Agile Development of Web Application by Supporting Process Execution and Extended UML Model*

Wookjin Lee[1], Sanghyun Park[2], Keeyoull Lee[2], Chunwoo Lee[2], Byungjeong Lee[3] **,
Woosung Jung[2], Taeksu Kim[2], Heechern Kim[4], and Chisu Wu[2]
[1]*Software Center, LG Electronics*
duri96@selab.snu.ac.kr
[2]*School of Computer Science and Engineering, Seoul National University*
{zez4shy, kylee, oniguni, wsjung, dolicoli, wuchisu}@selab.snu.ac.kr
[3]*School of Computer Science, University of Seoul*
bjlee@venus.uos.ac.kr
[4]*Department of Computer Science, Korea National Open University, Korea*
hckim@knou.ac.kr

Abstract

Agile and systematic development is important in Web application methodology because clients expect their frequent requirement changes to be implemented rapidly and regularly. However, previous studies may not be suitable for such development.

In this paper we propose a methodology for agile and somewhat systematic development of Web application. The methodology includes models, tools, and an agile process. The models extended from UML focus on the behavior of Web application. The agile process aims to have quick-to-market property and adaptability to requirement changes. Tools including Process Manager and Web Modeler support process execution and modeling activity to guide developers in applying the process. An example of university assets management system and a comparison between our methodology and other processes are given to validate our methodology.

Keywords: Agile Methodology, Process Execution, Web Application, extended UML model

1. Introduction

Web applications are widely used in e-commerce and online information systems. However, requirement changes are demanded frequently due to fast business environment shifts [1]. Furthermore, the development of large Web applications tends to be complex and not manageable because of the huge number of pages, the diverse technology, and dynamic generation of pages.

Several Web application development methodologies have been suggested. Web application design approaches such as Object-Oriented Hypermedia Design model (OOHDM) [2] and UML-based Web Engineering (UWE) [3] emphasize systematic design method, but require a number of activities and artifacts. On the other hand, agile processes like Extreme Programming (XP) [3] and Agile Web Engineering (AWE) process [5] are suitable for fast development, but are not customized for Web application development and does not provide proper tools and design methodology. Therefore agile and systematic methodology is needed for Web application development.

In this paper we present a methodology for agile and somewhat systematic Web application development. The Web application development methodology includes models, tools, and an agile process for systematic and fast development. The models extended from UML focus on the behavior of Web application. The agile process aims to have quick-to-market property and adaptability to requirement changes. Tools support process execution and modeling activity to guide developers in applying the process.

This paper is organized as follows. Section 2 explains related work, and Section 3 presents extended models and a development process for Web application. Section 4 shows two tools: Process Manager and Web Modeler. We describe a case study in Section 5. Finally, our conclusions are drawn in Section 6.

* This work was supported by grant No. R01-2002-000-00135-0 from the Basic Research Program of the Korea Science & Engineering Foundation.
** Corresponding author

2. Related Work

2.1. Web application design method

Most of Web design methods have conceptual modeling, navigation modeling, and presentation modeling in common. OOHDM [2] includes conceptual modeling, presentation modeling, abstract interface design, and implementation activities. In WebML [6] Web applications are developed through requirements gathering, data design, hypertext design, presentation design, user/group design and customization design activities. UWE [3] models Web applications using extended UML.

However, they are not likely to support quick-to-market property because a number of modeling activities and artifacts make the whole methodology heavy. Furthermore the methodology is not agile to allow frequent requirement changes because they concentrate on modeling activities. Selmi et al. [7] also point out some limitations such as

- Inability to model business processes
- Inability to support various abstraction levels,
- No use of standard notation,
- Inability to model interaction aspects,
- Inability to support dynamic generation content.

2.2. Agile Web Development Process

AWE(Agile Web Engineering) [5] is a lightweight process that focuses on short development lifecycle time, small multidisciplinary development teams and delivery of bespoke solutions. Moreover, AWE encourages requirement analysis, testing and evaluation. But AWE only focuses on agile process. Thus developers need proper tools and design methodology to apply AWE to Web development projects in practice.

Considering frequent requirement changes in Web application development, agile processes like XP [4] and SCRUM [8] are suitable to Web engineering domain. XP emphasizes values of communication and feedback. One of the SCRUM principles is adaptability. Web development process should be adaptable to rapidly changing requirements. Although theses principles and values are consistent with those of Web engineering, the processes are not fully customized to Web engineering domain. For example model design and review are generally required due to complexity of Web application [9].

We can not apply these agile processes to Web application development directly. Web engineering development environment including practical processes, concrete models and supporting tools has not been proposed.

3. Agile Development Methodology

Our methodology provides an agile process, models, and tools. The process is agile so as to accept requirement changes rapidly, covering an overall development process. We focus on a behavioral aspect of models such as navigation and communication. Tools including a storyboarding tool, Web Modeler, an HTML editor, and Integrated Development Environment (IDE) are required for agile and rapid development (Figure 1). We implement Web Modeler for modeling a behavioral view and Process Manager for guiding developers in applying the process.

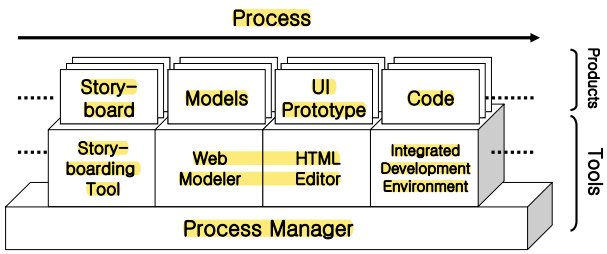


Figure 1. Overview of our methodology

3.1 Extended UML Model

Although UML is a generic modeling language, it is not sufficient for modeling the navigation of Web applications. We define a Web application model which consists of navigation model (NM), component communication model (CCM), conceptual model and architecture. The NM and the CCM are extended from StateMachines package and Interactions package of UML 2.0 [10], respectively. The conceptual model and the architecture are drawn using class diagram and component diagram without extension, respectively. In this work we focus on the NM and the CCM.

Navigation Model In this model, navigations are shown focusing on changes of a view which represents a state of display. A physical page can be expressed as many views according to its own status change. Basically a client page can have a *PageView*, whereas a server page which does not exist physically as a file can have a *VirtualPageView*.

Navigation can be classified by whether it contains data transmission or not. The former is defined as *Submit*, and the latter, *Navigation*. *Navigation* is similar to a simple link between pages. In the case of *Submit*, data is transferred from a presentation layer to a business logic layer, and corresponding components respond. Then the response is sent back to a client page or a virtually generated page.

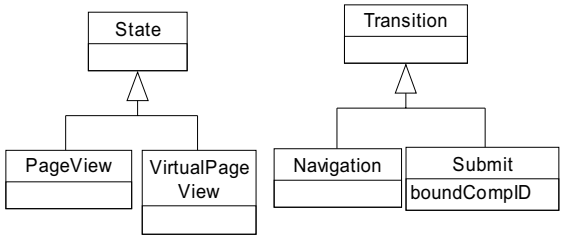


Figure 2. Structure of Navigation package

PageView, *VirtualPageView*, *Navigation* and *Submit* are metamodel elements of the NM. Figure 2 shows an abstract extension and elements of the Navigation package. Attribute *boundCompId* of *Submit* is an identifier of a corresponding component. Table 1 describes elements in NM.

Table 1. Elements in Navigation model

Node Type	Icon	Path Type	Icon
PageView	Ⓟ	Navigation	
VirtualPageView	Ⓥ	Submit	id

Component Communication Model In this model component collaborations are shown for each *Submit* between pages in the NM. There are metamodel elements of the CCM, *ClientPageLifeline*, *ServerPageLifeline* and *ComponentLifeline* which are life lines of a client page, a server page, and a component, respectively. (Table 2)

Table 2. Elements in Component Communication Model

Node Type	Icon
ClientpageLifeline	cpName P_c
ServerpageLifeline	spName P_s
ComponentLifeline	cmpName C

3.2 Agile Process

We propose a development process for Web applications considering the quick-to-market property and the frequent requirement changes of Web environment. To support the quick-to-market property, our process is composed of a small number of activities, compared with heavy-weight processes such as RUP [11]. To allow frequent requirement changes, a development process should be agile. Thus we split the whole development into small development cycles and make our process adaptable to the requirement changes.

Our development process takes the advantages of a model-driven approach and a test-driven approach [12] simultaneously. Therefore we apply test-driven principles to implementation activities, while high-level design decisions such as Web application architecture, inter-component communications and navigation structures are described by models.

We employ SPEM [13] to describe our process. SPEM is a UML profile, and accordingly it is expected to help stakeholders understand our process easily. The process helps process roles develop a Web application iteratively and incrementally. A process role indicates a person or group of persons who participates in development. Several process roles are identified in the development of Web applications: software analyst, domain expert, UI developer, component developer, tester, architect, DB developer and client.

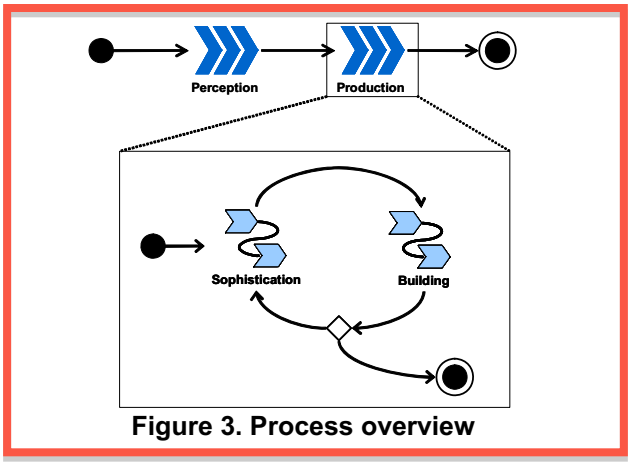


Figure 3. Process overview

The process has two phases: Perception phase and Production phase (Figure 3). Perception phase is a set of preliminary analysis activities, and is carried out at the outset. Production phase consists of detailed design activities and actual implementation activities, and becomes a basic unit of development cycle. Each development cycle aims to implement a small part of functionalities. Developers iterate Production phase until the development is completed.

Perception Phase Given problem statements from a client, a software analyst makes user requirements. With the user requirements, the software analyst draws a conceptual model in cooperation with a domain expert. Finally an architect designs an initial architecture using UML 2.0 component diagram.

Production Phase Production phase consists of Sophistication subphase and Building subphase. Subphase inherits from WorkDefinition class of SPEM.

Sophistication subphase is a set of detailed design activities and implementation activities of database layer. As Production phase is a unit of development cycle, the software analyst sets a goal of the current development cycle. A goal is defined as a subset of user requirements, which is to be developed in the current development cycle. Requirement changes are considered when the software analyst sets the goal. Based on the goal, selected reusable component list, DB schema and test cases are drawn by a component developer, a DB developer and a tester, respectively.

Then the software analyst writes a storyboard. A storyboard is a work product that shows how the part of Web application developed in the current development cycle interacts with users. The storyboard contains only functional concerns. For each interaction drawn in the storyboard, *boundCompID*s of corresponding components are annotated.

Based on the storyboard, a component developer draws out a NM, and validates the navigations in the model against the goal and the requirements. If there are erroneous navigations, the storyboard is modified by fixing the problematic navigations. Next, the client confirms that the storyboard is consistent with user requirements. If there is any inconsistency, the software analyst modifies the storyboard. Then CCMs for *Submit* transitions in the NM are drawn out. The CCM is referred to in implementation activities as it describes component interfaces.

Architecture and UI prototype are produced by using the information of the storyboard. Architecture is designed by arranging the components of UML component diagram. To consider non-functional requirements, Architect refers to the user requirements while designing architecture. The client confirms UI attributes such as layout and images, etc. Figure 4 shows a part of activities in Sophistication subphase.

In Building subphase, the Web application is actually implemented following the principle of test-driven approach: only if the written code passes all the test cases, new codes are written. The component developer implements the components, which constitute a business logic layer of the currently developed Web application. Then the components, UI prototype and DB system are combined into a subsystem of Web application. Finally the subsystem is integrated with that of the previous development cycles. All of the integration activities are performed with testing. If the current Web application does not pass all of the integration test cases, the corresponding goal is set to be a goal of the next development cycle.

We can draw out that our process is agile, considering the followings. First, client participates in the early stage of the development and developers get feedback from the client in our process. If requirements are changed during client validation, they are resolved at the beginning of the development cycle. Because each development cycle ends in short period, the requirement changes are implemented rapidly. Next, our process also emphasizes human factors [9] such as collaboration, competence of team members, and so on, which aids agile development. Finally, because agile development discourages many modeling activities, only essential modeling activities are performed.

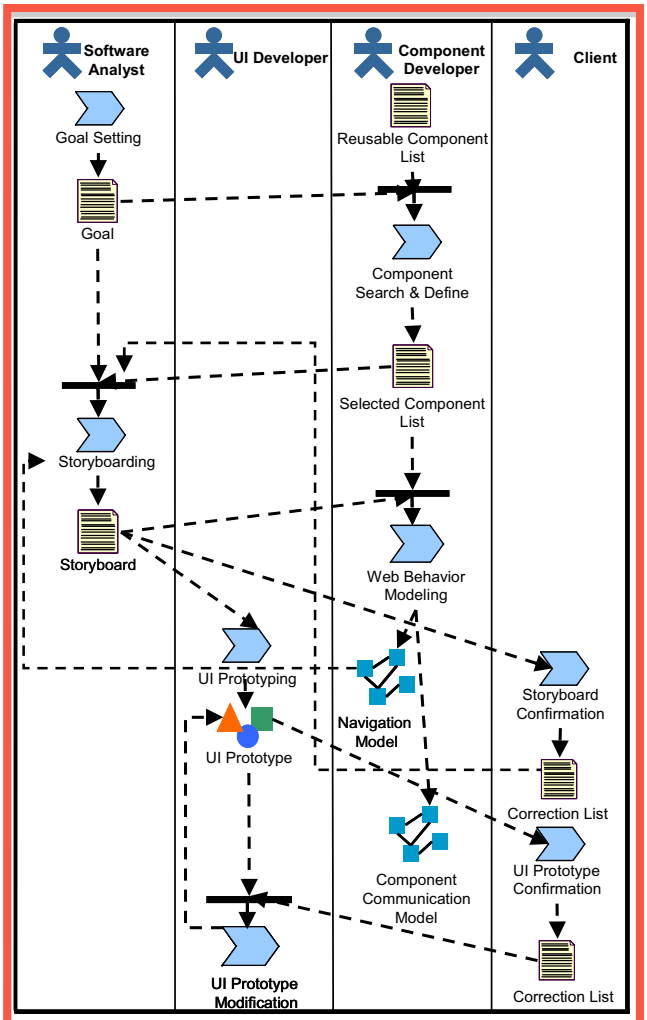


Figure 4. A part of Sophistication subphase

4. Tools for Agile Development

4.1 Process Manager

Our process is executed by Process Manager, which leads developers to follow the process by indicating a proper activity and tools. Process execution denotes that activities are presented, tools are activated, and necessary guidelines are shown to developers. Developers may

follow a new development process easily through this process execution.

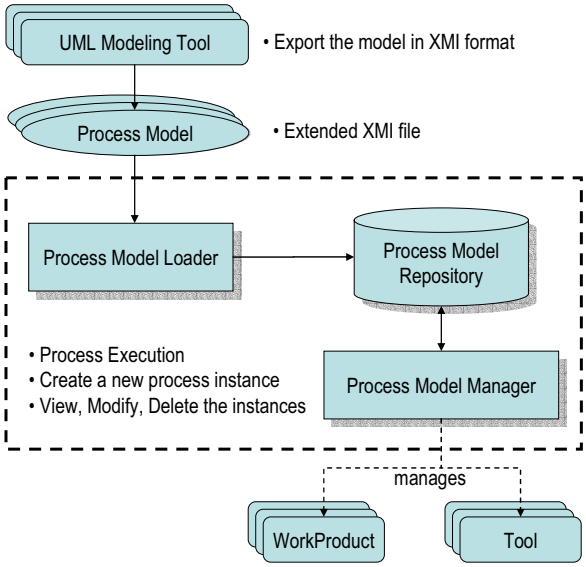


Figure 5. Process Manager structure

Table 3. Exported XMI file of a part of Sophistication subphase

```
<XMI xmi.version = '1.1'
    xmlns:UML='href://org.omg/UML/1.3'>
…
<UML:ActivityGraph xmi.id = 'S.231.1305.16.1'
     name = 'State/Activity Model' context = 'G.0' >
     <UML:StateMachine.top>
      <UML:CompositeState xmi.id = 'XX.20.135.17.2'
        name = '{top}' >
        <UML:CompositeState.subvertex>
          <UML:SimpleState xmi.id = 'G.1' name = 'Goal'
            outgoing = 'G.2 G.3 G.4' incoming ='G.11' />
                ...
          <UML:SimpleState xmi.id = 'G.9'
            name = 'Storyboard' incoming = 'G.13' />
                ...
          <UML:ActionState xmi.id = 'G.12'
            name = 'Storyboarding'
            outgoing = 'G.13' incoming = 'G.2 G.17' >
                  ...
          </UML:ActionState>
        </UML:CompositeState.subvertex>
      </UML:CompositeState>
     </UML:StateMachine.top>
     <UML:StateMachine.transitions>
      <UML:Transition xmi.id = 'G.2'
        source = 'G.1' target = 'G.12' />
           ...
     </UML:StateMachine.transitions>
   </UML:ActivityGraph>…</XMI>
```

Figure 5 shows a structure of Process Manager. The Process Manager consists of Process Model Loader and Process Model Executor. A process model is an instance of SPEM profile extended from UML to use traditional UML modeling tools, and is exported to XMI files. Then the model is loaded into Process Model Repository by Process Model Loader.

Table 3 is an example of the exported XMI file which contains a part of Figure 4. When we focus on the Storyboarding activity, there are several input or output workproducts, such as Goal and Storyboard. This information is written in the exported XMI file in Table 3, such as Storyboarding ActionState (G.12), Goal SimpleState (G.1), and Storyboard SimpleState (G.9). The XMI file also contains transitions between the states, such as G.2 from Goal SimpleState to Storyboarding ActionState.

Process Manager shows projects which are finished or in progress. If the software analyst chooses a project, Process Manger indicates which activity is carried out in turn and executes a corresponding tool. Process Manager knows the progress of project by examining the status of workproducts of each activity. Process Manager also delivers detailed steps of activities to developers. Examples or templates of workproducts may be delivered.

4.2 Web Modeler

To support Web behavior modeling in Figure 4, we have extended an open source modeling tool, ArgoUML [14]. ArgoUML is a well-known modeling tool and has many modeling functions. ArgoUML also provides an extension mechanism which is called as ‘module’. So we have implemented a module supporting our models.

Because ArgoUML currently supports only UML 1.3, there are many limitations to express our model extended from UML 2.0. Therefore we have added our model elements and diagrams by extending elements of UML 1.3 instead of UML 2.0.

Figure 6 shows a flow of modeling activities supported by Web Modeler. Storyboarding activity produces an xml file which contains information about pages, navigations, and components. Then Navigation Modeler and Component Communication Modeler use the xml file.

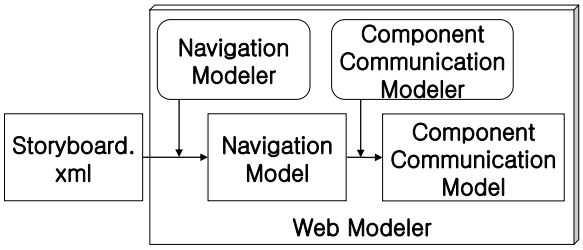


Figure 6. Model generation flow

Navigation Modeler This tool automatically generates NMs from the xml file of the storyboard which contains information of components, navigation, and client page or server page specification. The extracted NM is used to validate a navigation structure of a storyboard by the component developer.

Component Communication Modeler This tool builds CCM which helps developers decide component interfaces used in implementation. The elements of the model, *ClientPageLifeline*, *ServerPageLifeline* and *ComponentLifeline,* extend *ClassifierRole* of UML 1.3 Sequence Diagram, not *Lifeline* class of UML 2.0.

Each *Submit* element of the NM has an association with a corresponding CCM because the *Submit* indicates that components participate in the navigation. The component developer builds a corresponding CCM from a *Submit* element. The identifiers of the participating components are fetched from the NM. The component developer describes communication sequences between components and pages by using CCM.

5. A Case Study

We have implemented 'University Asset Management System (UNIVAS)' as a case study of our development methodology. UNIVAS is a Web application that manages assets of various types such as article and component, etc. Anonymous users can search and download assets from UNIVAS. Members are able to upload and manage assets.

UNIVAS has been built through several iterations of development cycle. Perception phase was carried out first, and each module of UNIVAS has been completed incrementally. We have employed the Process Manager for overall management of the development, and the Web Modeler for Web behavior modeling activities.

In the first development cycle, we implemented a module for article management and member management. We show the second development cycle that has aimed a module for component management.

Some functions of the second module such as member management function are shared with the first module, and the other functions are also similar. But there is an upgrade function only in the second module, which is a focus of this case study. User requirements related to the upgrade function are partly shown in Table 4. As there is no specific requirement for error handling, all errors occurred in upgrading function are forwarded to the error page.

Process Manager can check for each process role whether the required work products are ready or not. Developers can also get descriptions about the activities.

Table 4. User Requirements Specification

UseCase UC2: Upgrade A Component
Scope: UNIVAS Component Management Module
Level: user goal
Primary Actor: Member (Authorized User)
Main Success Scenario(or Basic Flow):
...
2. Member should fill the *-marked fields, and may fill the other fields. Then Member clicks the 'find file to be uploaded' button, and find the file to fill the 'file' field, and click 'Submit' button.
3. If the upgrade process succeeds, the retrieval result page will appear with the message 'Successfully Upgraded'.

Following the process, the software analyst set a goal for the current development cycle at the beginning of Production phase considering user requirement specification produced in Perception phase. Figure 7 shows a screenshot of the Process Manager guiding Goal Setting activity, where input workproducts are user requirements and output workproducts are goals.

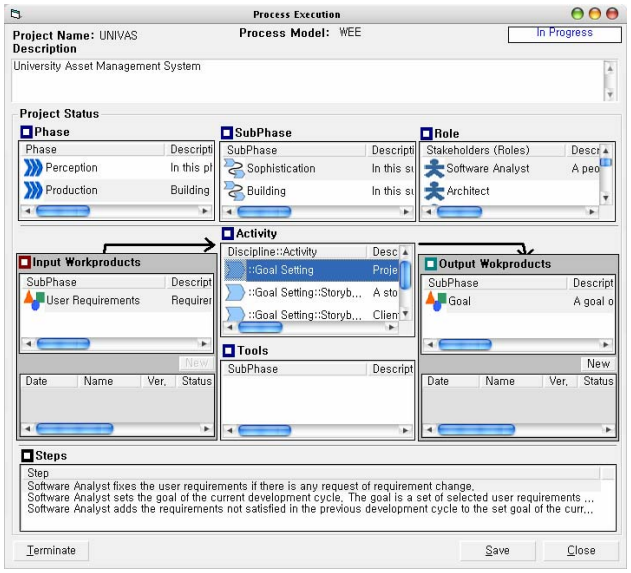


Figure 7. A screenshot of Process Manager

Then the software analyst has drawn out a storyboard. Figure 8 shows a page which shows an html form for the upgrade function, and this page draws input fields of page and indicates reactions to user's action. The software analyst has produced a storyboard xml file at the end of Storyboarding activity.

To validate the storyboard, NM was produced. Figure 9 is a NM initially derived from the storyboard xml file. We found a problematic navigation: a user restarts from index.html if an error occurs during upgrading

components. Because we did not specify any error flow from the upgrade function in the user requirements, the software analyst directed error.html for errors in the upgardeDo.class.

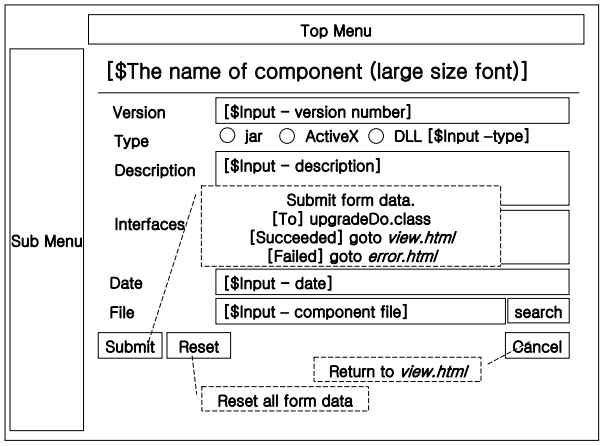


Figure 8. A part of storyboard

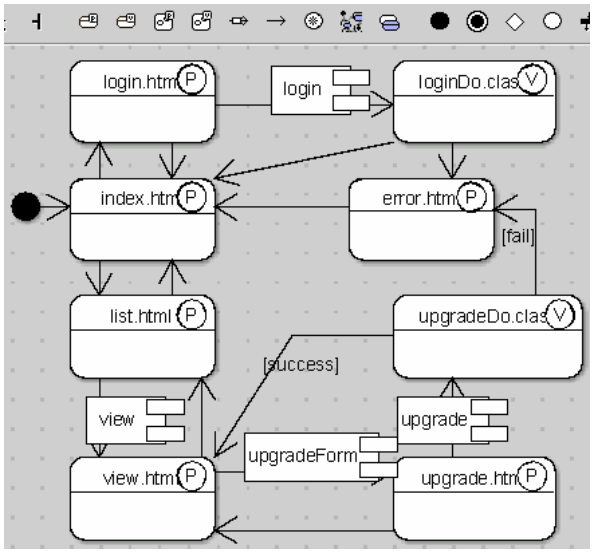
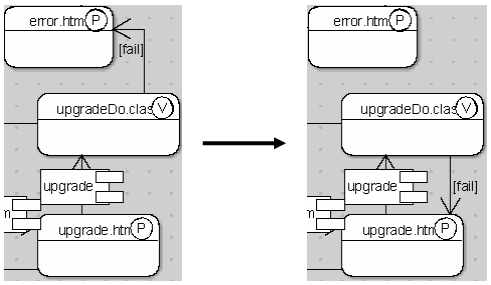


Figure 9. Navigation Model

Thus we modified the navigation: if an error occurs, go to upgrade.html page again and enable the user to retry. Figure 10 shows the modified part of Figure 9. And the change has been fed back to both the goal. Table 5 shows an added goal for error handling.

Table 5. Added Goal

… If Member clicks the 'Submit' button before filling all *-marked fields of 'Upgrade Page', which holds the value typed by Member before, the page will appear with the 'Should fill *-marked fields!' message



(a) Before modification (b) After modification

Figure 10. Modified Navigation Model

After the storyboard has been modified, the UI developer has produced UI prototypes based on the storyboard. These prototypes have been used to check the client's confirmation for the UI attributes.

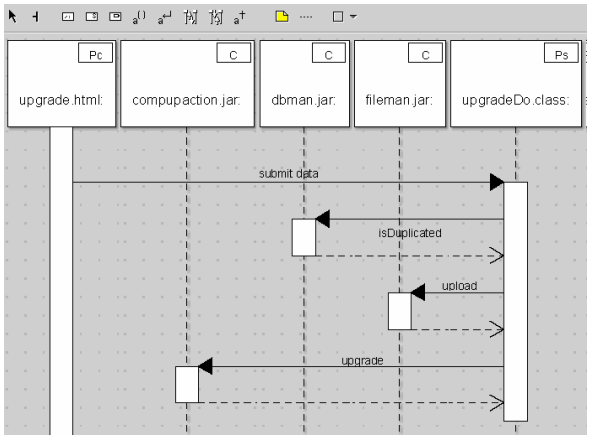


Figure 11. Component Communication Model

At the same time, the component developer has built CCM with the NM and a component list. Figure 11 shows a CCM for the upgrade *Submit* in the NM of the Figure 9. We can find the communication of components through interfaces, which have been used in implementing components.

Actual implementation activities have been done in Building subphase. Each component developer has implemented components including compupaction.jar and upgradeDo.class. Passing all the test cases, the second development cycle ended. Figure 12 shows the screenshot of the upgrade function page.

6. Discussion and Conclusion

We have proposed an agile development methodology for web application. The methodology includes Web models, an agile process, and supporting tools. The Web models focus on a behavioral view which is represented with NM and CCM. The agile process supports quick-to-

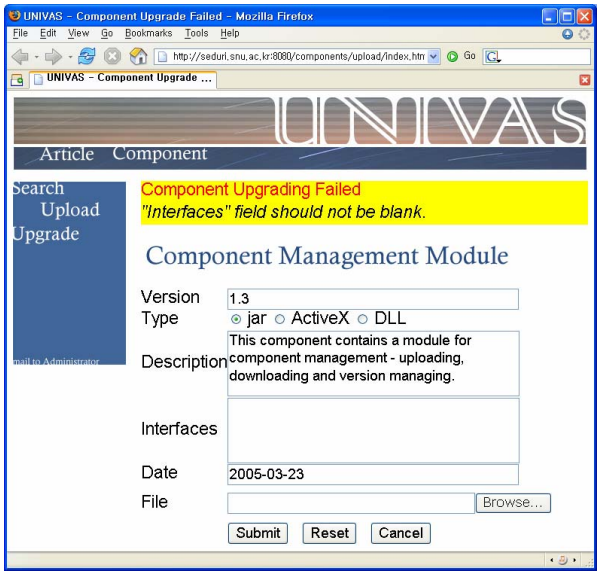


Figure 12. A screenshot of Web application

market property and adaptability to frequent requirement changes while taking advantages of modeling activities. Tools have also been implemented for process execution and Web behavior modeling. We have given a case study to validate that our methodology is agile for Web application development.

We compare our process with AWE, RUP, and XP considering five aspects (Table 6). Five aspects are the adaptability, short (rapid) lifetime cycle, visual modeling, test-driven, and project size. Some recent variations of RUP tend to support 'Short life cycle' and 'Test-driven' aspects. Therefore we classify the aspects of RUP as 'Weak'.

Table 6. Comparison with other processes

	Our process	RUP	XP	AWE
Adaptability	Yes	Yes	Yes	Yes
Short life cycle	Yes	Weak	Yes	Yes
Visual model	UML extension	UML	No	WAE
Test-driven	Yes	Weak	Yes	Yes
Project size	Small-medium	Large	Small	N/A

Our methodology has advantages of model-driven and test-driven approaches. Advantages of model-driven processes, such as RUP, have several features like manageability, traceability, and visibility. They demand, however, lots of activities and artifacts. By defining minimal artifacts based on UML, our process follows systematic approaches which can quickly respond to frequent requirements changes. Especially our process is distinguished from other processes because it is optimized for developing Web applications.

But our methodology may not be enough for inter-team collaboration in the large-scale project. Thus it may not be effective to apply this process to large scale projects with many teams.

Storyboarding tool and model checker are being implemented. In the future, we plan to make Web Engineering Environment which support an overall development process, and validate the process and the environment by applying to real projects.

References

[1] D.J. Reifer, "Web Development: Estimating Quick-to-Market Software", *IEEE Software*, Vol. 17, No. 6, pp. 57-64, 2000.

[2] D. Schwabe, G. Rossi, L. Esmeraldo, and F. Lyardet, "Engineering Web Applications for Reuse", *IEEE Multimedia*, Vol. 8, pp. 2-12, 2001.

[3] N. Koch and A. Kraus, "Towards a Common Metamodel for the Development of Web Applications", *ICWE 2003,* LNCS 2722, pp. 497-506, July 2003.

[4] K. Beck, *Extreme Programming Explained: Embrace Change,* Addison-Wesley, 2001.

[5] A. McDonald, and R. Welland, "Agile Web Engineering (AWE) Process", *Technical Report*, University of Glasgow, Scotland, 2001.

[6] S. Ceri, P. Franternali, A. Bongio, "Web Modeling Language (WebML): a modeling language for designing Web sites", *Computer Networks,* Vol. 33, pp. 137-157, 2000.

[7] S. Selmi, N. Kraiem, and H. Ghezala, "Toward a Comprehension View of Web Engineering", *ICWE 2005*, LNCS 3579, pp. 19-29, 2005.

[8] K. Schwabwer and M. Beedle, *Agile Software Development with SCRUM,* Prentice-Hall, 2001.

[9] R. Pressman, *Software Engineering, 6th Ed.,* McGraw-Hill, 2005.

[10] *Unified Modeling Language, Ver 2.0,* OMG, 2005.

[11] P. Kruchten, *The Rational Unified Process: An Introduction, 2nd ed.,* Addison-Wesley Professional, 2000.

[12] K. Beck, *Test Driven Development: By Example,* Addison Wesley, 2002.

[13] *Software Process Engineering Metatmodel, Ver. 1.1,* OMG, 2005.

[14] ArgoUML, http://argouml.tigris.org/