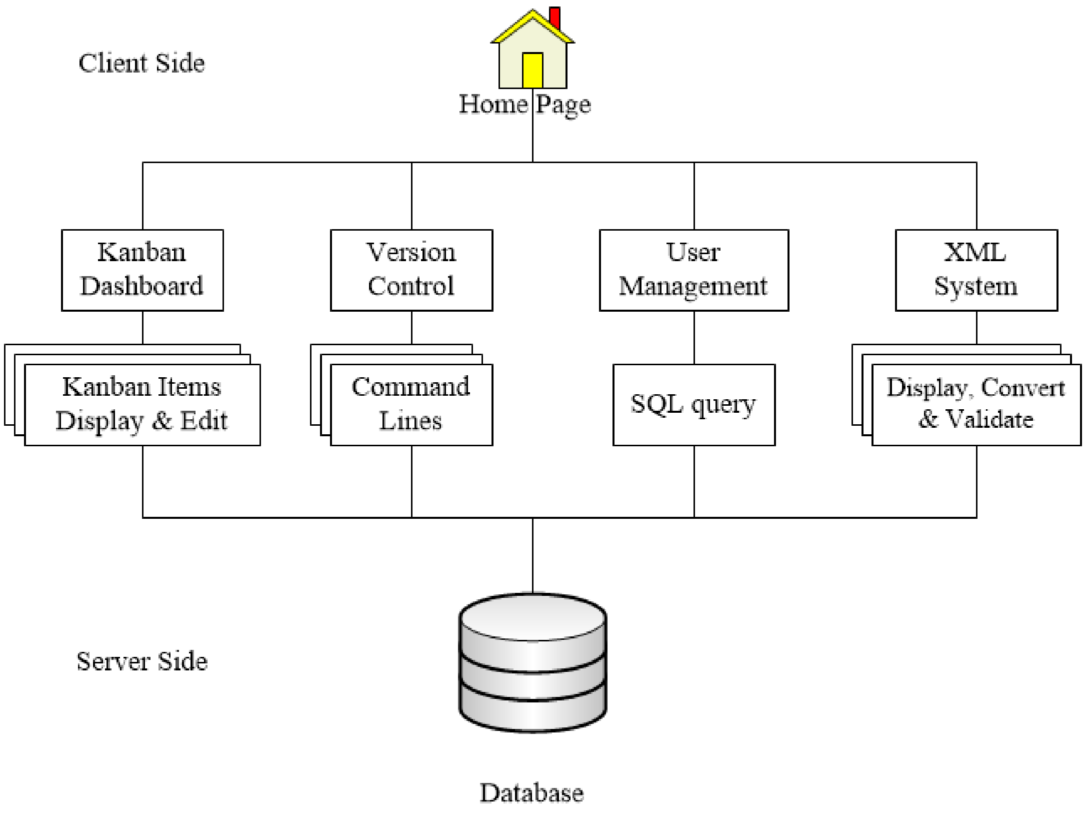


Figure 3.2 Software System Architecture of Unite Portal

The major features of this system are listed as follows:

- **Customizable Kanban boards for different groups of user:** Each user can select preferred templates for displaying their own Kanban boards and customize the status of Kanban boards.
- **Task management:** Programmers are able to create, edit and share tasks among development teams. After sharing, multiple programmers can contribute to single task at the same time. Tasks can also be deleted if needed.
- **File version control:** All versions of documents and programs related to a project can be retrieved once they are stored on Unite Portal.
- **XML file management:** All documents stored on Unite Portal are converted to XML files and dynamically displayed on web browser.

3.3 System Architecture

3.3.1 System Requirements

The functional requirements of Unite Portal are listed in Table 3.1.

Functional Requirements
The system can be visited online through web browser.
The system should be fitted to mainly used web browsers.
The users' information and projects' content should be store in a secure database.
The users need to go through registration and login function to visit our system.
The system must support version control tool.
The system must embed a Kanban Board creating and storing system.

Functional Requirements
The users can create, edit, share or delete their projects.
The users can personize their Kanban Board's outlook.
The users can edit and share their boards.
A user management system should be added in order to support different levels of authorizations of project contents.
This system should be easily and efficiently to use.
This system can enable users transform the original source code to XML style files and display those files on browser.
This system can allow users modify the displayed XML files and save the changed XML files to version control system.
A mail server and online chat room for communication between team members.

Table 3.1 Functional Requirements

3.3.2 Prgramming Languages

We used HTML, CSS, JavaScript, jQuery, PHP for implementing Unite Portal. HTML and CSS were used for interface design. JavaScript and jQuery were used for client-side programming. PHP was used for server-side programming

3.3.3 System Components

Unite Portal is composed by four main parts, Kanban board system, version control system, User management system and XML file management system. The portal displayed information in two ways, static HTML web page and dynamic HTML web page. The static parts of our web portal are directly written in HTML format. These parts of code are delivered to users as it is stored in server. For the dynamic contents, our portal uses PHP language to retrieve, store data and dynamic display.

Figure 3.3 illustrates the workflow of these four components. The Kanban board module displays the visualized tasks and allowed user to concurrently modify. It enabled users to optimize tasks efficiency. The version control module helps users store current and history versions of documents and programs. The user management module is used for managing users' authorization and authentication. It enables administrators to change users' authorization level in order to guarantee the security to our portal. The XML file management module is designed for displaying, editing and storing documents and programs.

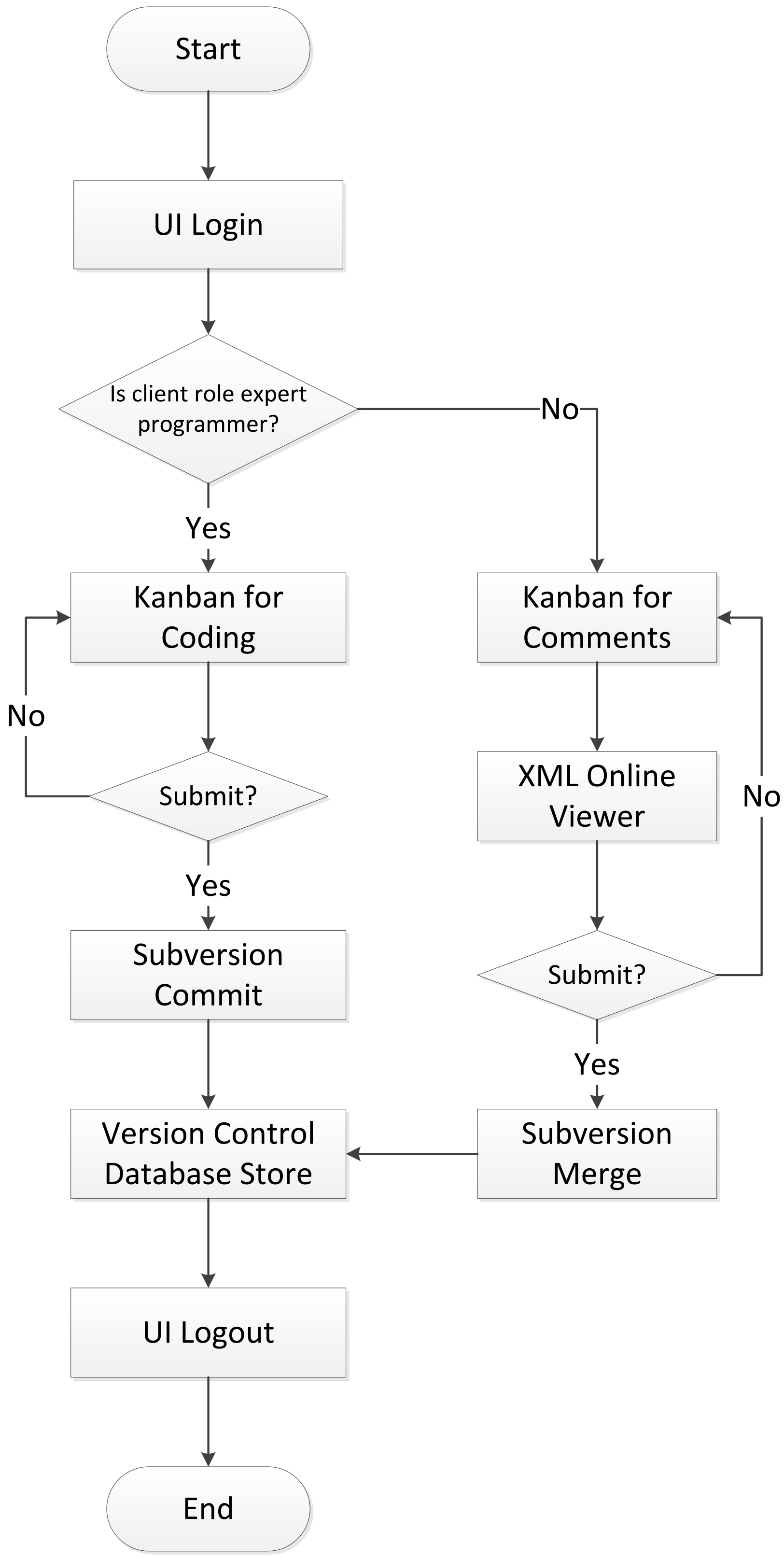


Figure 3.3 The workflow of Unite Portal

Different types of users can be identified by the user management module through the login process. Two types of Kanban Board with default setting are displayed to corresponding groups of users. For masters, version control can help users to create local working copies and project commitments. For apprentices, XML online viewer together with the version control module help project comments editing and version merging operations.

Figure 3.4 illustrates the overall system architecture.

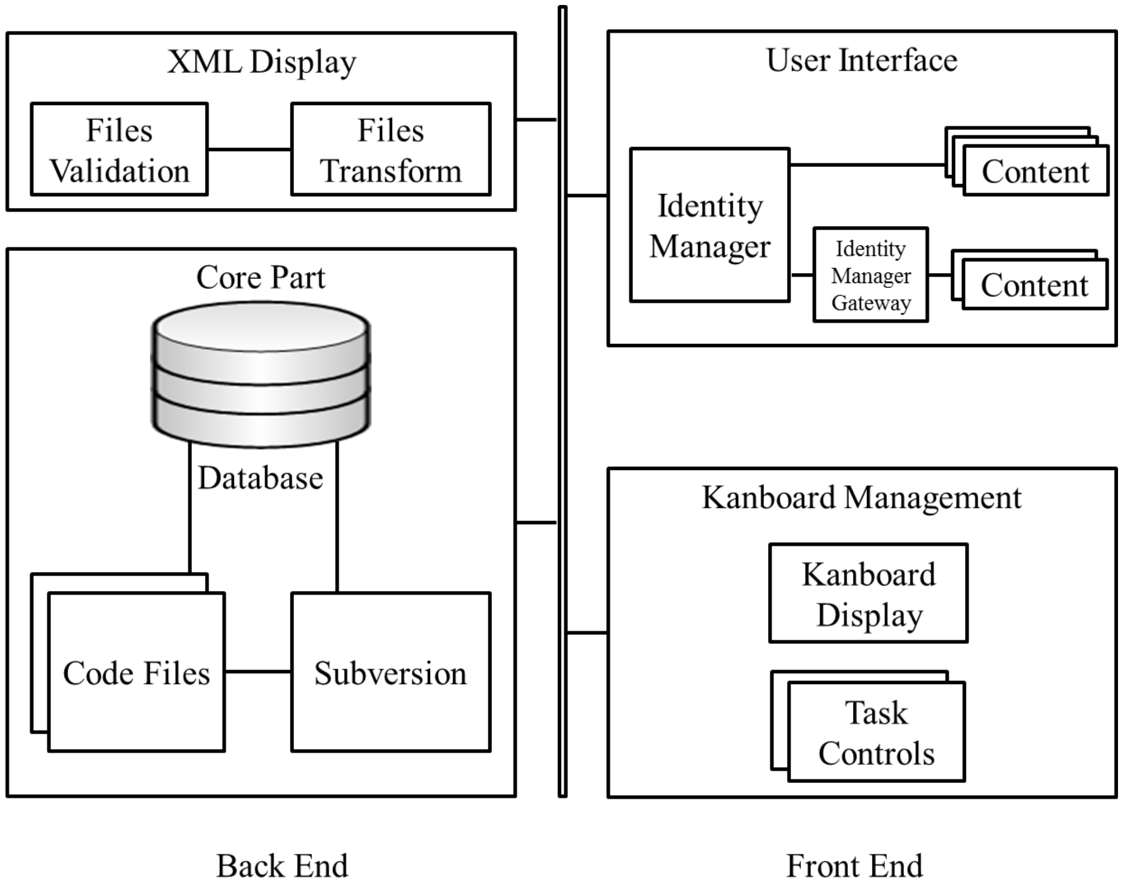
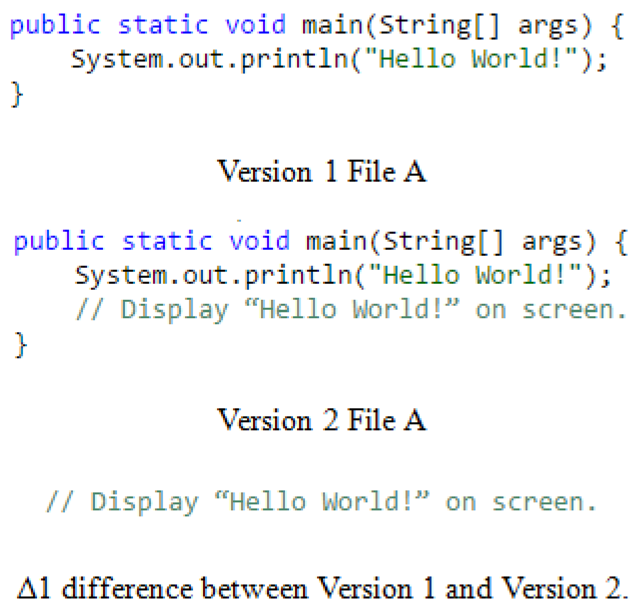


Figure 3.4 Overall System Architecture

3.4 Version Control

All current commercial and open-source version control systems use delta encoding to store different versions of a file. Unite Portal also uses delta encoding instead of wasting space on duplicated contents between versions. It means our system only store the difference between the current version and the previous latest version.

Figure 3.5 shows an example of delta encoding.



```
public static void main(String[] args) {
    System.out.println("Hello World!");
}
```

Version 1 File A

```
public static void main(String[] args) {
    System.out.println("Hello World!");
    // Display "Hello World!" on screen.
}
```

Version 2 File A

```
// Display "Hello World!" on screen.
```

Δ1 difference between Version 1 and Version 2.

Figure 3.5 An example of delta encoding

In Figure 3.5, this difference between the two versions is represented as "Δ1". The delta encoding method can save a lot of space to store different versions of files.

Our system includes a branch-based versioning model. It is based on a popular version control system called Apache Subversion (SVN) (Collins-Sussman *et al.* 2004). Our system enables several programmers to work with same program source code file at the same time. In order to achieve this functionality, the repositories to each project must have one or more branches for users collaborating at the same time.

As a well-defined branch-based versioning model, user can create a new branch as a specific version from an existing branch. This new branch could be still in the same project or be saved as a primary branch in a new project (Ekanayake *et al.*, 2011). The primary branch includes the first version and the versions come after the first version created in a project. Non-primary branches of a project are branches derived from and paralleling with the primary branch. One important element of version control system called current version is the latest version of a branch.

When a user is editing with an existing version, once she saves the modification, a new version is created and store in the same branch. The saved version becomes the current version. With the project name, branch name and version number, the version control module can easily find out the wanted version. The model of our system is shown in Figure 3.6 and Figure 3.7, as a project using our proposed knowledge management framework for agile programming.

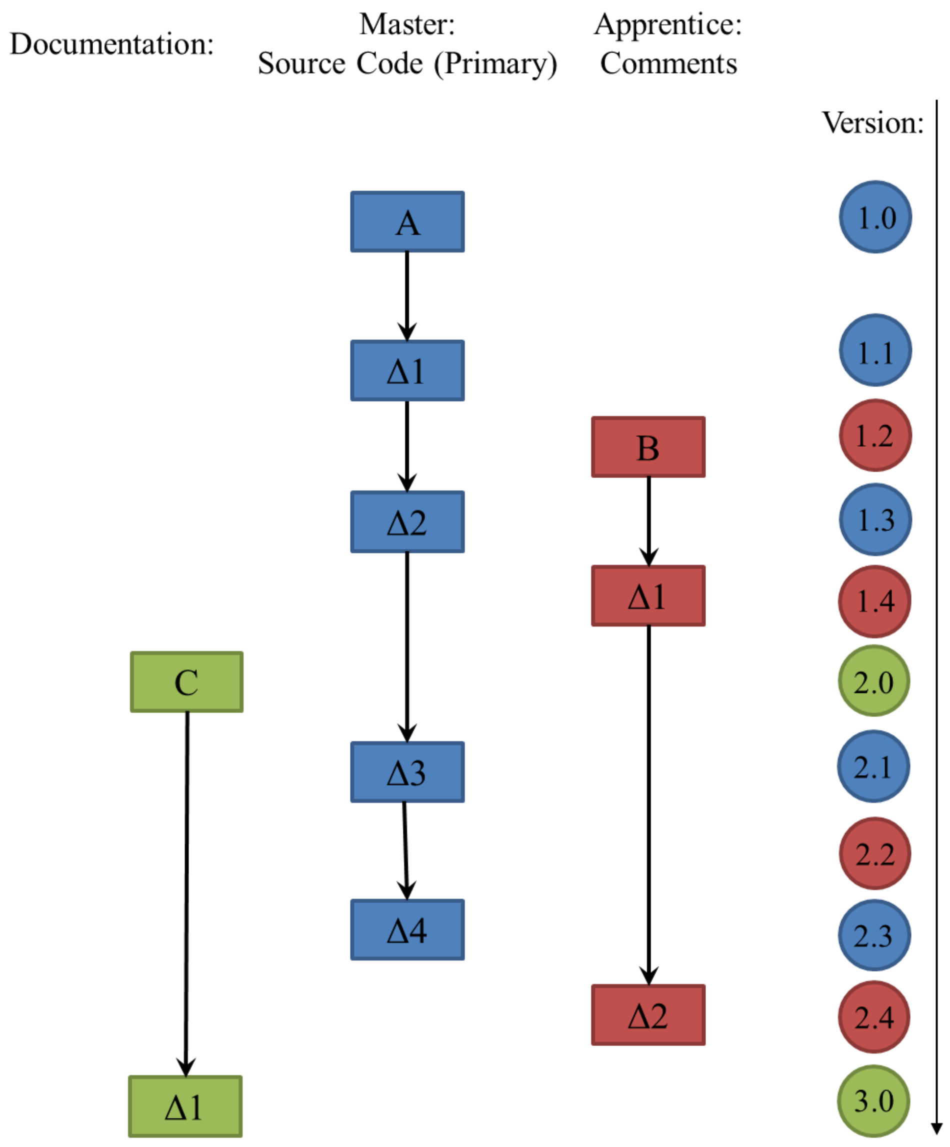


Figure 3.6 Delta Encoding for the Version Control of Our Proposed Knowledge Management Framework for Agile Programming

Figure 3.6 shows the delta encoding of the version control of our proposed knowledge management framework for agile programming, and Figure 3.7 shows the versions from a user's point of view. In Figure 3.6 and Figure 3.7, the branches are defined as "Master", "Apprentice" and "Documentation". In our model, the primary branch belongs to the master. The apprentice and documentation branches are secondary branches. Version 1.2 which derived from version 1.1 is the first version of the apprentice branch. Version 2.0 and Version 3.0 are completed iteration outputs, which are also well-documented and ready to release. Masters are mainly doing programming work in a primary branch. Meanwhile, apprentices are able to check out working copy from the primary branch and focusing the documentation task on their own branch. Then version control system will automatically merge the apprentices' work into the primary branch.

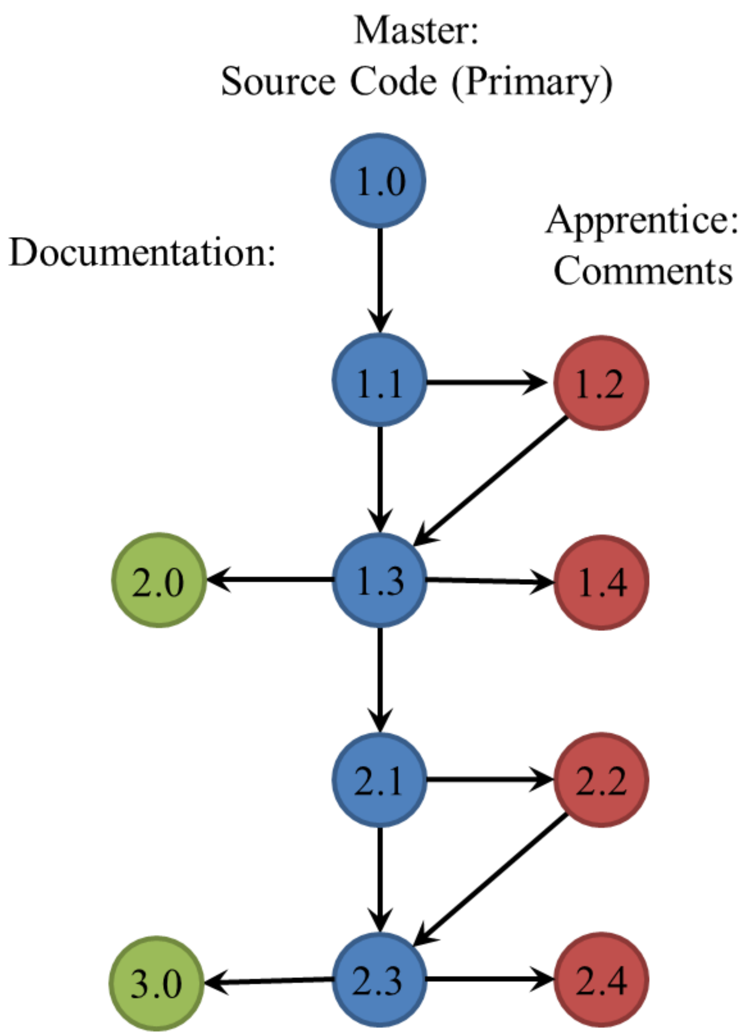


Figure 3.7 Actual Versions of Our Proposed Knowledge Management Framework for Agile Programming

Chapter 4
System Design

In this chapter, we explain the design of our prototype Unite Portal, which enables knowledge management and sharing for the agile software development process. The chapter is divided in two parts. The first part states the design of the database used for our system, and explains the detailed data model adopted for storing important data for different users. The second part illustrates important sections of our web portal, including user interface design and dynamic system responses to different user inputs.

The main purpose of our web portal is to build a collaboration tool supporting an agile software developing team working in a unique platform. The portal must be easily accessed from any web browser and all mobile screens.

4.1 Database Design

A fully functional database is the key to a successful web portal. A good database is able to store large amount of user information and project data in an organized format, allowing users easily access data through operations from web browser. In addition, the database must be able to extend easily when additional application requirements added. We chose MySQL as our Database management system to build our web portal infrastructure, which is the most popular open source database and is compatible with all operating systems.

4.1.1 Data Model

The tool used in this section to illustrate the conceptual model of our database is "Entity – Relationship model"; it uses graphical diagrams showing flow of data and information. Our web portal works as a tool supporting communication between programmers and customers. All data used for our portal are separated in three parts, developing programmers, project owners and projects in progress. The design of our database is shown in Figure 4.1.

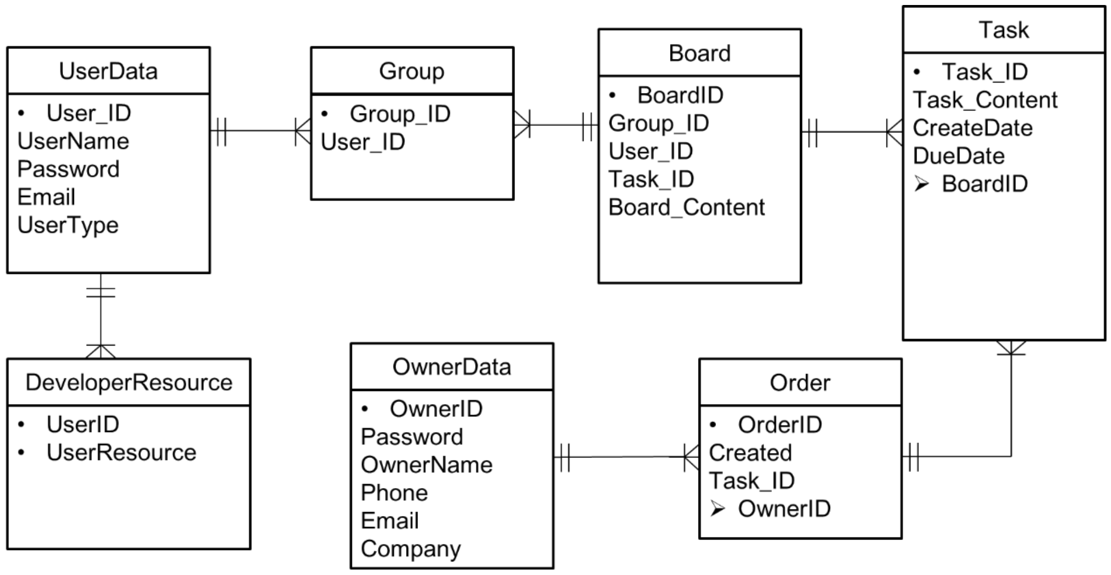


Figure 4.1 Database Schema

Our database contains seven tables. The "*UserData*" and "*OwnerData*" tables are used to store registration information for two groups of users, programmers and project owners. The "*DeveloperResource*" table helps users distinguishing junior and senior programmers and arranging different authorities to each programmer. For letting different groups of users access wanted Kanban Boards, The "*Group*" and "*Board*" tables are used to store development teams' and Kanban boards' information. These two joint tables help system managing multiple development teams and setting access authorization for

Kanban boards. The "*Task*" table stores information of minor projects on work and has a relationship with the "*Board*" table. The "*OwnerData*" table has a "one-to-many" relationship with the "*Order*" table, which has also has a "one-to-many" relationship with the "*Task*" table. These relationships enable project owners checking working status easily.

4.1.2 Database Schema

The "*UserData*" table stores registration information for all developers, which has five fields, User_ID, Username, Password, Email and UserType. This table is used to identify developers. For example, the UserType field is used for distinguish junior programmers, senior programmers and Scrum masters. In the "*UserData*" table, all fields can't have NULL value.

UserData				
Field name	SQL data type	Data type	Field Description	Key
User_ID	VARCHAR(20)	Text String	User Identifier	PRI
UserName	VARCHAR(15)	Text String	Login Username	
Password	VARCHAR(30)	Text String	Login Password	
Email	VARCHAR(50)	Text String	User Email	
UserType	VARCHAR(15)	Text String	Developer Type	

Table 4.1 UserData Table

The "*Board*" table stores all Kanban boards' information and status, including personized settings like preset WIP numbers for different working phrases. This table helps developers check working status of projects on work. Each Kanban board can be visited by multiple users. Boards could also be shared among users and even groups. The

User_ID and Group_ID fields help managers define whether a user or a group of users is allowed to visit a Kanban board.

Board				
Field name	SQL data type	Data type	Field Description	Key
Board_ID	VARCHAR(50)	Text String	Kanboard Identifier	PRI
Group_ID	VARCHAR(20)	Text String	Group Identifier	
User_ID	VARCHAR(20)	Text String	User Identifier	
TaskID	VARCHAR(50)	Text String	Project Task Identifier	
Board_Content	VARCHAR(255)	Text String	WIP data, Board status	

Table 4.2 Board Table

Each Kanban board represents a project in progress, which may contain many mini tasks. The "*Task*" table is used for storing all tasks' information. This table has five attributes corresponding tasks' ID, content, creating date, due date and the belonging board. The TaskContent field stores all important information of each task, like creator information, latest version and user who modify this task most recently.

Task				
Field name	SQL data type	Data type	Field Description	Key
Task_ID	VARCHAR(50)	Text String	Project Task Identifier	PRI
TaskContent	VARCHAR(255)	Text String	Project Task Content	
CreateDate	DATE	2015-08-01	Task Create Date	
DueDate	DATE	2015-12-31	Task Deadline	
Board_ID	VARCHAR(50)	Text String	Kanban Identifier	FK

Table 4.3 Task Table

For convenient management, users are grouped with unique Group_ID. Two more tables "*Group*" and "*DeveloperResource*" are used for grouping developers. These two tables also help managers assigning authorization to general users. Only authorized users can see certain web portal sections.

Group				
Field name	SQL data type	Data type	Field Description	Key
Group_ID	VARCHAR(20)	Text String	Group Identifier	PRI
User_ID	VARCHAR(20)	Text String	User Identifier	

Table 4.4 Group Table

DeveloperResource				
Field name	SQL data type	Data type	Field Description	Key
UserID	VARCHAR(20)	Text String	User Identifier	PRI
UserResource	VARCHAR(20)	Text String	User Authorized Resources	PRI

Table 4.5 DeveloperResource Table

Apart from these tables designed for programmers, two more tables are prepared only for project owners. The "*Customer*" table stores clients' information. This table has a one-to-many relationship with the "*Order*" table, which stores all orders' status.

OwnerData				
Field name	SQL data type	Data type	Field Description	Key
OwnerID	VARCHAR(20)	Text String	Customer Login Identifier	PRI
Password	VARCHAR(30)	Text String	Login Password	
OwnerName	VARCHAR(50)	Text String	Customer Name	
Phone	VARCHAR(20)	Text String	Phone Number	
Email	VARCHAR(50)	Text String	Email Address	
Company	VARCHAR(50)	Text String	Company Name	

Table 4.6 Customer Table

Order				
Field name	SQL data type	Data type	Field Description	Key
OrderID	VARCHAR(20)	Text String	User Identifier	PRI
Created	DATETIME	2015-05-08 12:35:29.123	Order Created Date	
TaskID	VARCHAR(50)	Text String	Project Task Identifier	
OwnerID	VARCHAR(20)	Text String	Customer Login Identifier(Owner)	FK

Table 4.7 Order Table

4.2 User Interface Design

For building our web portal, we chose HTML to create web pages; CSS to define web page style and JavaScript to perform client-side validation. The language used to connect database is PHP 5, which is a server-side scripting language, ideally for web portal development. In the following section, we are going to illustrate how our web portal works.

4.2.1 Login and Registration

A login form is the only available window displayed to the first time user. In order to access other functions in this system, users must successfully log in. The back-end database, which stores users' registration information, automatically validate user's input and direct user to main page - dashboard.

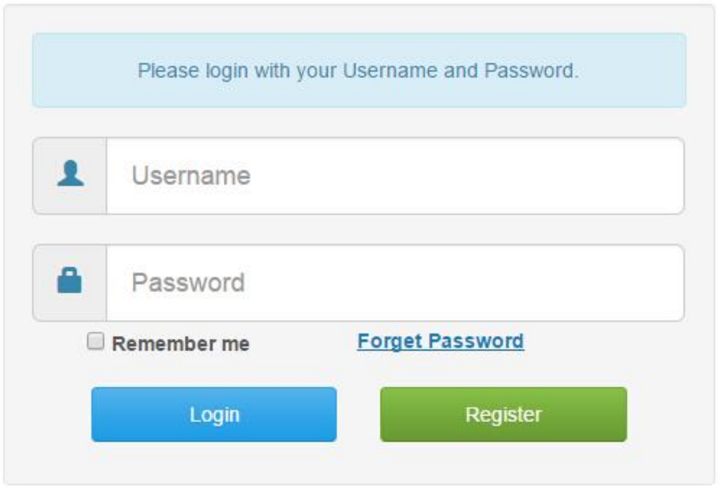


Figure 4.2 Login Form

For first time users, there is a button for them to sign up. The registration form asks user type in necessary information, including real name, username, email address, and personalized password.

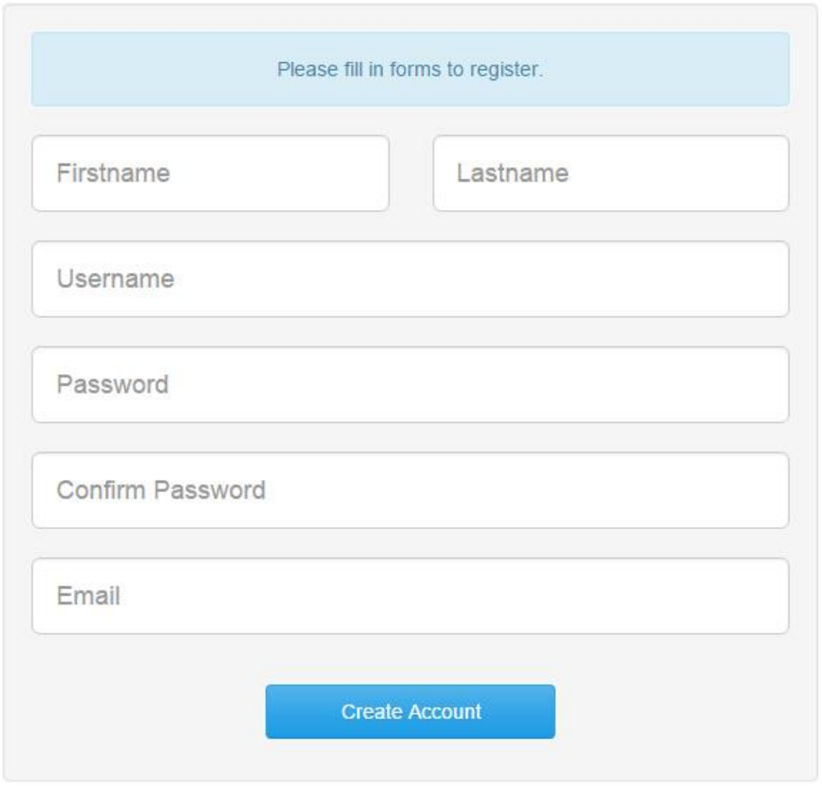
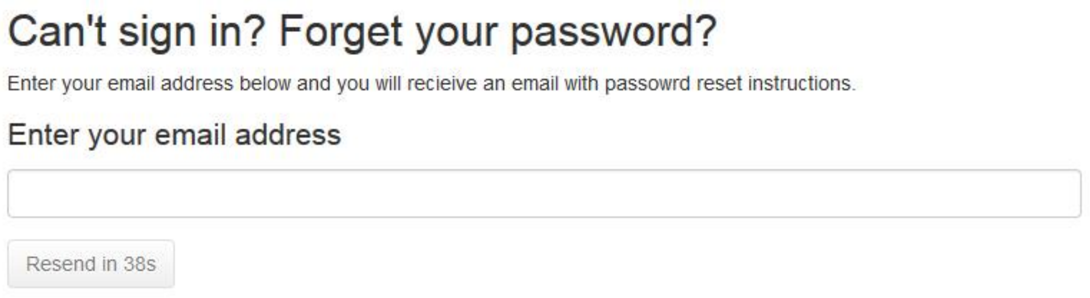


Figure 4.3 Registration Form

If a user forgot his login password, our system supplies a password reset function. Users are able to reset their password in a new web page by clicking "Forget Password" button at the login form. After entering the email address used for registration, the user will receive an email with instructions if the entered email address is the same as the one stored in system database. The validation and email sending functions are running at the back-end server side.



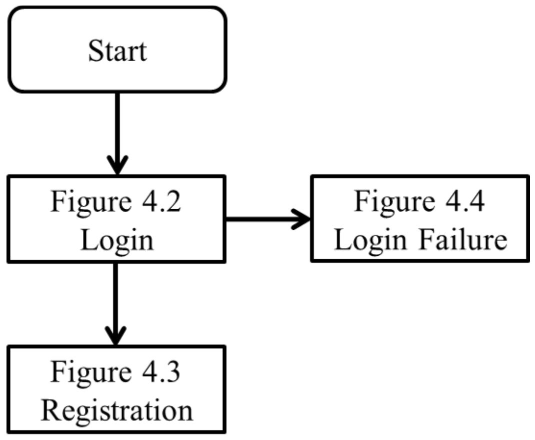
Can't sign in? Forget your password?

Enter your email address below and you will recieive an email with passowrd reset instructions.

Enter your email address

Figure 4.4 Password Retrieving

The login process is illustrated in Workflow 4.1.

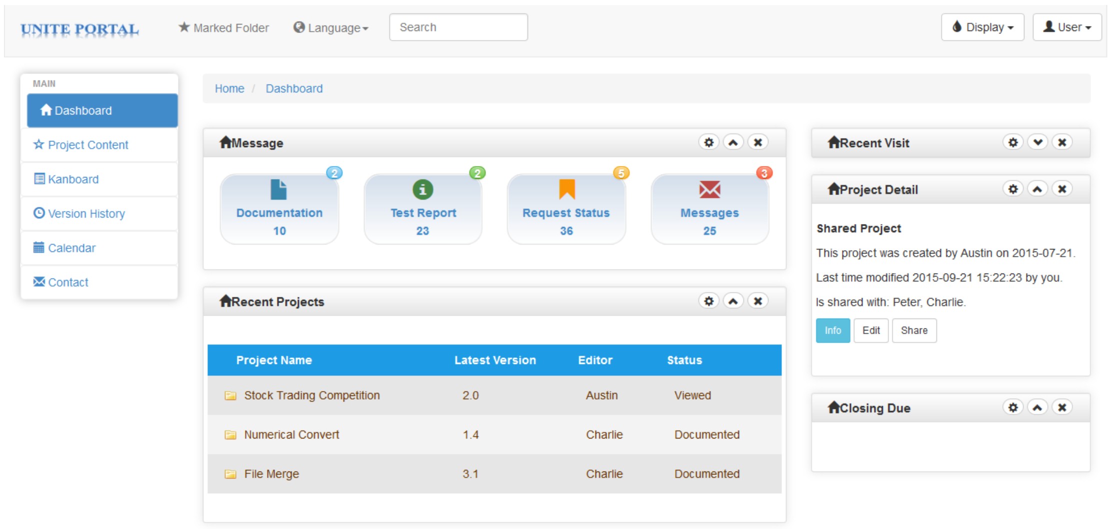


$$\begin{tikzcd}
\text{Start} \arrow[d] & \\
\text{Figure 4.2 Login} \arrow[r] \arrow[d] & \text{Figure 4.4 Login Failure} \\
\text{Figure 4.3 Registration} &
\end{tikzcd}$$

Workflow 4.1 Login Process

4.2.2 User Dashboard

The web page displayed after successful login is the Unite Portal dashboard; it contains navigation buttons, message center, personal setting and status of projects in progress.

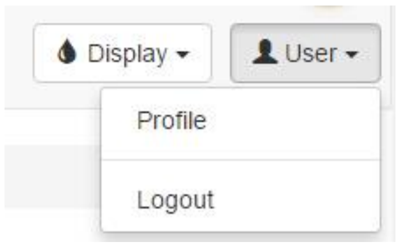


Project Name	Latest Version	Editor	Status
Stock Trading Competition	2.0	Austin	Viewed
Numerical Convert	1.4	Charlie	Documented
File Merge	3.1	Charlie	Documented

Shared Project
This project was created by Austin on 2015-07-21.
Last time modified 2015-09-21 15:22:23 by you.
Is shared with: Peter, Charlie.

Figure 4.5 Dashboard

The “Display” and “User” buttons are located at the top right corner of the dashboard. The “Display” button enables users to set web portal display templates. The “User” button is designed for user profile management and logout. Users are able to close the current session by clicking the “Logout” button. Then they will be directed to the initial login form.



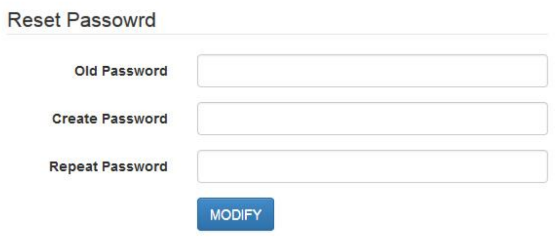
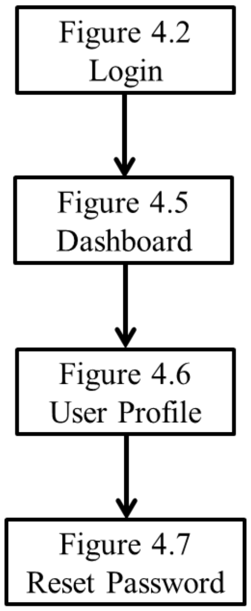


Figure 4.6 User Profile

Figure 4.7 Reset Password

Changing password regularly is a helpful way to keeping security. This function is hidden in the "Profile" button of the showing dropdown list in the "User" button.

The functions for the user dashboard are illustrated in Workflow 4.2.

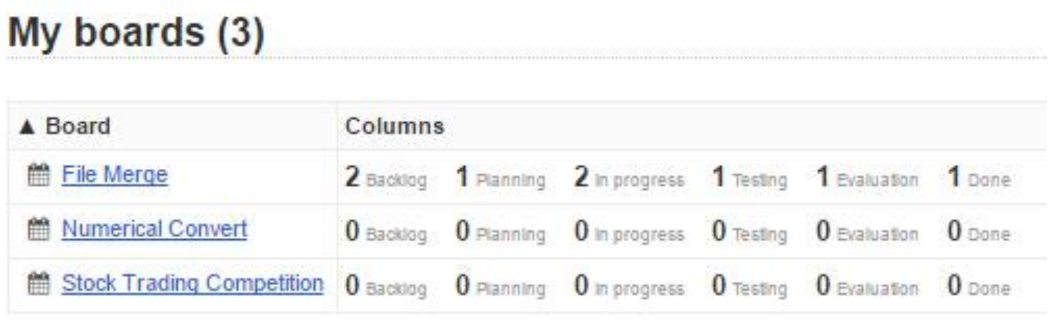


Workflow 4.2 Dashboard Features

4.2.3 Kanban Board

Following is an important part of our thesis, the Kanban board.

The table "My boards" shows up after users clicking the "Kanban Board" button in the left-side menu of dashboard. This table lists all the boards' information the user involved in. These boards are ordered by their names. The current working status of each board is also displayed in this table.



My boards (3)

▲ Board	Columns					
File Merge	2 Backlog	1 Planning	2 In progress	1 Testing	1 Evaluation	1 Done
Numerical Convert	0 Backlog	0 Planning	0 In progress	0 Testing	0 Evaluation	0 Done
Stock Trading Competition	0 Backlog	0 Planning	0 In progress	0 Testing	0 Evaluation	0 Done

Figure 4.8 Board Preview

The "New Kanban" window only shows on Scrum masters' web browser. Only Scrum masters are authorized to create new Kanban boards and make changes to Kanban boards' settings.

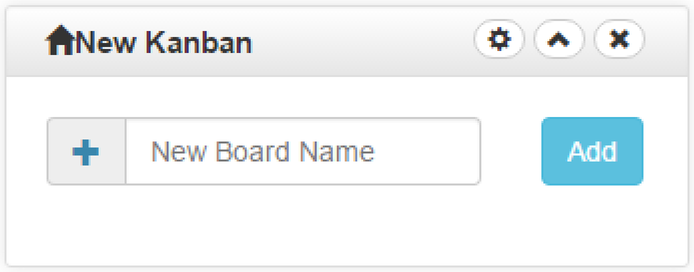
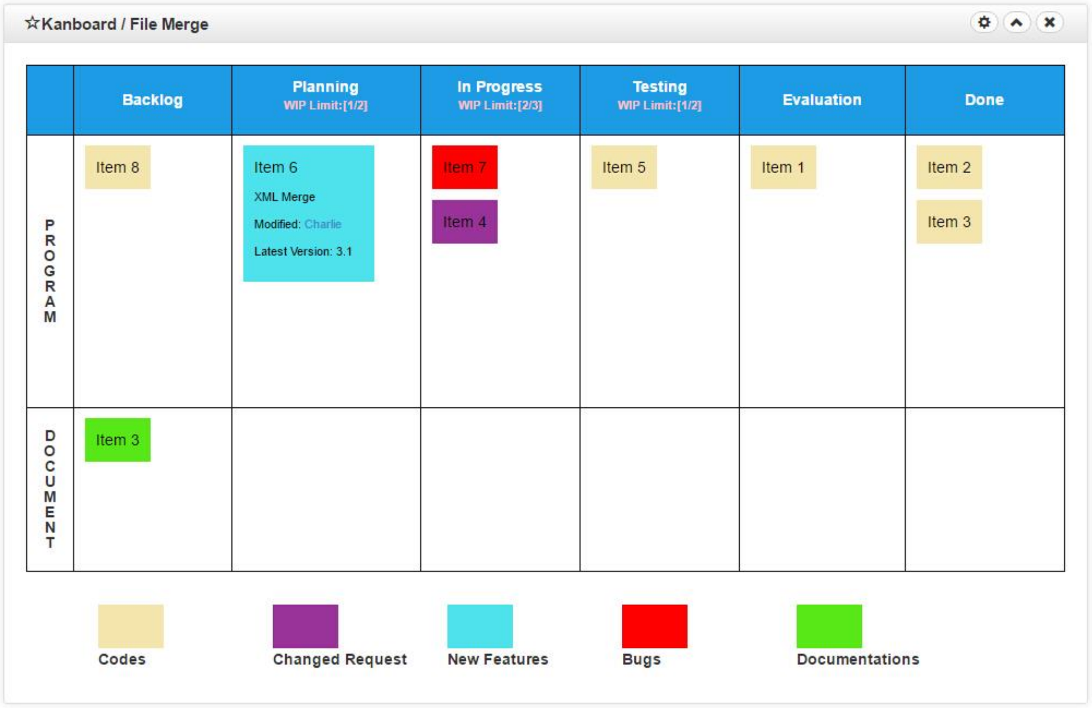


Figure 4.9 Create New Kanban Board

After selecting a board from the previous "My Board" table, users will be directed to a new web page vender by our web portal. Figure 4.10 is a sample of the Kanban board for project "File Merge". In this Kanban board, there are six columns representing six different working phases. Tasks in this project are named as "Items" followed by sequential numbers. Different types of tasks are marked with different colors. Another feature of our Kanban board system is the preset WIP numbers. These numbers locate beneath to the name of different working phases and restrict the number of items allowed in corresponding columns.



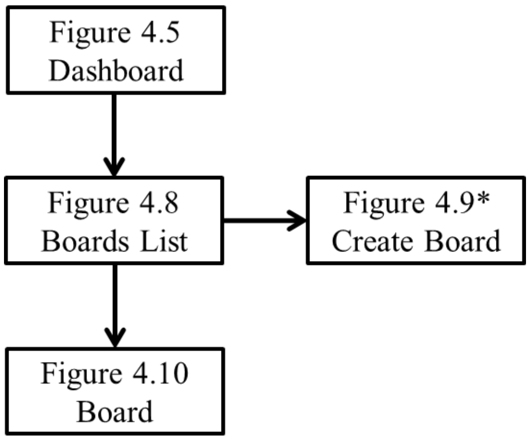
☆Kanboard / File Merge

	Backlog	Planning WIP Limit:[1/2]	In Progress WIP Limit:[2/3]	Testing WIP Limit:[1/2]	Evaluation	Done
PROGRAM	Item 8	Item 6 XML Merge Modified: Charlie Latest Version: 3.1	Item 7 Item 4	Item 5	Item 1	Item 2 Item 3
DOCUMENT	Item 3					

🟨	Codes
🟪	Changed Request
🟦	New Features
🟥	Bugs
🟩	Documentations

Figure 4.10 Kanban Board Sample

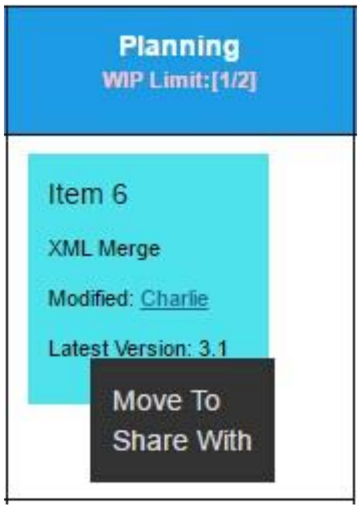
The board management process is illustrated in Workflow 4.3.

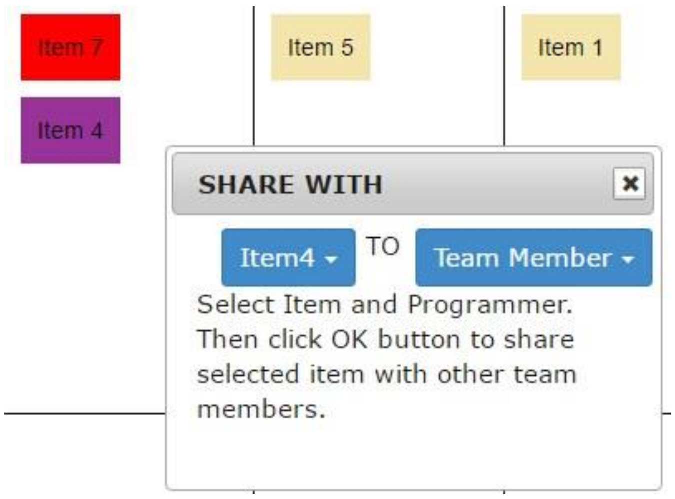


Workflow 4.3 Board Management System

4.2.4 Task Management

As mentioned in the previous section, tasks are displayed as small colorful blocks in Kanban board viewer. When users move their mouse over a task, a larger block will toggle out, containing further information about this item. For additional tasks' management, users are able to move or share tasks by right-click on their icons. The popped out "Share With" window help developers sharing tasks with each other.

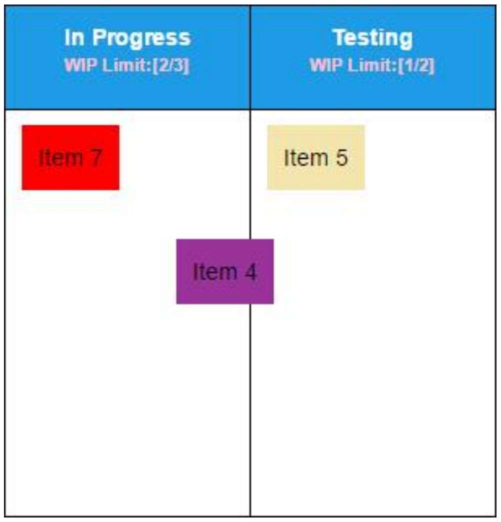




Select Item and Programmer. Then click OK button to share selected item with other team members.

Figure 4.11 Task Management Figure 4.12 Share with window

After choosing "Move To" option, the selected task block is dragged out from the original place. Users can relocate the selected task by moving mouse. After reaching preferred location, the moved task can be dropped to the designated column with a single left-click.



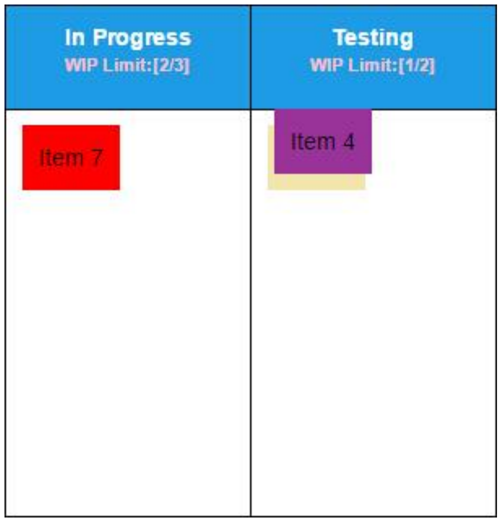
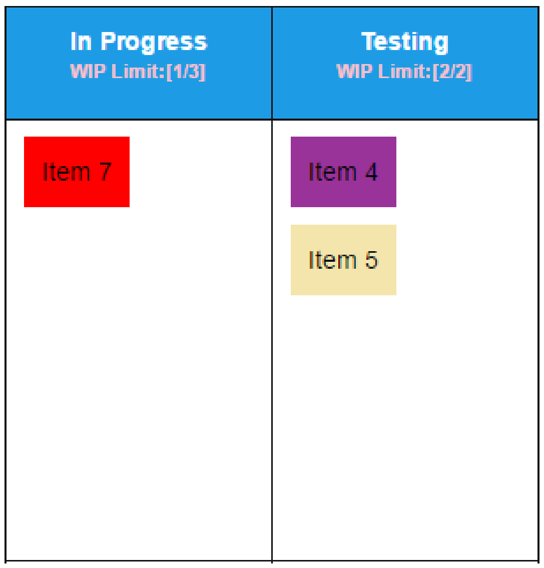


Figure 4.13(a) Move Task Figure 4.13(b) Drop Task

System can automatically arrange Kanban boards' contents, calculate the new WIP load and detect whether overloads happened. If a user accidentally making a column exceeds its WIP limit, an alert will show up and the previous operation will be cancelled.



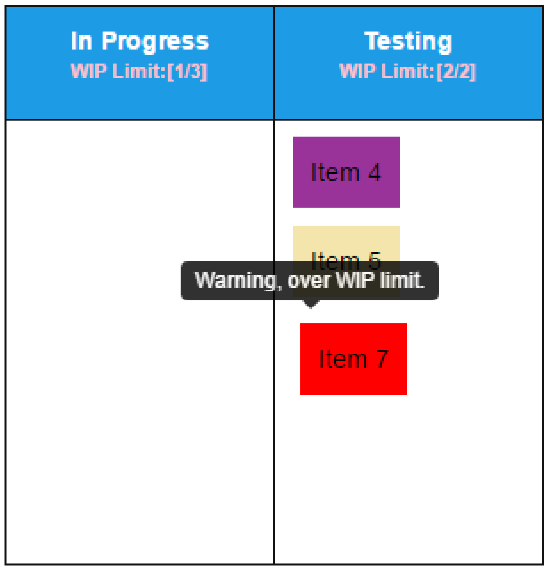


Figure 4.13(c) Automatic Relocation Figure 4.13(d) Warning of Over WIP Limit

As additional requirements may come out from customers, our system must be enabling to let programmers creating new tasks. For easy management, only senior programmers are authorized to use this function. By right-click at the blank area of a working phase, the "New Task" window jumps out. In this window, users can enter the name of the new task and select types of this task.

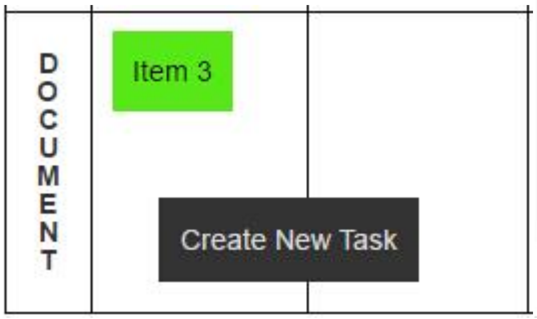


Figure 4.14 Create Task

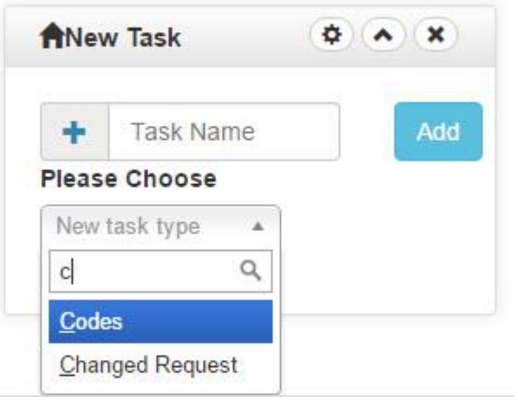
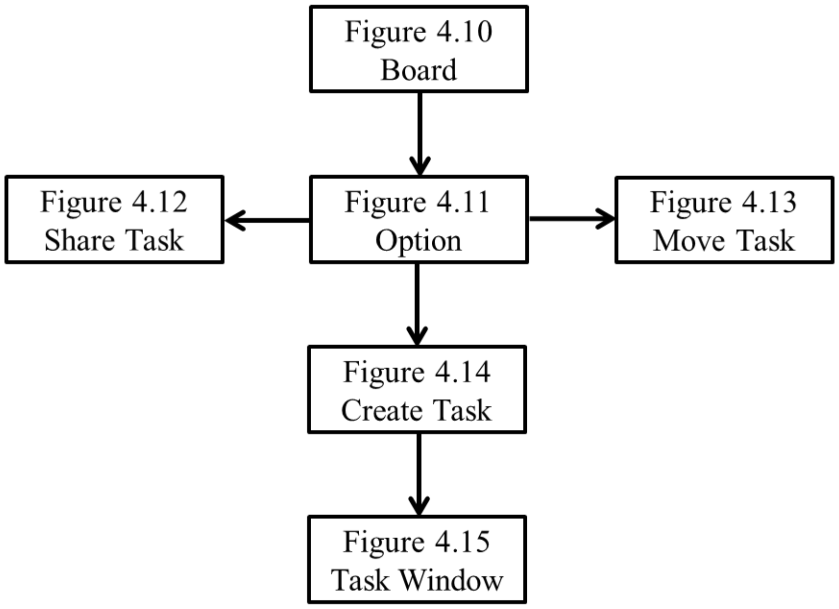


Figure 4.15 New Task Window

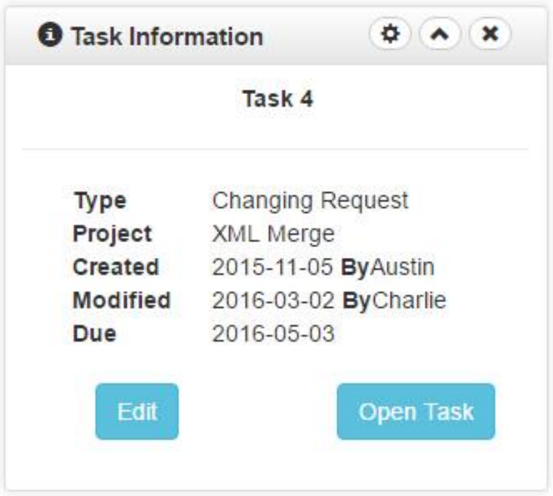
The task management process is illustrated in Workflow 4.4.



Workflow 4.4 Task Management System

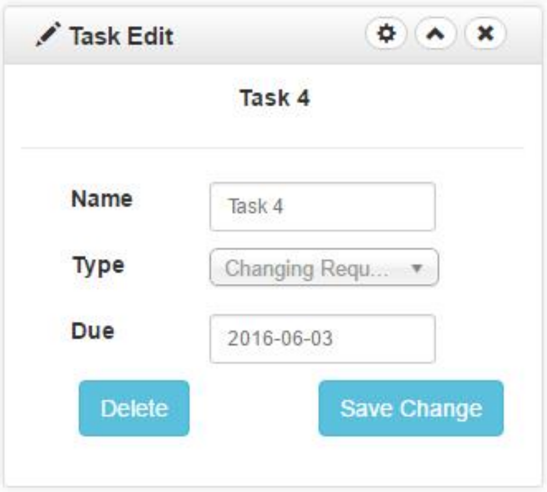
4.2.5 Task Information

Every task can be easily open or edited by left-click on its icon. The “Task Information” window (Figure 4.16) shows up, displaying basic information of the selected task and enabling users edit or open this task. Users can only change the task’s name, type and due date at the “Task Edit” window (Figure 4.17). Further editing to the selected task can be finished by selecting the “Open Task” button.



Task 4

Type	Changing Request
Project	XML Merge
Created	2015-11-05 **By**Austin
Modified	2016-03-02 **By**Charlie
Due	2016-05-03

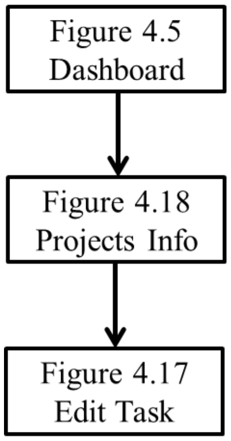


Task 4

Name	Task 4
Type	Changing Requ...
Due	2016-06-03

Figure 4.16 Task Info Figure 4.17 Edit Task

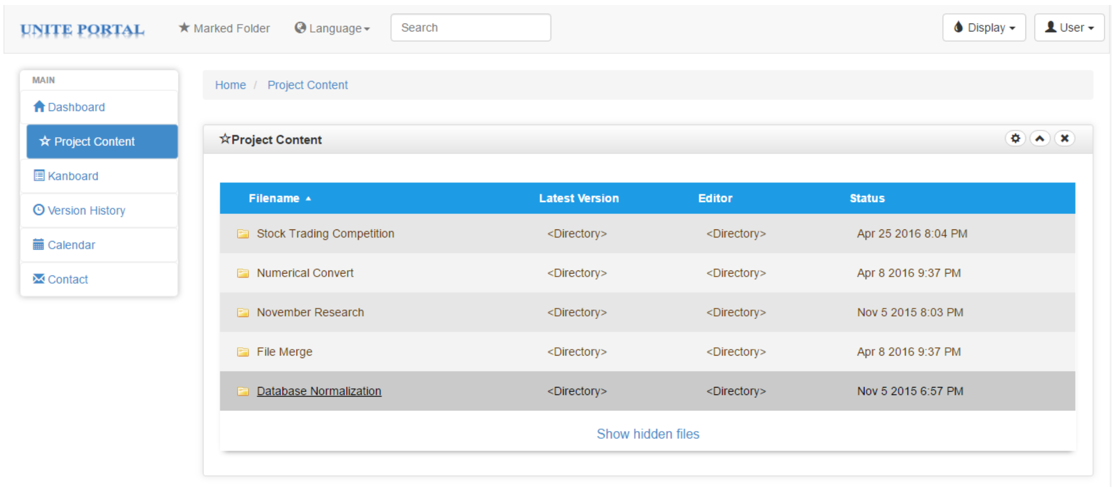
The task editing process is illustrated in Workflow 4.5.



Workflow 4.5 Task Editing

4.2.6 Version Control

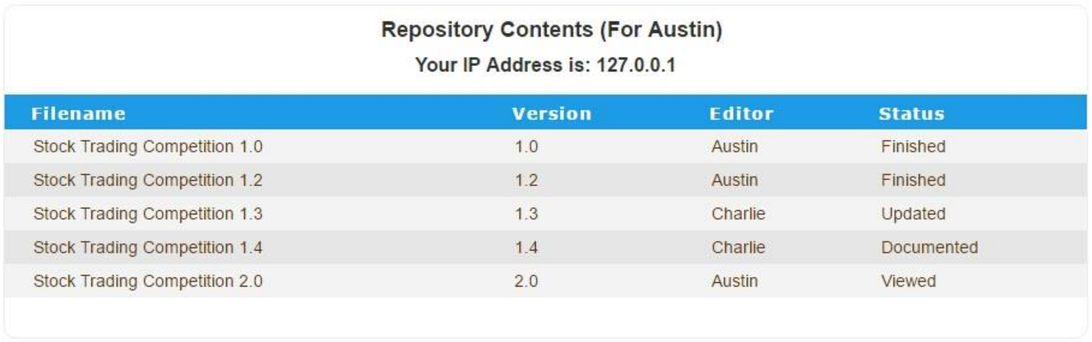
A large project always contains a lot of folders and files. After selecting the "Project Content" button in dashboard, users are able to see a new window in which all authorized projects are listed. In order to store, read, edit and copy files efficiently, our system uses a version control system to help users managing project contents. Figure 4.18 lists the brief information of all projects on progress.



Filename	Latest Version	Editor	Status
Stock Trading Competition	<Directory>	<Directory>	Apr 25 2016 8:04 PM
Numerical Convert	<Directory>	<Directory>	Apr 8 2016 9:37 PM
November Research	<Directory>	<Directory>	Nov 5 2015 8:03 PM
File Merge	<Directory>	<Directory>	Apr 8 2016 9:37 PM
Database Normalization	<Directory>	<Directory>	Nov 5 2015 6:57 PM

Figure 4.18 Current Projects

Figure 4.19 shows all available versions of the "Stock Trading Competition" project. Our system also shows the current working status of each version. In addition, the User IP address is detected by the system. If changes are made to a version, user IP address will be documented. As the contents are currently viewed from the local server, the current IP address is 127.0.0.1.



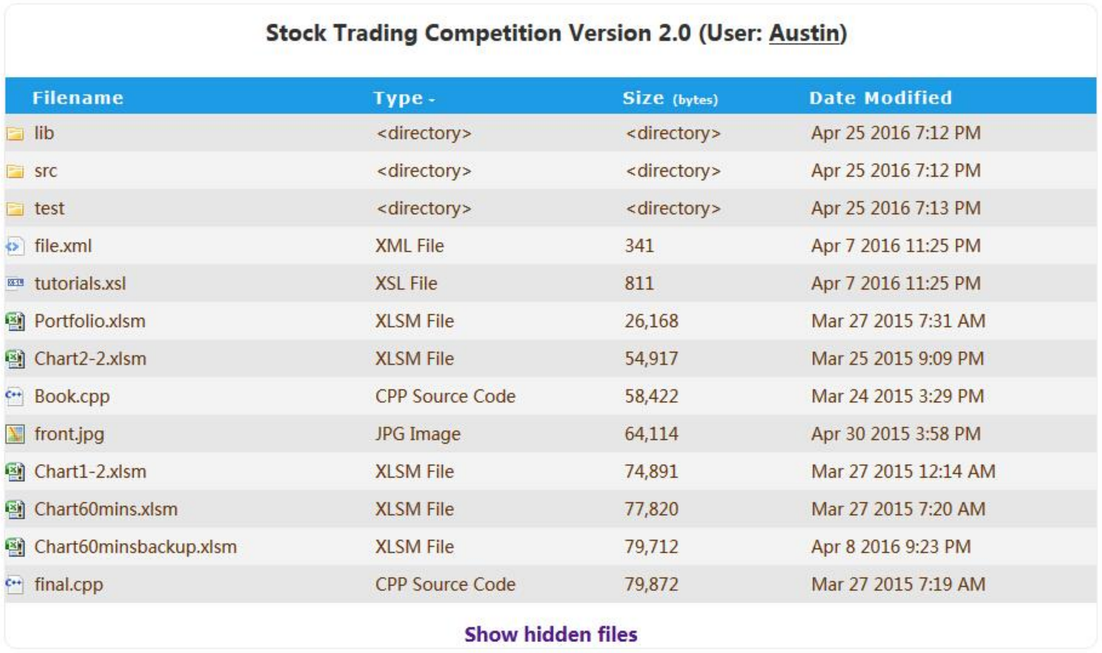
Repository Contents (For Austin)

Your IP Address is: 127.0.0.1

Filename	Version	Editor	Status
Stock Trading Competition 1.0	1.0	Austin	Finished
Stock Trading Competition 1.2	1.2	Austin	Finished
Stock Trading Competition 1.3	1.3	Charlie	Updated
Stock Trading Competition 1.4	1.4	Charlie	Documented
Stock Trading Competition 2.0	2.0	Austin	Viewed

Figure 4.19 Versions of Project

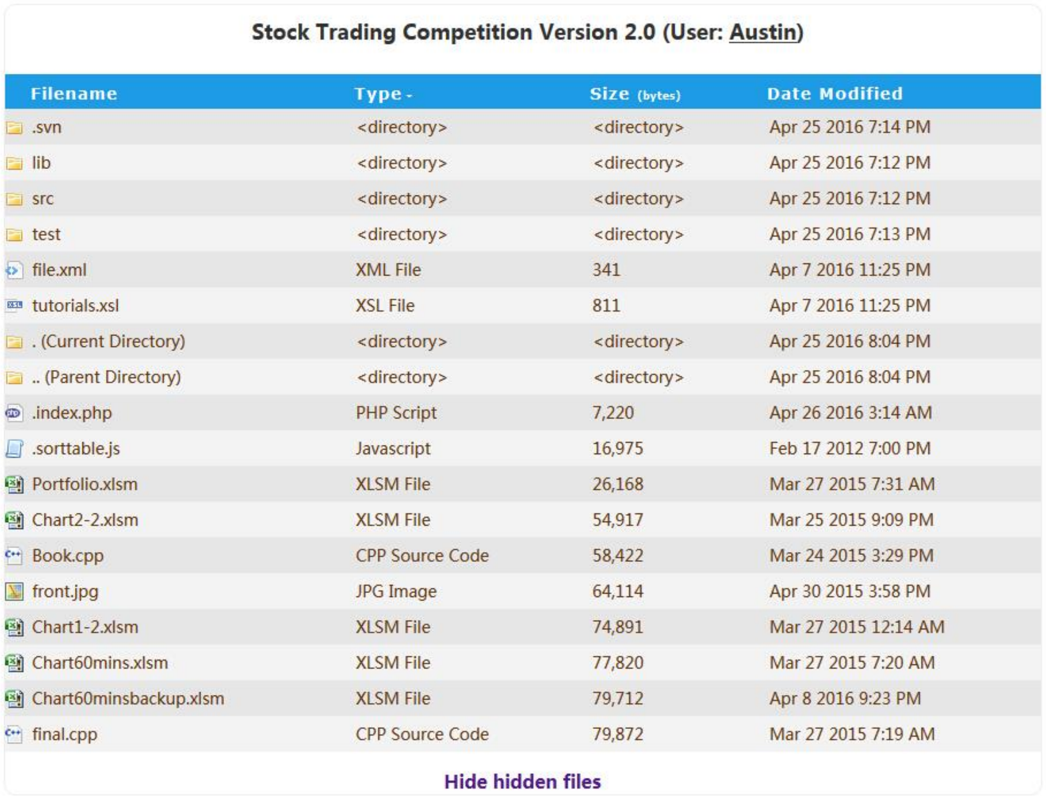
Users can review a specific version by clicking its name. Our system will automatically list all available project contents for users. Compare to Figure 4.20, Figure 4.21 displays the hidden files which support online version control system. The ".index.php" file retrives all contents of the selected version and is the file used to display infomation on web browsers. The ".svn" folder contains the command lines for our version control system.



Stock Trading Competition Version 2.0 (User: Austin)

Filename	Type -	Size (bytes)	Date Modified
lib	<directory>	<directory>	Apr 25 2016 7:12 PM
src	<directory>	<directory>	Apr 25 2016 7:12 PM
test	<directory>	<directory>	Apr 25 2016 7:13 PM
file.xml	XML File	341	Apr 7 2016 11:25 PM
tutorials.xsl	XSL File	811	Apr 7 2016 11:25 PM
Portfolio.xlsm	XLSM File	26,168	Mar 27 2015 7:31 AM
Chart2-2.xlsm	XLSM File	54,917	Mar 25 2015 9:09 PM
Book.cpp	CPP Source Code	58,422	Mar 24 2015 3:29 PM
front.jpg	JPG Image	64,114	Apr 30 2015 3:58 PM
Chart1-2.xlsm	XLSM File	74,891	Mar 27 2015 12:14 AM
Chart60mins.xlsm	XLSM File	77,820	Mar 27 2015 7:20 AM
Chart60minsbackup.xlsm	XLSM File	79,712	Apr 8 2016 9:23 PM
final.cpp	CPP Source Code	79,872	Mar 27 2015 7:19 AM

Figure 4.20 Version Content

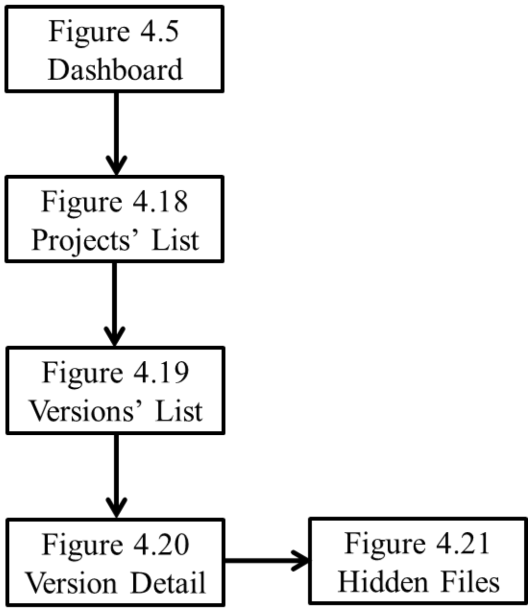


Stock Trading Competition Version 2.0 (User: Austin)

Filename	Type -	Size (bytes)	Date Modified
.svn	<directory>	<directory>	Apr 25 2016 7:14 PM
lib	<directory>	<directory>	Apr 25 2016 7:12 PM
src	<directory>	<directory>	Apr 25 2016 7:12 PM
test	<directory>	<directory>	Apr 25 2016 7:13 PM
file.xml	XML File	341	Apr 7 2016 11:25 PM
tutorials.xsl	XSL File	811	Apr 7 2016 11:25 PM
. (Current Directory)	<directory>	<directory>	Apr 25 2016 8:04 PM
.. (Parent Directory)	<directory>	<directory>	Apr 25 2016 8:04 PM
.index.php	PHP Script	7,220	Apr 26 2016 3:14 AM
.sorttable.js	Javascript	16,975	Feb 17 2012 7:00 PM
Portfolio.xlsm	XLSM File	26,168	Mar 27 2015 7:31 AM
Chart2-2.xlsm	XLSM File	54,917	Mar 25 2015 9:09 PM
Book.cpp	CPP Source Code	58,422	Mar 24 2015 3:29 PM
front.jpg	JPG Image	64,114	Apr 30 2015 3:58 PM
Chart1-2.xlsm	XLSM File	74,891	Mar 27 2015 12:14 AM
Chart60mins.xlsm	XLSM File	77,820	Mar 27 2015 7:20 AM
Chart60minsbackup.xlsm	XLSM File	79,712	Apr 8 2016 9:23 PM
final.cpp	CPP Source Code	79,872	Mar 27 2015 7:19 AM

Figure 4.21 Version Content (Show Hidden Files)

The version control system is illustrated in Workflow 4.6.



Workflow 4.6 Version Control System

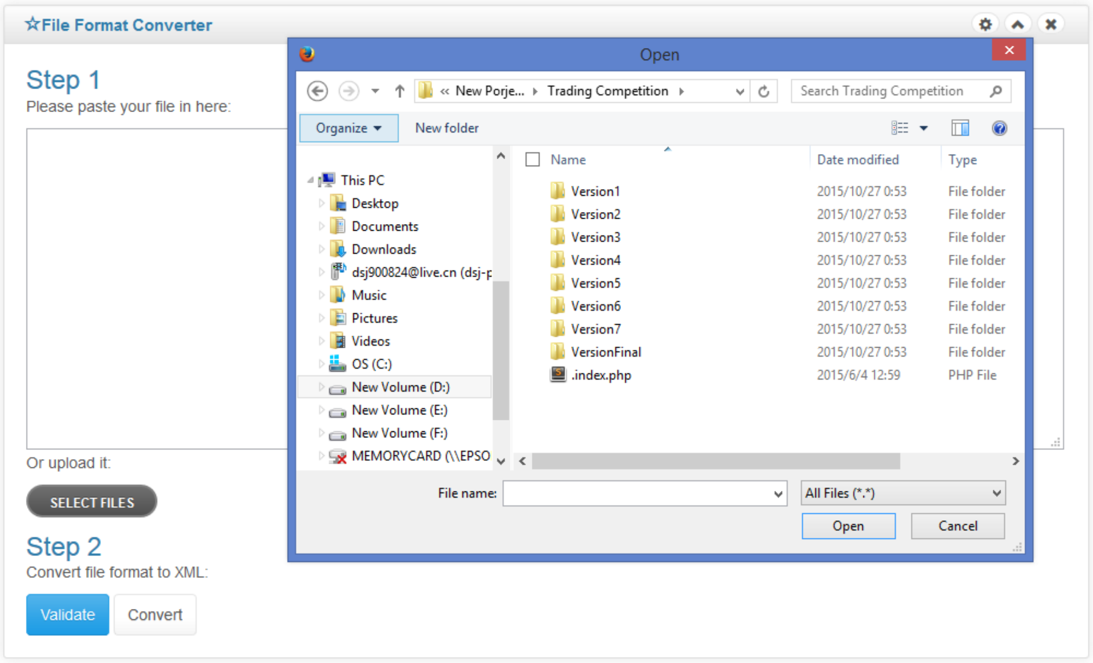
4.2.7 File Upload & Edit

Our system is aim to support collaboration among developers. A key function of this portal is a web-based, code file management system. It allows users easily upload files to the web server and check out from it.

XML is a formal and systematic format for describing structured data. Our system uses XML files to manage source code files and document files. Thus, a XML format files management system is needed to our system.

Firstly, a fundamental function - converting code files to XML format is functioned. This function can automatically detect source code types including C, C++, Java, Python, Ruby, and convert the source code files to XML-format files.

Figure 4.22 shows the process of file uploading.



Name	Date modified	Type
Version1	2015/10/27 0:53	File folder
Version2	2015/10/27 0:53	File folder
Version3	2015/10/27 0:53	File folder
Version4	2015/10/27 0:53	File folder
Version5	2015/10/27 0:53	File folder
Version6	2015/10/27 0:53	File folder
Version7	2015/10/27 0:53	File folder
VersionFinal	2015/10/27 0:53	File folder
.index.php	2015/6/4 12:59	PHP File

Figure 4.22 File Management Upload

By clicking the “Select Files” button, a window pops out allowing user to choose and upload code files to server. The “Validate” button below helps users validating their files. As file contents are displayed in textbox, users are able to edit the original file directly if mistakes were found by online validation.

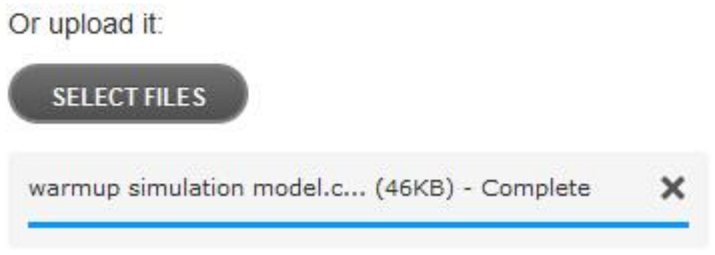
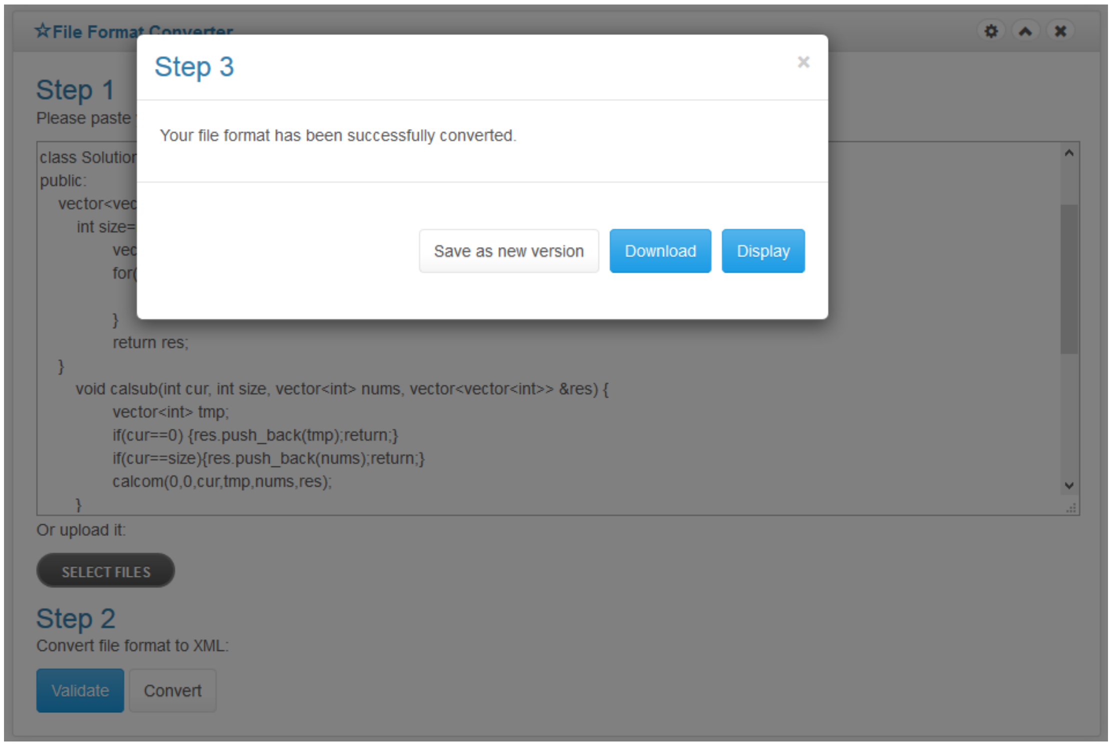


Figure 4.23 File Uploading Animation

After validation, our system helps users converting original source code files to XML format. A pop-out window will show up after successful conversion, which helps users uploading the file to database, downloading to local computer or checking on the web browser.



Your file format has been successfully converted.

```
class Solutior
public:
    vector<vec
        int size=
            vec
            for(

            }
            return res;
    }
    void calsub(int cur, int size, vector<int> nums, vector<vector<int>> &res) {
            vector<int> tmp;
            if(cur==0) {res.push_back(tmp);return;}
            if(cur==size){res.push_back(nums);return;}
            calcom(0,0,cur,tmp,nums,res);
    }
```

Figure 4.24 Conversion Confirm Window

The core component of our system is the XML file merge function. As source code files and corresponding document files are both converted in XML format, these files can be easily edited, merged, displayed and converted back to the original format. Figure 4.25 shows the view of online merge function for source code files and document files.

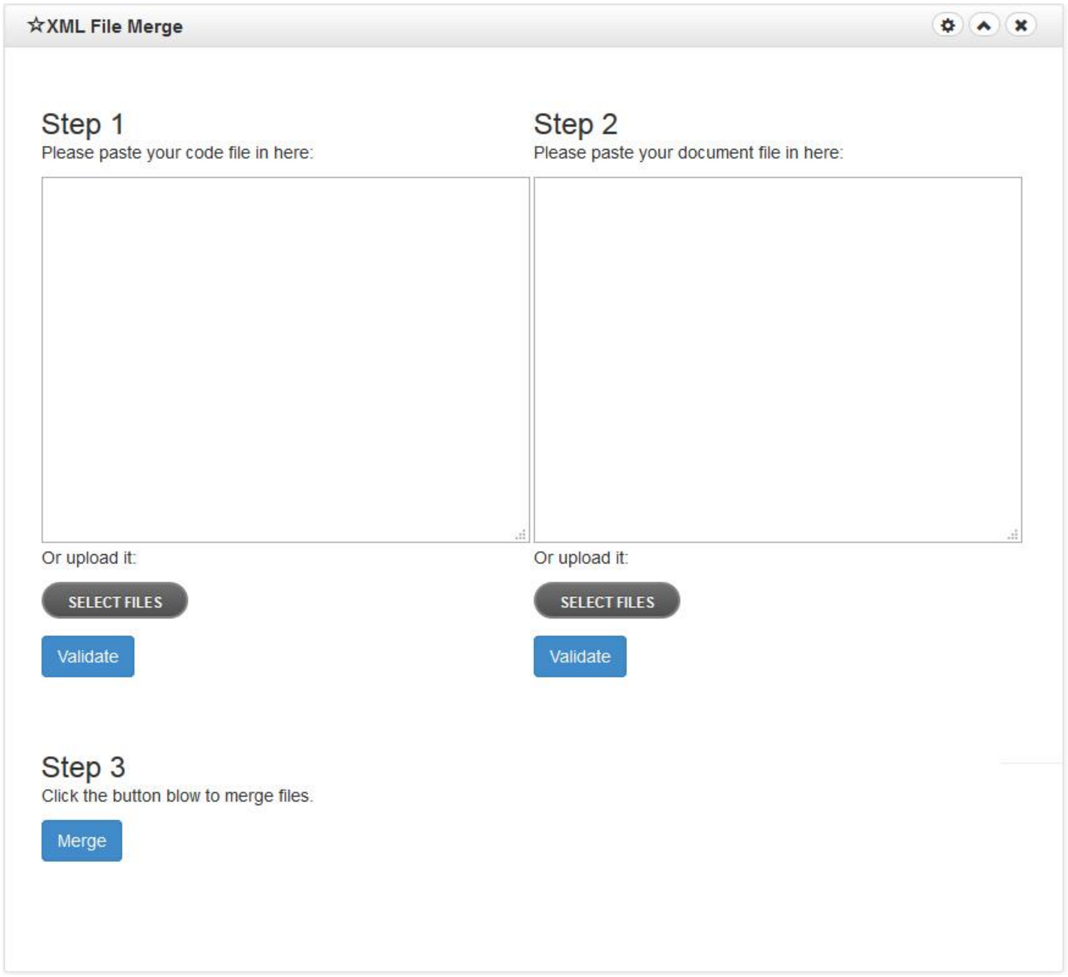
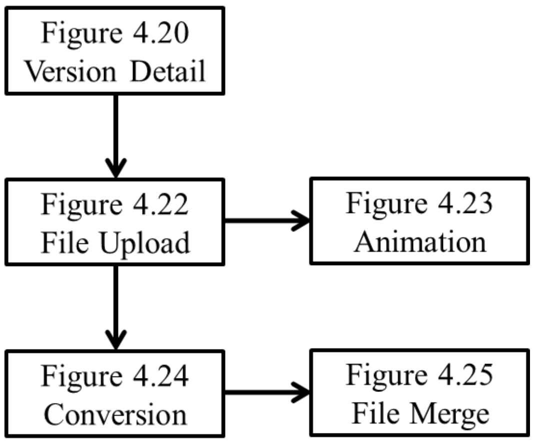

☆XML File Merge
Step 1
Please paste your code file in here:
Or upload it:
SELECT FILES
Validate
Step 2
Please paste your document file in here:
Or upload it:
SELECT FILES
Validate
Step 3
Click the button blow to merge files.
Merge

Figure 4.25 File Merge

The file management process is illustrated in Workflow 4.7.

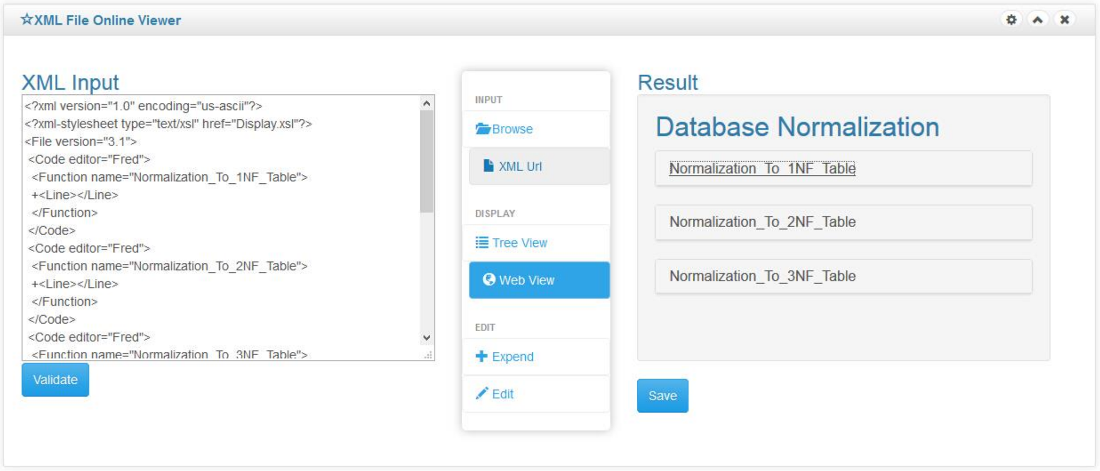


Workflow 4.7 File Management System

4.2.8 XML Online Viewer

Our system allows users read and edit source code files dynamically online.

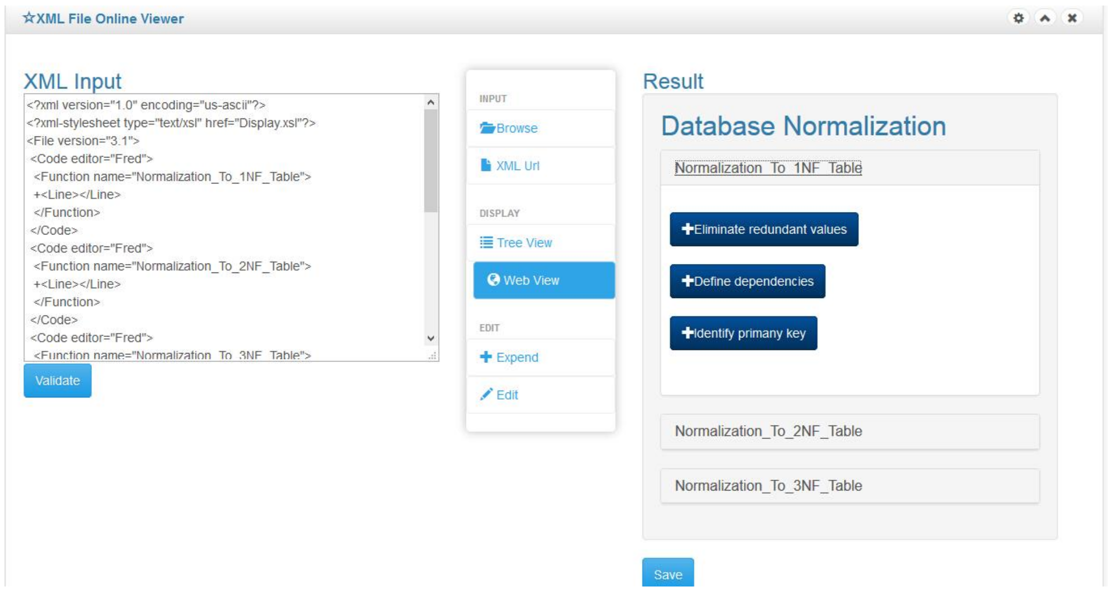
This online, extendible source code viewer supports online collaboration among programmers. This viewer helps apprentices merging source code files and documents files together. Figure 4.26 is a sample for our XML viewer.



```
<?xml version="1.0" encoding="us-ascii"?>
<?xml-stylesheet type="text/xsl" href="Display.xsl"?>
<File version="3.1">
 <Code editor="Fred">
  <Function name="Normalization_To_1NF_Table">
  +<Line></Line>
  </Function>
 </Code>
 <Code editor="Fred">
  <Function name="Normalization_To_2NF_Table">
  +<Line></Line>
  </Function>
 </Code>
 <Code editor="Fred">
  <Function name="Normalization_To_3NF_Table">
```

Figure 4.26 XML Online Viewer

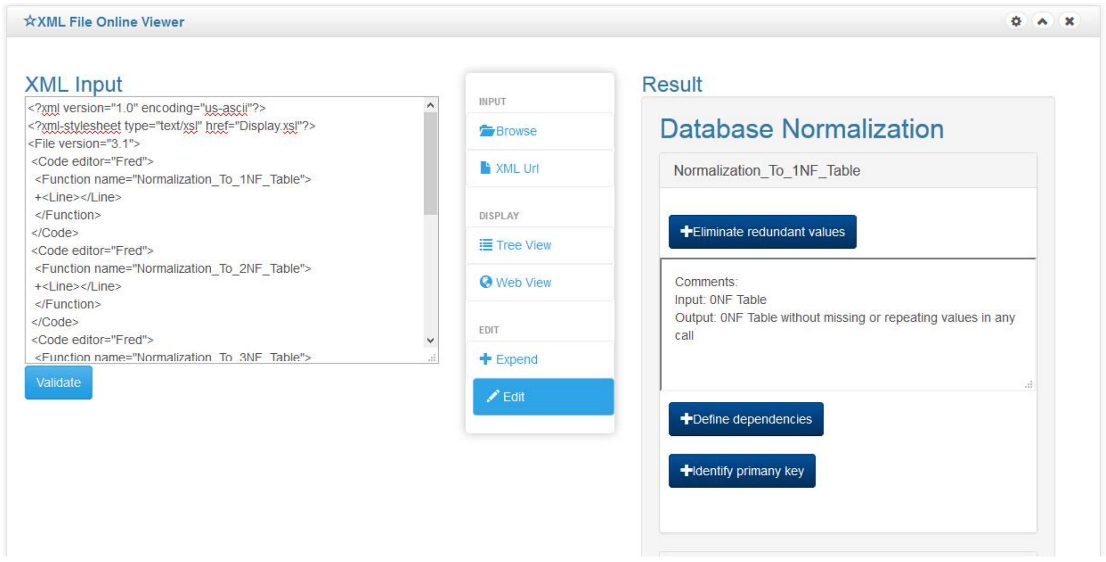
. In the “Database Normalization” project, all three functions are displayed in extendible blocks. In addition, the function “Normalization_To_1NF_Table” has three sub functions “Eliminate_redundant_values”, “Define_dependencies” and “Identify primary_key”.



```
<?xml version="1.0" encoding="us-ascii"?>
<?xml-stylesheet type="text/xsl" href="Display.xsl"?>
<File version="3.1">
 <Code editor="Fred">
  <Function name="Normalization_To_1NF_Table">
  +<Line></Line>
  </Function>
 </Code>
 <Code editor="Fred">
  <Function name="Normalization_To_2NF_Table">
  +<Line></Line>
  </Function>
 </Code>
 <Code editor="Fred">
  <Function name="Normalization_To_3NF_Table">
```

Figure 4.27 Web View – Function View

By using the XML online viewer, users are able to select the “Edit” button to add comments to XML files online through Textbox.

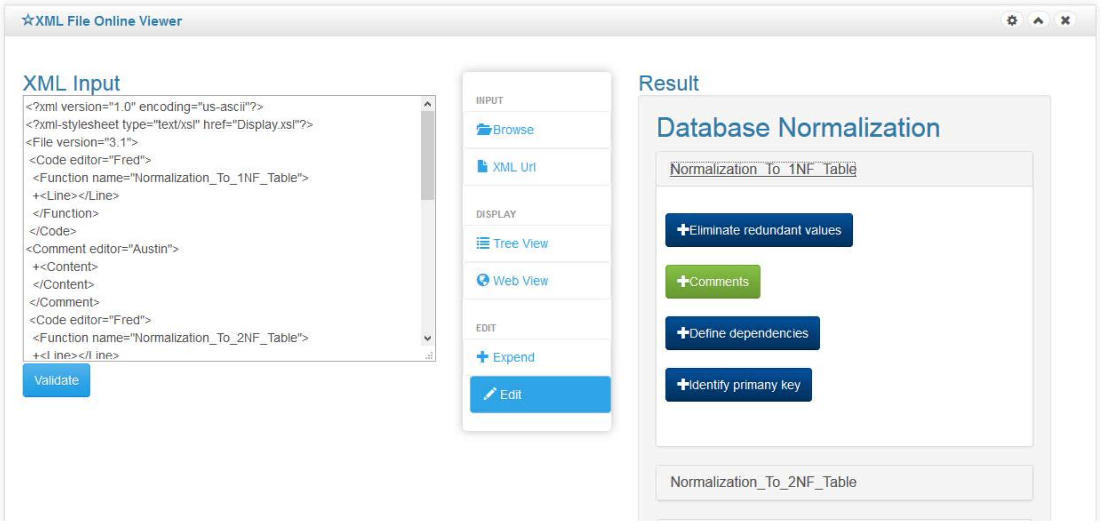


```
<?xml version="1.0" encoding="us-ascii"?>
<?xml-stylesheet type="text/xsl" href="Display.xsl"?>
<File version="3.1">
 <Code editor="Fred">
  <Function name="Normalization_To_1NF_Table">
  +<Line></Line>
  </Function>
 </Code>
 <Code editor="Fred">
  <Function name="Normalization_To_2NF_Table">
  +<Line></Line>
  </Function>
 </Code>
 <Code editor="Fred">
  <Function name="Normalization_To_3NF_Table">
```

Comments:
Input: 0NF Table
Output: 0NF Table without missing or repeating values in any call

Figure 4.28 XML File Edit

Unlike source codes, all comments are displayed in another color, in order to make the whole file more readable.



```
<?xml version="1.0" encoding="us-ascii"?>
<?xml-stylesheet type="text/xsl" href="Display.xsl"?>
<File version="3.1">
 <Code editor="Fred">
  <Function name="Normalization_To_1NF_Table">
  +<Line></Line>
  </Function>
 </Code>
<Comment editor="Austin">
 +<Content>
 </Content>
</Comment>
<Code editor="Fred">
 <Function name="Normalization_To_2NF_Table">
 +<Line></Line>
```

Database Normalization

Normalization_To_1NF_Table

+Eliminate redundant values

+Comments

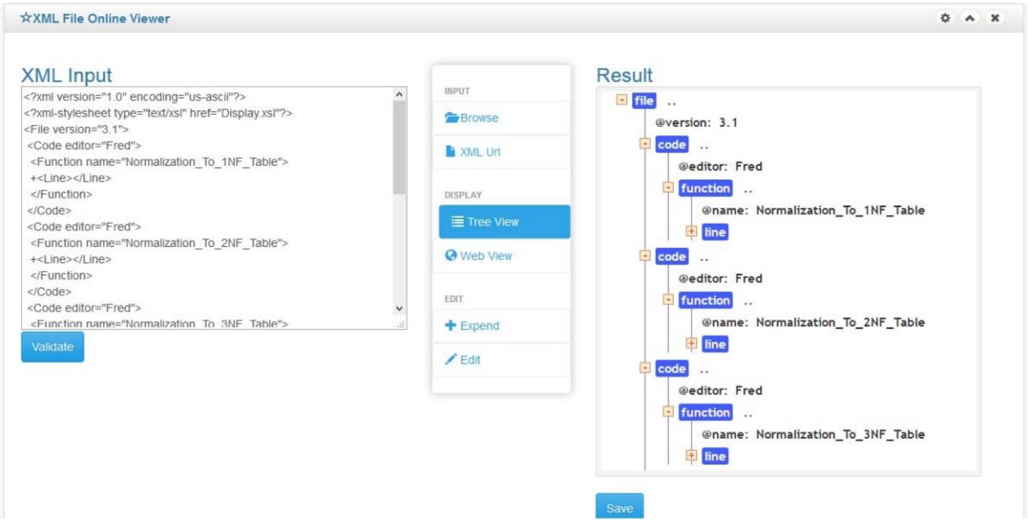
+Define dependencies

+Identify primary key

Normalization_To_2NF_Table

Figure 4.29 XML File with Comments Merged

Our system has two different ways displaying XML files. In addition to the web view, the other one is the tree view. As mentioned in Chapter 3, our system uses tags' attributes to store editors' information. The tree view can display the attributes' contents for certain tags of an XML file. In Figure 4.30, the current version number and all editors' information are shown in the tree view window.

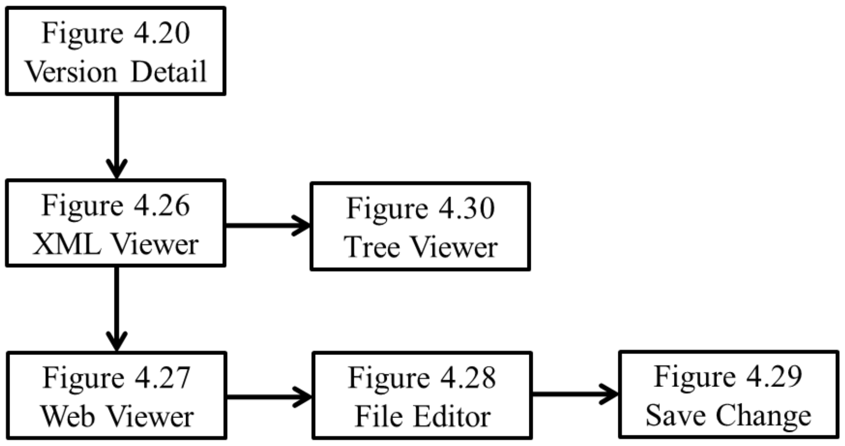


```
<?xml version="1.0" encoding="us-ascii"?>
<?xml-stylesheet type="text/xsl" href="Display.xsl"?>
<File version="3.1">
 <Code editor="Fred">
  <Function name="Normalization_To_1NF_Table">
  +<Line></Line>
  </Function>
 </Code>
 <Code editor="Fred">
  <Function name="Normalization_To_2NF_Table">
  +<Line></Line>
  </Function>
 </Code>
 <Code editor="Fred">
  <Function name="Normalization_To_3NF_Table">
```

```
⊟ file ..
   @version: 3.1
  ⊟ code ..
     @editor: Fred
    ⊟ function ..
       @name: Normalization_To_1NF_Table
      ⊞ line
  ⊟ code ..
     @editor: Fred
    ⊟ function ..
       @name: Normalization_To_2NF_Table
      ⊞ line
  ⊟ code ..
     @editor: Fred
    ⊟ function ..
       @name: Normalization_To_3NF_Table
      ⊞ line
```

Figure 4.30 Tree View

The XML files management process is illustrated in Workflow 4.8.



Workflow 4.8 XML Files Management System

Chapter 5
Conclusions & Future Work

5.1 Conclusions

Agile programming has been widely adopted for software development. It consists of several software development methodologies based on iterative and incremental developing. Although the efficiency of agile programming has already been proven, the lack of documentation and knowledge sharing requires improvements. Unite Portal is designed not only support communication and collaboration between developers, but also manage knowledge in an explicit format.

Unite Portal is intended to provide an alternative knowledge management method to the agile developing teams. Developers are divided into two groups. Senior developers, who are referred to as the Masters, focus on agile software development mainly and leave all documentation jobs to junior developers, who are referred to as the Apprentices. With a specified group of developers assigned to create, manage, and share various types of documents, the whole development teams can achieve high quality project with proper documentations. In addition, knowledge management can help make development progress well-organized.

Unite Portal supports collaboration among development team at various levels. In a software development lifecycle, junior software developers are mainly responsible for creating, managing and sharing various types of documents for the agile software development process. Unite Portal works as a platform for junior programmers working with and learning from senior programmers through reading code and writing documentation.

5.2 Future Work

It would be simple and necessary to add multiple language user interface support (such as English/French) to the current Unite Portal system. We will use XML for the implementation of multiple language user interface support. In future, we plan to conduct real experiments to study the effectiveness of Unite Portal on knowledge management for agile software development. Moreover, we will conduct a detailed cost-benefit analysis on the proposed Master-Apprentice model for knowledge management of Agile Programming. Firstly, we need to examine the efficiency of our knowledge analytics portal for experienced programmers who are familiar with Agile Programming. Within a software development organization, we can randomly divide programmers to two groups. By letting one group use our system and the other group work in the original way, we can have our control groups for experienced programmers. Secondly, as Unite Portal may also help undergraduate students to learn programming skills, the value of such a knowledge analytics portal will be investigated in the university settings. This can be another control group for testing the efficiency of our system for unexperienced user.

References

[1] Abiteboul, S., Buneman, P., & Suciu, D. (2000). *Data on the Web: from relations to semistructured data and XML*. Morgan Kaufmann.

[2] Ahmad, M. O., Markkula, J., & Oivo, M. (2013, September). Kanban in software development: A systematic literature review. In *2013 39th Euromicro Conference on Software Engineering and Advanced Applications*(pp. 9-16). IEEE.

[3] Alavi, M., & Leidner, D. E. (1999). Knowledge management systems: issues, challenges, and benefits. *Communications of the AIS*, *1*(2es), 1.

[4] Alavi, M., & Leidner, D. E. (2005). Review: knowledge management and knowledge management systems: conceptual foundations and research issues. *Knowledge Management: Critical Perspectives on Business and Management*, 163-202.

[5] Ambler, S. W. (2008). Agile adoption rate survey results. *available on-line at http://www.ambysoft.com/surveys/agileFebruary2008.html*.

[6] Ambler, S. W. (2011). Agile/lean documentation: Strategies for agile software development.

[7] Anderson, D. J. (2010). *Kanban: successful evolutionary change for your technology business*. Blue Hole Press.

[8] Beck, K. (2000). *Extreme programming explained: embrace change*. addison-wesley professional.

[9] Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., ... & Kern, J. (2001). Manifesto for agile software development.

[10] Bossi, P. (2003, January). eXtreme Programming applied: a case in the private banking domain. In *Proceedings of OOP*.

[11] Cockburn, A., & Williams, L. (2000). The costs and benefits of pair programming. *Extreme programming examined*, 223-247.

[12] Collins, C. J., & Clark, K. D. (2003). Strategic human resource practices, top management team social networks, and firm performance: The role of human resource practices in creating organizational competitive advantage. *Academy of management Journal*, *46*(6), 740-751.

[13] Collins-Sussman, B., Fitzpatrick, B., & Pilato, M. (2004). *Version control with*

subversion. " O'Reilly Media, Inc.".

[14] DeLone, W. H., & McLean, E. R. (1992). Information systems success: The quest for the dependent variable. *Information systems research*, *3*(1), 60-95.

[15] Denny, N., Crk, I., Nadella, R. S., & Gupta, A. (2009). Agile software processes for the 24-hour knowledge factory environment. *Available at SSRN 1017184*.

[16] Desouza, K. C., Awazu, Y., & Baloh, P. (2006). Managing knowledge in global software development efforts: Issues and practices. *IEEE software*, *23*(5), 30.

[17] Dybå, T., Arisholm, E., Sjøberg, D. I., Hannay, J. E., & Shull, F. (2007). Are two heads better than one? On the effectiveness of pair programming. *IEEE software, 24*(6), 12-15.

[18] Fliaster, A., & Schloderer, F. (2010). Dyadic ties among employees: Empirical analysis of creative performance and efficiency. *Human Relations*, *63*(10), 1513-1540.

[19] Friedman-Hill, E. J. (2001, January). Software verification and functional testing with XML documentation. In *System Sciences, 2001. Proceedings of the 34th Annual Hawaii International Conference on* (pp. 8-pp). IEEE.

[20] Fruhling, A., McDonald, P., & Dunbar, C. (2008, January). A case study: introducing extreme programming in a US government system development project. In *Hawaii International Conference on System Sciences, Proceedings of the 41st Annual* (pp. 464-464). IEEE.

[21] Ge, X., Paige, R. F., & McDermid, J. A. (2010, August). An iterative approach for development of safety-critical software and safety arguments. In *Agile Conference (AGILE), 2010* (pp. 35-43). IEEE.

[22] He, W., & Wei, K. K. (2009). What drives continued knowledge sharing? An investigation of knowledge-contribution and-seeking beliefs. *Decision Support Systems*, *46*(4), 826-838.

[23] Hurtado, J. (2013). Open Kanban-An Open Source. *Ultra Light, Agile & Lean Method.[Online] Available from: http://agilelion. com/[Accessed: 7 June 2015]*.

[24] Jackson, S. E., DeNisi, A., & Hitt, M. A. (Eds.). (2003). *Managing knowledge for sustained competitive advantage: Designing strategies for effective human resource*

management (Vol. 21). John Wiley & Sons.

[25] Jensen, R. (2003). A pair programming experience. *CrossTalk*, *16*(3), 22-24.

[26] Kankanhalli, A., & Tan, B. C. (2004, January). A review of metrics for knowledge management systems and knowledge management initiatives. In *System Sciences, 2004. Proceedings of the 37th Annual Hawaii International Conference on* (pp. 8-pp). IEEE.

[27] Kniberg, Henrik, and Mattias Skarin. *Kanban and Scrum-making the most of both*. Lulu. com, 2010.

[28] Ladas, C. (2008). Scrumban. *Lean Software Engineering-Essays on the Continuous Delivery of High Quality Information Systems*.

[29] Layton, M. C. (2015). *Scrum for Dummies*. John Wiley & Sons.

[30] Lepak, D., & Snell, S. A. (2007). Employment Subsystems and the 'HR Architecture'. *Oxford Handbook of Human Resource Management, The*, 210.

[31] Levy, M., & Hazzan, O. (2009, May). Knowledge management in practice: The case of agile software development. In *Cooperative and Human Aspects on Software Engineering, 2009. CHASE'09. ICSE Workshop on* (pp. 60-65). IEEE.

[32] Liker, J. K. (2005). *The toyota way*. Esensi.

[33] Lui, K. M., & Chan, K. C. (2008). Software process fusion by combining pair and solo programming. *IET software*, *2*(4), 379-390.

[34] Mahnic, V. (2014). Improving Software Development through Combination of Scrum and Kanban. *Recent Advances in Computer Engineering, Communications and Information Technology, Espanha*.

[35] Mannaro, K. (2008). Adopting agile methodologies in distributed software development.

[36] Monden, Y. (1983). *Toyota production system: practical approach to production management*. Engineering & Management Press.

[37] Muller, M. M., & Tichy, W. F. (2001, May). Case study: extreme programming in a university environment. In *Software Engineering, 2001. ICSE 2001. Proceedings of the 23rd International Conference on* (pp. 537-544). IEEE.

[38] Nonaka, I., & Takeuchi, H. (1995). The knowledge creation company: how Japanese

companies create the dynamics of innovation. *Oxford University Press. New York, USA*, 304.

[39] Otte, S. (2009). Version Control Systems. *Computer Systems and Telematics*.

[40] Ovais, M., Markkula, J., Oivo, M., Kuvaja, P., Ahmad, J., Markkula, M., & Oivo, P. (2014, October). Usage of Kanban in Software Companies An empirical study on motivation, benefits and challenges. In *9th International Conference on Software Engineering Advances*.

[41] Palmieri, D. W. (2002). Knowledge management through pair programming.

[42] Reddy, A. (2015). *The Scrumban [r] evolution: Getting the Most Out of Agile, Scrum, and Lean Kanban*. Addison-Wesley Professional.

[43] Salis, S., & Williams, A. M. (2010). Knowledge Sharing through Face-to-Face Communication and Labour Productivity: Evidence from British Workplaces. *British Journal of Industrial Relations*, *48*(2), 436-459.

[44] Salminen, A., & Tompa, F. (2012). *Communicating with XML*. Springer Science & Business Media.

[45] Schneider, K. (2009). *Experience and knowledge management in software engineering* (Vol. 235). Berlin: Springer.

[46] Schneider, K., Stapel, K., & Knauss, E. (2008, September). Beyond documents: visualizing informal communication. In *Requirements Engineering Visualization, 2008. REV'08.* (pp. 31-40). IEEE.

[47] Schwaber, K. (1997). Scrum development process. In *Business Object Design and Implementation* (pp. 117-134). Springer London.

[48] Schwaber, K., Sutherland, J., & Beedle, M. (2013). The definitive guide to scrum: the rules of the game. *Recuperado de: http://www. scrumguides. org/docs/scrumguide/v1/scrum-guide-us. pdf.*

[49] Scotland, K. (2010). Aspects of kanban. *Method and Tools-Summer*, *2010*.

[50] Silva, L., Santana, C., Rocha, F., Paschoalino, M., Falconieri, G., Ribeiro, L., ... & Gusmão, C. (2008, June). Applying XP to an Agile–Inexperienced Software Development Team. In *International Conference on Agile Processes and Extreme Programming in Software Engineering* (pp. 114-126). Springer Berlin Heidelberg.

[51] Stapel, K., & Schneider, K. (2012). Managing knowledge on communication and information flow in global software projects. *Expert Systems*.

[52] Stapleton, J. (1997). *DSDM, dynamic systems development method: the method in practice*. Cambridge University Press.

[53] Turner, R., Ingold, D., Lane, J. A., Madachy, R., & Anderson, D. (2012). Effectiveness of kanban approaches in systems engineering within rapid response environments. *Procedia Computer Science*, *8*, 309-314.

[54] Wadler, P. (1999, December). A formal semantics of patterns in XSLT. In *Markup technologies* (Vol. 99).

[55] Willem, A., & Scarbrough, H. (2006). Social capital and political bias in knowledge sharing: An exploratory study. *Human relations*, *59*(10), 1343-1370.

[56] Williams, L., & Kessler, R. (2002). *Pair programming illuminated*. Addison-Wesley Longman Publishing Co., Inc..

[57] Williams, L., Kessler, R. R., Cunningham, W., & Jeffries, R. (2000). Strengthening the case for pair programming. *IEEE software*, *17*(4), 19.

[58] Wright, K. B., & Webb, L. M. (Eds.). (2011). *Computer-mediated communication in personal relationships*. Peter Lang.

[59] Yen, D. C., Huang, S. M., & Ku, C. Y. (2002). The impact and implementation of XML on business-to-business commerce. *Computer Standards & Interfaces*, *24*(4), 347-362.

[60] Zholudev, V., & Kohlhase, M. (2009, July). TNTBase: a versioned storage for XML. In *Proceedings of Balisage: The Markup Conference* (Vol. 3, p. 64).