Chapter XV
Modeling Variant User Interfaces for Web-Based Software Product Lines

Suet Chun Lee
BUSINEX, Inc., USA

ABSTRACT

Software product line (SPL) is a software engineering paradigm for software development. A software product within a product line often has specific functionalities that are not common to all other products within the product line. Those specific functionalities are termed “variant features” in a product line. SPL paradigm involves the modeling of variant features. However, little work in SPL investigates and addresses the modeling of variant features specific to user interface (UI). Unified Modeling Language (UML) is the de facto modeling language for object-oriented software systems. It is known that UML needs better support in modeling UIs. Thus, much research developed UML extensions to improve UML support in modeling UIs. Yet little of this work is related to developing such extensions for modeling UIs for SPLs in which variant features specific to UI modeling must be addressed. This research develops a UML extension -Web User Interface Modeling Language (WUIML) to address these problems. WUIML defines elements for modeling variant features specific to user interfaces for Web-based SPLs. The model elements in WUIML extend from the metaclass and BasicActivity of the UML2.0 metamodel. WUIML integrates the modeling of variant features specific to user interfaces to UML. For example, in a Web-based patient registration software product line, member products targeting British users may use British date format in the user interface, while member products targeting United States users may use United States date format in the user interface. Thus, this is a variant feature for this product line. WUIML defines a model element, XOR, to represent such exclusive or conditions in a product line user interface model. WUIML would reduce SPL engineers’ efforts needed in UI development. To validate the WUIML research outcome, a case study was conducted. The results of this empirical study indicate that modeling UIs for Web-based SPLs using WUIML is more effective and efficient than using standard UML.
INTRODUCTION

Software product line (SPL) (Chastek, Donohoe, Kang, & Thiel, 2001; Clements & Northrop, 2002; SEI, 2005a) is a software engineering paradigm to develop software products. One important step in the SPL paradigm is the modeling of the functional features of software products across the product line. The features are called common core. An even more important step in the SPL paradigm is the modeling of the specific functional features within a particular member product in a product line. These specific functional features are called variant features because they are the features that differentiate member products in the product line. Then based on the model, a product is ‘assembled’ by reusing the common core and selected variant features.

Unified Modeling Language (UML) (OMG, 2003b, 2004) (Rumbaugh, Jacobson, & Booch, 2005) is a standard object-oriented modeling language. UML includes multiple views and diagram types to capture software functionalities from user perspective. However, UML seems to have not been developed for modeling user interface specific issues (Kovacevic, June 1998; Silva & Paton, 2003). One of the usages of user interface models is that in model-based user interface management systems (MB-UIMSs) (Pedro A. Szekely, Piyawadee Noi Sukaviriya, Pablo Castells, Jeyakumar Muthukumarasamy, & Salcher, 1996; Tony Griffiths et al., September 1999), user interface models can be used to generate user interface codes. There are extensions of UML (Blankenhorn & Jeckle, 2004) (Nunes, 2003) (Silva, 2002) to make UML better support user interface modeling. Yet these extensions often assume the modeling of a single system instead of a SPL. On the other hand, although standard UML (OMG, 2003b, 2004) seems to have not been developed to support the modeling of SPLs, there are works (Gomaa, 2004; Gomaa & Gianturco, 2002; Tewfik Ziad, Loïc Héloüët, & Jézéquel, May 2004) on extending UML to improve UML supports in modeling SPLs. Yet these works do not focus on user interface modeling. Currently, many software products are Web-based. However, some (Silva, 2002) observe that there are specific modeling challenges for modeling user interfaces of Web-based software systems.

Thus, it is not clear how to model variant features for user interface specific issues in Web-based software product lines. This is an important barrier to overcome if software product line development of Web-based products is to take greater advantage of software reuse objectives: increased quality, decreased effort, or decreased time to market. Therefore, this paper is concerned with reporting research about developing a UML extension, Web User Interface Modeling Language (WUIML) that decreases effort by increasing effectiveness and efficiency needed in using UML to model user interfaces for Web-based software product lines. User interface development has been found (Myers, 1989) to account for a significant amount of overall software development work. WUIML would improve SPL software engineering paradigm in the user interface development perspective by reducing engineers’ efforts needed in user interface modeling.

BACKGROUND AND RELATED WORK

Unified Modeling Language


UML is a standardized notation for object-oriented development. UML consists of views, diagrams, model elements, and general mecha-
nisms. Views are used to present different aspects of complex systems from both the ‘system’ in the problem and the ‘system’ in the solution. Tacitly, UML defaults to ‘system’ as meaning the system in the solution. This is consistent with the classical use cases modeling and design process model. Therefore, whenever the term ‘system’ is used in this paper, this usual UML use cases default as ‘system in the solution’ is implied. Moreover, each view requires a number of diagrams, such as use case diagram, class diagram, and interaction diagrams. Each diagram captures a particular aspect of a system. Views that are often used with UML include: use case view, logical view, implementation view, process view, and deployment view. Use case views describe the functionality a system should deliver from external actors’ point of view. An actor is a human user or another system external to the system. A use case is often used to describe functions requested for a system. Logical views are used to describe a system’s functionality. Use cases often are the first set of models being created for a system. However, some found common pitfalls in use cases for one who is new to use case modeling (Lilly, 2000; Rosenberg & Scott, 2001).

Modeling concepts are captured in model elements. Example model elements are class, object, state, node, package, and component (Eriksson, Penker, Lyons, & Fado, 2004). Model elements are used in diagrams. A model element can have a visual representation. There are rules that limit types of model elements for different diagrams. Relationships between model elements are specified by relationships, such as association, generalization, dependency, and aggregation.

Additional information that cannot be represented using model elements is represented using UML general mechanisms, such as adornments, comments, tagged values, and properties (Eriksson et al., 2004). Adornments are used to add visual impact to the element. For example, “underline” an instance of a type is an adornment. Comments can be placed anywhere in a diagram. Comments often contain explanations or questions to resolve issues at a later time. Model elements have properties to store data values about an element. A property is defined with a name and a value called a tagged value. Properties are used to add additional specifications about element instances that are not normally shown in the diagram.

UML can be extended or adapted by its extension mechanisms: stereotype, tagged values, and constraints. A stereotype defines a new kind of model element based on existing model elements. A stereotype is described by placing its name as a string and within a pair of guillemots. For example, a class with the stereotype <<Radio>> is read as “a class of the Radio stereotype,” meaning that it is a radio type of class. The particular characteristics of a Radio class must have been defined when the stereotype is defined. A constraint is a restriction on an element that limits the usage of the elements. A constraint is enclosed in a curly bracket. For example, {student = “Dorothy”}. Alternatively, constraints can be written in Object Constraint Language (OCL) (OMG, 2003a). A UML profile (OMG, 2003b) is a stereotyped package that contains model elements that have been customized for a specific purpose using extension mechanisms, such as stereotypes, tagged values, and constraints. A profile can only extend an existing UML model element using stereotypes, tagged values, or constraints. A profile cannot create new model elements that were not previously defined in the UML 2.0 infrastructure specification (OMG, 2003b).

UML 2.0 is defined in Meta-Object-Facility (MOF) (OMG, 2002). MOF is an abstract language for defining modeling languages. Therefore, another way to extend UML is using MOF. MOF extends UML by defining new metamodel elements. This approach is often called metamodel approach. In metamodel approach, new model elements can be created. With metamodel approach, one can also create model elements that change UML symbols and semantics.
**Software Product Line**

Software product line (SPL) (SEI, 2005a) is a software engineering paradigm (SEI, 2005b). The paradigm mainly includes two software engineering processes: product line engineering and member product engineering. The product line engineering process involves a number of techniques in software engineering, such as domain analysis, requirements engineering, product architecture, and component development (Clements & Northrop, 2002). The product engineering process involves the configuration (‘assemble’) of member products from the product line assets. The SPL paradigm depends on the notion of variability and variability mechanisms. In SPL, the term variability is used to indicate the extent of differences in user visible software properties among member products. A user visible software property is a feature. A feature of a member product that is different from other member products is a variant feature. Therefore, a variant feature is a feature only to one or more (but not all) products in a SPL. Variability mechanisms are a set of methods and steps that can be applied to develop variant features for SPLs.

The user interface of member products of a SPL may vary in one or more user interface specific features. Example user interface specific features are the layout of the user interface, the functionality of the user interface, the interaction of the user interface to and from the user, etc. The user interface specific variant features, like other non-user interface specific features, also need to be captured in a SPL model.

It seems that issues in user interface developments have been largely ignored by the SPL researches. It is not clear how to go about applying the software product line paradigm in user interface modeling. WUIML developed in this research is differentiated from other related works (Gomaa, 2004; Gomaa & Gianturco, 2002; Shin, 2002; Webber, 2001), (Tewfik Ziadi et al., May 2004) in SPL researches in that WUIML is addressing the modeling of user interfaces in a SPL using UML. It is known (Silva & Paton, 2003) that UML needs better support in modeling user interfaces. The related works extend UML to improve support on modeling SPL, yet those works have not addressed the modeling issues in user interfaces for SPL. User interface development has been found to (Myers, 1989) account for a significant amount of overall software development work. WUIML would improve SPL software engineering paradigm in the user interface development perspective by reducing engineers’ efforts needed in user interface modeling.

**Web User Interface Modeling**

User interface is a computer-based media to facilitate communication between users and a software system (Marcus, 2002). Early user interfaces in software systems are text or form-based. Modern software systems however use graphical user interfaces (GUI) implemented according to programming packages, such as Java Swings classes (Eckstein, Loy, & Wood, 1998). More recently, HTML is used to create Web-based user interfaces that can be rendered by Internet browsers (Sommerville, 2001).

The user interfaces of a Web-based software product are called Web user interface (WUI) in this research. The modeling of a WUI is the development of an abstract understanding of the requirements of the WUI and represents it in an abstract notation. The modeling of WUI using WUIML does not provide a visual look of a WUI, thus, WUI modeling is not the design of a WUI. Currently, a major use of WUI modeling is to provide inputs for WUI code generation. In particular, WUI models resulting from WUI modeling are fed into a model-based tool. The tool either auto-generates (or semi-auto-generates) WUI implementations for the WUI models (Behrens, Ritter, & Six, 2003)(Gómez & Cachero, 2003).
UML-Based User Interface Modeling


A Web UI development method is proposed in (Behrens et al., 2003). The method consists of two tasks: UI requirements modeling and mapping UI requirements model to UI architecture. The modeling of UI requirements includes the specification of use cases (called ‘work units’ in the paper) and then the work units are modeled using user-defined UML stereotypes representing user interface elements. A UI element is modeled as a <<scene>>. A <<scene>> may consist of zero or more <<class view>>. A user interface is then the composition of different <<scene>> elements. The navigations between user interfaces are based on the parent-child relationship between the user interfaces, domain requirements, and usability aspects. In (Hennicker & Koch, 2001), a UML profile for modeling user interfaces for Web applications is proposed. The UML profile includes UML stereotypes to model the presentation (‘look and feel’) and navigation aspects of Web UIs. The Object-Oriented Hypermedia (OO-H) Method (Gómez & Cachero, 2003) extends UML with a set of new views for modeling Web interface model. In (Silva & Paton, 2003), a UML extension, UMLi, was developed to support user interface modeling for interactive systems. However, UMLi has yet to address Web UI as indicated in (Silva, 2002) that there are additional user interface modeling properties for Web applications.

The major difference between this work and the related works (Gómez & Cachero, 2003) (Hennicker & Koch, 2001) (Behrens et al., 2003) (Silva, 2002) (Silva & Paton, 2003) (Jacobson et al., 1992) in user interface modeling is that WUIML addresses the modeling of user interfaces for SPLs and the related works focus on the modeling of user interfaces for a single system. In addition, the modeling elements in WUIML capture the Web user interface properties specific to Web software products according to important Web standards (Bos, Celik, Hickson, & Lie, 2004; Dubinko, Leigh, Klotz, Merrick, & Raman, 2002; Jonny Axelsson et al., 2003; Lie & Bos, 1999; Meyer, 2000, May 2000; W3C). WUIML improves SPL software engineering paradigm by reducing efforts needed by SPL engineers in user interface modeling for SPLs.

WEB USER INTERFACE MODELING LANGUAGE (WUIML)

WUIML improves UML support on user interface modeling for Web-based software product lines. User interface modeling falls into the scope of user interface development. The user interface models specified in WUIML form one generic user interface software product line asset for the entire main software product line. The idea is to have user interface models as core assets. Then to develop user interfaces for member products one would reuse all common user interface components and reuse selectively the variant user interface components.

In this research, the requirements analysis process for user interfaces is as follows. Given software requirements for a software product line, user interface related use cases from product line requirements are derived. A use case that is user interface related can be developed into many user interface scenarios. Each scenario is due to one or more variant user interface aspects. A user interface scenario involves one or more user interfaces. Any interaction between a human user and a user interface or between a user interface and a back-end application is also captured in the use case scenario. Once the user interfaces are identified, one can decide on the user interface components. After eliciting and describing the scenarios, the
next step is to identify the relationships between user interfaces. The interacting relationships between user interfaces are captured in extended activity diagrams where nodes in the activity diagrams can represent variant user interfaces. The extended activity diagram shows the variant interactions between user interfaces.

In order to specify common and variant features for user interfaces, WUIML defines stereotyped relationships between modeling elements. These new relationships are selected syntheses and extensions from Jacobson, Griss, and Jonsson (Jacobson, Griss, & Jonsson, 1997) and Anastasopoulos’ (Anastasopoulos & Gacek, 2001) variability mechanisms. In addition, the variant notations and rules of the FODA (Cohen, Stanley, Peterson, & Krut Jr., 1992) technique have been adopted and incorporated into WUIML. In addition, this research developed a new variability mechanism called WUIAggregation. WUIAggregation models a WUI that is an aggregation of different parts of other WUIs. For example, a portal WUI is a WUIAggregation of a number of other WUIs. The difference between WUIAggregation and the UML aggregation is that WUIAggregation limits its aggregated components to be instances of WUIComposite or specific WUIElements while the standard UML aggregation does not have this restriction. This restriction helps modelers on deciding what components are appropriate for the aggregation. The variability mechanisms are used to generate variants on user interface aspects. User interface aspects include (but not limited to) user functional requirements, style, presentation, layout, events, data model, and constraints.

The structural contents of WUIML are built from various World Wide Web Consortium (W3C) (see http://www.w3c.org) specifications: XHTML (Altheim, Boumphrey, McCarron, Schnitzenbaumer, & Wugofski, 2001; Altheim & Shane McCarron, 2001; Jonny Axelsson et al., 2003; Powell, 2003; Raggett, Hors, & Jacobs, 1999; Sauers & Wyke, 2001; W3C, 2002), XForms (Dubinko, 2003; Dubinko et al., 2002; Khare, 2000), and Cascading Style Sheets (CSS) (Bos et al., 2004; Lie & Bos, 1999; Meyer, 2000, 2003; Schengili-Roberts, 2004; Schmitt, 2003).

WUIML is also built on the UML 2.0 specifications (OMG, 2003b, 2004) and is an extension to the metamodeling approach of the UML 2.0. In particular, WUIML extends metaclass and BasicActivity of the UML 2.0 metamodel. The metaclass extension is achieved via stereotype extension mechanism. The extension to the BasicActivity is achieved via MOF.

A WUI structurally consists of multiple user interface elements, such as user interface visual elements (such as a button, a checkbox, etc.), hypertext links, images, text, etc. These elements are ‘mapped’ to stereotyped UML classes that extend the UML2.0 metaclass.

Events (sometimes termed ‘triggers’) occur when user interface elements interact with user actions. Actions are mapped to UML interfaces. An interface defines a set of operations triggered by events. Events and actions characterize a user interface interaction. Events and actions are modeled as operations of metamodel elements. Interactions between user interfaces are modeled using UML activity diagrams. In addition, UML activity diagram is extended to model variant interactions in WUIML.

**Basic WUIML Elements**

Figure 1 defines the Basic WUIML Elements in UML 2.0 notations. These model elements extend from the metaclass of the UML2.0 metamodel. These model elements have attributes and operations. The attributes are used to model the static features of user interfaces; for example, an attribute can be used to describe the background color of a user interface element. The operations are used to model the dynamic aspects of user interfaces, for example, events (e.g., a user clicks a button can be tracked by an event listener) and actions (e.g., an error message box popup can
be implemented by a method) can be modeled as operations. Figure 2 shows the class diagram of the WUIElement and the Command model elements.

A WUI is modeled as a WUITemplate. A WUITemplate may be composed of one or more WUIComposite. A WUITemplate may also be composed of one or more WUIElement and one or more WUIComposite. A WUIComposite has one or more specific WUI modeling elements (such as Submit, Input, etc.) that extend from the WUIElement. The style, layout, and position within a WUI of a WUIElement can be described using the StyleDecorator model element. A StyleDecorator composes of a number of Dimension element, Position element, and the Style element. The Dimension (how large or small the WUI element is), Position (where the WUI element is positioned within a WUI), and Style (such as what is the foreground color of the WUI element) elements together model abstractly the style of a WUI element. A set of StyleDecorator elements forms a StyleStrategy. A StyleStrategy element describes the style of a WUIComposite element. A set of StyleStrategy elements models the style of a WUITemplate.

These WUIML elements integrate Web properties according to important Web standards (Bos et al., 2004; Dubinko et al., 2002; Jonny Axelsson et al., 2003; Lie & Bos, 1999; Meyer, 2000, May 2000; W3C) to the modeling of Web-based user interfaces for SPLs. With regard to the modeling of the static aspects of user interfaces, what WUIML adds to the standard UML is like what application programming interfaces (APIs) add to a programming language. For example, the Swing
Modeling Variant User Interfaces for Web-Based Software Product Lines

API (Eckstein et al., 1998) provides the building blocks for programmers to code user interfaces. Similarly, WUIML provides the building blocks for modelers to model user interfaces. When using Swing, a programmer does not have to construct fundamental user interface elements, such as textboxes, radio buttons, etc. The programmer would just use them. Similarly, when using WUIML, a modeler does not have to model the fundamental user interface elements, such as textboxes, radio buttons, etc. When using standard UML, those fundamental user interface elements must be modeled first. In addition, WUIML provides a way to conveniently model the layout, size, and position (where it is within a user interface) for a particular user interface element. Currently, there are no standard way in using UML to model the layout and style of a Web-based user interface. WUIML defines a way for modeling the layout and style of a Web-based user interface.

**Example I: Web User Interface Modeling Using Basic WUIML Elements**

In a SPL, a user interface feature may vary among the member products. For example, consider a SPL of online patient registration systems. A member product to be deployed for US customers would have a user interface customized for US address...
format. On the other hand, member products to be deployed for Hong Kong customers would have a user interface customized for Hong Kong address format.

Figure 3 shows a product line Web user interface model that consists of instances of \texttt{<WUIComposite> elements. An instance of \texttt{<WUIComposite> may compose of a number of fundamental user interface elements (such as a textbox, an input box, a listbox, etc.) but they are not shown in Figure 3. A \texttt{<WUIComposite> can be considered as a portion of a user interface called user interface fragment in this research. A user interface is composed of a number of user interface fragments.

**Modeling Variant Interactions**

WUIML defines elements to model the different interactions between a user interface and its backend process. If a MVC architecture (Gamma, Helm, Johnson, & Vlissides, 1995) is the example, then the back-end application is the controller and the model. For the same functional task, such as a WUI for remittance of an online banking service SPL, each member product may implement it in a slightly different way. The difference would lead to differences in the WUI related to the task among member products. The differences in WUI would in turn lead to different patterns of user interactions to the WUI as well as different implementation of backend processes. For example, to send money to a foreign country, the WUI must collect the destination country (as country code) and currency (currency code) information from the user. On the other hand, to send money domestically within United States, the WUI does not need to collect country code or currency information. The different user interactions to a user interface in a SPL are called variant interactions. User actions, such as clicking a submit button in a WUI may trigger transitions between WUIs. In this
research, UML activity diagram is used to model the interactions between WUIs.

WUIML defines two new metamodel elements using MOF to model variant interactions. The two new elements are VariantAction and SPLAction. In UML, a node in an activity diagram is represented by the ActivityNode element. The ActivityNode element has three children: ExecutableNode, ObjectNode, and ControlNode. Object Nodes represent the objects involved in the activity diagram. The ControlNode signifies the control flow on the activity diagram; for example, a fork bar is a ControlNode. The ExecutableNode has a child, the Action element. The Action element in UML is the representation of a node that characterized by one or more actions. Since interactions are characterized by actions, WUIML extends the UML Action element to model variant interactions.

Figure 4 shows the new metamodel elements: VariantAction and SPLAction and how they relate to other elements in the activity class diagram. The VariantAction element is a specialization of the UML Action element. The SPLAction is a specialization of the Element metamodel element from the UML2.0 kernel package. A SPLAction element contains one or many Variant Action elements. An UML Activity is now also consisting of one or more of the SPLAction and VariantAction elements.

Suppose we have an activity called ‘submit order’. The ‘submit order’ activity is very common in Web-based shopping applications because a user must submit an order so that the order can be processed. But there are variants in ‘submit order’ activity. For example, most online shopping sites accept only credit card payments, thus, those applications may want ‘submit credit card order’.

Figure 4. Extension to UML 2.0 Activity
In other case, corporate purchasing often submits purchase orders, and then pays the bill after receiving the invoice. In this case, ‘submit purchase order’ may be desired. Yet another variant is in the case where customer wants to receive the purchase and check it first before make any payment, these customers may want ‘submit Cash-On-Delivery (COD) order’. Yet some merchants may offer a combination of these payment methods. There are a number of variants in the ‘submit order’ activity and we need to be able to model the variants. This paper extends UML activity diagram to model the requirements as follows.

Figure 5 shows a graphical symbol for the SPLAction. The symbol exposes the base action and its four variants actions. The shape of the base ‘submit order’ action is filled, while the variants are not. The base Submit Order action defines the common logics that are applicable across all the variants. Each variant differs from other variants by some features. Our concern is on user interface modeling, it is clear that the user interface to collect information on a credit card order is different from the user interface to collect a COD order in some way.

Sometimes there are variant logics and behaviors within a variant member product that capture different execution scenarios. Those logics and behaviors can be modeled using standard UML activity diagrams. For example, suppose there is a WUI that provides user interface elements to accept either credit card based order option or purchase order option. If a customer chooses credit card order, the next WUI displayed is to collect the customer’s credit card information. If a customer chooses purchase order based order option, then the next WUI displayed is to collect the customers purchase order related information. Thus, the software’s behavior varies due to different execution scenarios. This variant is within a product.

These variant logics and behaviors due to different execution scenarios often easy for one to confuse with the product line variant logics and behaviors. To identify the product line variants, one must consider the same particular variant logics and behaviors across multiple member products at the same time. If the variant logics and behaviors work the same across the member products, then the variant logics and behaviors are not a product line variant.
Example II: Modeling Variant Interactions

Figure 6a shows an activity diagram for a SPLAction for an online e-retailer SPL. The SPLAction consists of four variant actions: ‘Submit Credit Card Order’, ‘Submit COD Order’, ‘Submit Check Order’, and ‘Submit Purchase Order’. A member product of this product line would have the same activity but with only one variant of the submit order action. Figure 6b shows the activity diagram for a member product that allows users to submit Credit Card Orders.

WUIML Elements for Variability Mechanisms

Figure 7 shows the WUIML elements for variability mechanisms that are based on Jacobson, Griss, and Jonsson (Jacobson et al., 1997) and Anastasopoulos’ (Anastasopoulos & Gacek, 2001) variability mechanisms and the variant notations and rules of the FODA (Cohen et al., 1992) technique. Delegation, WUIDerive, WUIExtend, Use, TemplateInstantiation, and RequirePresenceOf are stereotypes extended from Dependency metaclass. WUIAggregation, Configuration, and Parameterization are defined as stereotypes of Class stereotype. WUIGeneralization is a stereotype of the Generalization metaclass. OptionalElement is a constraint to express that an element is optional in the model. XOR and OR are Constraints. A Parameterization element is composed of one or more Property. A Configuration element is composed of one or more Property and zero or more Constraint.

The Use model element is used to specify that the source user interface element (e.g. a user interface fragment to capture the travel date) depends on the target user interface element (e.g., a user interface of a calendar for setting a travel date). The Configuration model element models a set of parameters and rules. These parameters and rules manipulate the composition of the software component of a software system or the setting of software components or software systems, so that variants can be created. The Parameterization model element models a set of parameters that vary the features of a software product. The TemplateInstantiation model element is a special case of the Parameterization model element. In TemplateInstantiation, the parameters are templates. WUIExtend allows small extensions in functionality or ‘look’ to a user interface due to new requirements. WUIDerive extends UML derive stereotype to apply on all basic WUIML elements. Delegation is used to model the situation in which a Web user interface is acting as a representative of another Web user interface in performing certain operations. WUIAggregation allows one to model the situation in which various user interface fragments from various Web user interfaces are used to compose another Web user interface based on a business goal. WUIAggregation differs from UML aggregation in that the aggregated components in WUIAggregation must be instance of WUIComposite or specific WUI elements. WUIGeneralization models the ‘is-a-kind-of’ relationship between two user interfaces. WUIGeneralization differs from UML generalization in that the generalization in WUIGeneralization applies to specific WUI elements only. OptionalElement is used to specify that a
Example III: Modeling Variant Web User Interface Features

To illustrate WUIML for modeling variant WUI features, let us continue the example shown in Figure 3. But this time, we apply the variability mechanisms and using the corresponding variability mechanism model elements defined in WUIML. Figure 8 shows a more detail WUIML model for the online patient registration systems...
Modeling Variant User Interfaces for Web-Based Software Product Lines

SPL. Figure 3 shows the common user interface features across the product line. Figure 8 shows the variant user interface features among member products within the product line in addition to the common features across the product line.

Applying variability mechanisms and the corresponding WUIML notations, variant WUI features are specified in Figure 8. The variant notations in WUIML are only used in product line WUIML models to show the variant features among member products. Thus, the variant notations in WUIML are not to appear in the WUIML model of a member product. For example, notice that the USAddress and HKAddress WUIComposite elements appear in Figure 3 are now specified with exclusive or constraint. This indicates that if a member product’s WUI uses USAddress format, then it will not use the HKAddress format. TemplateInstantiation variability mechanism is used to create two variant WUIComposites of PatientInfoReporter. The patient information report can be in graphic format or summary format.

Modeling Dynamic Aspects of Web User Interfaces

In addition to interactions as described in the section above named ‘Modeling Variant Interactions’, the dynamic aspects of WUIs also include events and actions. In this paper, event literally means “something that takes place; an occurrence.” (See American Heritage® Dictionary of the English Language, Fourth Edition, Houghton Mifflin Company.). In this paper, an event means an occurrence that takes place in the user interface. Example occurrence may be a mouse click on a button, a selection on a selection box, etc. The
term action literally means “the state or process of acting or doing” (See American Heritage® Dictionary of the English Language, Fourth Edition. Houghton Mifflin Company, 2000.) In this paper, an action means doing some user interface related tasks in response to user interface events. Example action may be popping up a message box on the user interface, setting a value for a text input box, etc. Moreover, an event is observed by a user interface control, such as a button, and then the user interface control triggers an action. For example, when a button is clicked, a message box is popped up to display a message. In this scenario, the button is the user interface control that is observing the event: clicking the button. The popping up of a message box is the action being triggered by the button click event. The concept of events and actions described in this section are built on the XForms specification.

XForms specification includes a set of events and actions. Many events and actions have to do with XForms processing and are not suitable for use at modeling level. For this reason, only a selected set of events and actions are taking as the bases for building WUIML elements for modeling the dynamic nature of a user interface. In WUIML, events are modeled as operations of a basic WUIML metamodel element. Each basic WUIML metamodel element has a distinct set of events. Extra events can be added as needed by the modeler at design time as extensions. An action models the resulting behavior of a basic WUIML element in response to events. Actions are represented as operations of a basic WUIML element.
Example IV: Modeling Product Web User Interface Using WUIML

Figure 9 shows an even more detail WUIML model for the online patient registration systems SPL than Figure 8. In Figure 9, more variability mechanisms are applied to create variants. In addition to the variability mechanisms shown in Figure 8, the variability mechanisms applied in Figure 9 include WUIExtend, Use, Configuration, WUIAggregation, Delegation, and WUIDerive. Once we have a SPL WUIML model, we can derive product WUIML model from the SPL WUIML model. A product WUIML model is a WUIML model that reuses all the common features of a SPL WUIML model but reuses only selected variant features from a SPL WUIML model. Figure 10 and Figure 11 show two product WUIML models derived from Figure 9.

The variant patient registration WUIML model shown in Figure 10 is created by taking all the common model elements (i.e., Email, LoginInfo, Address, PhoneNum, NameInfo, OtherContent, PatientInfoReporter, and SubmitRequest) and selected variant user interface features (i.e., USAddress, USPhoneNum, and NameInfo_US-Style) from Figure 9.

Figure 11 shows another variant product WUIML model derived from Figure 9. Compare Figure 11 with Figure 10, Figure 11 shows more variant user interface features. Figure 11 includes the BritishDate WUIComposite that is extended from DateRegistered WUIComposite.

VALIDATION

The goal of this research is to improve UML support in modeling user interfaces for Web-based SPLs. The improvement goal is to have the new WUIML method exhibit decreased effort needed by increasing effectiveness and efficiency. The
improvement will reduce the SPL engineers’ efforts needed for developing user interfaces. To exhibit that this improvement goal is met a case study research validation method is applied to supply supporting evidence for the thesis (research hypothesis).

Case Study Research Method

Using case study (Lynch, 1999; Tellis, 1997a, 1997b; Yin, 2003) as an empirical research validation method has increasingly been adopted in software engineering research (Lee, 2004). This research uses a case study research method to validate the research hypothesis.

The rationale of a case study research method is the notion of ‘analytical generalization’. Analytical generalization depends on whether the findings of a case study can be replicated. If the findings of a case can be replicated, an analytical generalization may be drawn.

In analytical generalization, the results of a case study should be compared with a proposition. For example, in this case study, the results of a multiple-case study are to compare with proposition 1 and proposition 2 (proposition 1 and proposition 2 will be discussed later.) Multiple cases are needed to test a proposition through replications of findings in order to lead to an analytical generalization (Yin, 2003). Many research methods, such as a survey, depend on statistical generalization (Yin, 2003). In statistical generalization, an inference is made about a population (or universe) based on the empirical data collected about a sample (i.e., surveys) (Yin, 2003). Since a case study is not based on statistical generalization, a sampling logic should not be used in a case study research method. As a result, the number of cases in a multiple-case design and the typical criteria regarding sample size are irrelevant (Yin, 2003).

The evidence from multiple-case studies makes the overall study more robust (Yin, 2003). In this empirical study, multiple case studies were conducted. In a multiple-case study, each case must be selected to predict similar results (a literal
replication) or to produce contrasting results but for predictable reasons (a theoretical replication) (Yin, 2003). A literal replication explains the conditions under which a particular phenomenon is likely to be found. A theoretical replication explains the conditions when it is not likely to be found. This empirical study relies on literal replications. The two case studies conducted are expected to predict similar results. The study is trying to show that under the condition (i.e. using WUIML in modeling WUIs for a SPL), a particular phenomenon (i.e. increased modelers’ effectiveness and efficiency in modeling) is likely to be found. For each case in the multiple-case study, the study indicates the way a particular proposition was demonstrated (or not demonstrated). Across cases in the multiple-case study, the study indicates the extent of the replication logic and the reasons on the prediction of supporting or contrasting results (Yin, 2003).

**Case Study Design**

The goal of this empirical study is to investigate the effectiveness and efficiency of WUIML. This empirical study uses a case study research method (Lee, 2004; Tellis, 1997a, 1997b; Yin, 2003) as the validation method. The design of a case study is characterized by five important case study components (Yin, 2003): a study’s research questions, study propositions, units of analysis, the logic linking data (results) to the propositions, and the criteria for interpreting the findings.

**Study’s Research Questions**

The study’s research questions define the validation goal of a case study. The study’s research questions should be clarified precisely (Yin, 2003). For example, in this empirical study, the study’s research question is “Does Web user interface modeling for a Web-based medical SPL using WUIML increase SMEs’ modeling efficiency and effectiveness (thus decreases work)?” This study’s research question needs to be clarified further because the notion of efficiency and effectiveness need further clarification. This study’s research question can be decomposed into a set of propositions.

**Study Propositions**

Study propositions are derived from the study’s research questions (Yin, 2003) but are more specific than the study’s research questions. Study propositions quantify the quality variables (indirect metrics) in a study’s research question into directly measurable quantitative metrics (direct metrics or indicators). For example, in this multiple-case study, the study’s research question is decomposed into two propositions:

1. SMEs are able to correctly model a larger number of required modeling items using WUIML than standard UML in modeling WUIs for a Web-based medical SPL in approximately the same amount of person-hours.
2. SMEs are able to correctly model larger numbers of required modeling items via reuse using WUIML than standard UML in modeling WUIs for a Web-based medical SPL in approximately the same amount of person-hours.

In this multiple-case study, for each WUI, the investigator has identified from the requirements a set of modeling items, called required modeling items that must be modeled by the SMEs. The resulted models produce by SMEs are inspected by the investigator. The investigator first checks the models for required modeling items. Then, base on his modeling experience, the investigator decides the correctness of the required modeling items found in the resulted models.
Units of Analysis

Units of analysis are materials such as documents or other resources that the subject matter experts (SMEs) use as inputs or materials to apply the method or tools being validated. In this study, the method under investigation is WUIML. The units of analysis are the Web user interface requirements for a medical SPL.

In the first case study, the requirements for the Pediatric Medical Profile Login WUI, the Adolescent Medical Profile Login WUI, and the Adult Medical Profile Login WUI are provided. These WUIs are each from a different member product (Pediatric Medical Management System, Adolescent Medical Management System, and Adult Medical Management System) of a medical product line (Medical Management Systems). The WUIs to model are extracted from three Web-based medical products under development in BUSINEX Inc.

The WUIs for the medical Web software products are based on actual medical forms from health-care providers in the United States. For the first case study, three member products of the product line are considered for WUI modeling. In particular, this case study requires the SMEs to model the Medical Profile Login WUI and a related activity across three member products of the product line. This WUI is chosen because it allows one to exercise the modeling of commonality and variability found in product lines.

In the second case study, the requirements for the Pediatric Exercise Record WUI, the Adolescent Exercise Record WUI, and the Adult Exercise Record WUI are provided. These WUIs are each from the same member products in the same product line as in the first case study. These requirements are the units of analysis for the second case study.

Note that within each case study, there are two SMEs. One SME applies WUIML to the requirements to generate results while the other SME applies standard UML to the requirements to generate results. The results generated by applying standard UML are used as the baseline for analyzing the results generated by applying WUIML.

In this research, the SMEs are well trained in software engineering; they are professional software engineers with experience ranging from seven to fifteen years. Their specialties are focused on user interface development. They are representative users for the new WUIML. The investigator provides training on WUIML to the SMEs before they begin conducting the case studies. In each case, all SMEs participate in the case study only once. Not having the same SME to participate in more than one case is to prevent the introduction of bias due to the familiarity of WUIML by SMEs participating in multiple case studies. The SMEs are to carry out the case study without help from the investigator. The SMEs are also not given information on the expected results of the case study. This is to prevent bias by the SMEs in performing the case studies.

Table 1 shows the required modeling items for the Pediatric Medical Profile Login WUIs. The result of whether these items are correctly modeled by SMEs, the number of correctly modeled items over total number of required modeling items by SMEs with or without product line reuse, and the approximate person-hours spent will be used to support/reject the study propositions 1 and 2. There are specific required modeling items for Adult Medical Profile Login WUI, Pediatric Medical Profile Login WUI, Adult Exercise Record WUI, Adolescent Exercise Record WUI, and Pediatric Exercise Record WUI, for brevity they are not shown here.

Linking Data to Propositions

Data analysis is done in the step of linking data (results) to propositions. In a case study, data are collected using formal case study worksheets. A case study worksheet consists of specific concrete questions associate with each one of the study
propositions. The answers to those questions are measures to concrete criteria metrics that can be analyzed to support (or reject) the propositions. Thus, the most concrete criteria metrics are measure terms found within the questions on the case study worksheets. Figure 12 shows a case study worksheet used in the first case study for modeling the Pediatric Medical Profile Login WUI. In Figure 12, each concrete criteria metrics in questions that link to propositions is identified by a name formed by three sections. For example, “Pediatric_WUIML_personHours” is the metric about person-hours spent in using WUIML to model the Pediatric Medical Profile Login WUI. The same format is used in naming other metrics found in the questions in other case study worksheets in the case studies.

Table 2 summarizes the evidence collected through the case study worksheets in the case study about Medical Profile Login WUIs.

In each case study, WUIML and standard UML are applied to the units of analysis (that is the Web user interface requirements of a Web-based medical SPL) respectively. The results, that
is the resulting models and the completed use case worksheets, are analyzed to find out the following:
D1) How many of the required modeling items are correctly modeled when the modeling was done in WUIML? D2) how many of the required modeling items are correctly modeled when the modeling was done in standard UML? D3) how many of the required modeling items are correctly modeled via reuse when the modeling was done in WUIML? To model via reuse is to create new models by re-using previously created models or model elements. For example, suppose one previously created a model that includes a class representing fruit. Now one can reuse the fruit class to create a class that represents a specific fruit, such as an apple, by extending the fruit class. Both standard UML and WUIML allow SMEs to model via reuse. D4) how many of the required modeling items are correctly modeled via reuse when the modeling was done in standard UML? D5) how many person-hours spent to generate the WUIML models? D6) how many person-hours spent to generate the standard UML models? D7) the total number of required modeling items. D1, D2, D5, D6, and D7 link to proposition 1. D3, D4, D5, D6, and D7 link to proposition 2.

Criteria for Interpreting a Case Study’s Findings

In this empirical study, the criteria for interpreting a case study’s findings correspond to the metric and measures used in evaluating the results of applying WUIML and standard UML respectively to the units of analysis. “A measure provides a quantitative indication of the extent, amount, dimension, capacity, or size of some attribute of a product or process (Pressman, 2004).” A metric is an important directly measurable attribute of a software product, a software service, a software process or a software resource. Direct metrics may be either used as indicators (predictors), or of other more valuable outcomes that are indirect metrics. For example, the indirect metrics in this case study are efficiency and effectiveness. The direct metrics are: m1) number of the required modeling items that are correctly modeled when the modeling was done in WUIML; m2) number of the required modeling items that are correctly modeled when the modeling was done in standard UML; m3) number of the required modeling items that are correctly modeled via reuse when the modeling was done in WUIML; m4) number
Table 2. Evidence collection through the case study worksheets

<table>
<thead>
<tr>
<th>Questions (Units)</th>
<th>Evidence captured</th>
<th>Propositions to support/reject</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pediatric_WUIML_modelingTool</td>
<td>The modeling tool used to perform the modeling in WUIML for the Pediatric Medical Profile Login WUI.</td>
<td>Proposition 1 and Proposition 2</td>
</tr>
<tr>
<td>Pediatric_WUIML_modelsFilename</td>
<td>The name of the software copy of the resulted models in WUIML for the Pediatric Medical Profile Login WUI.</td>
<td>Proposition 1 and Proposition 2</td>
</tr>
<tr>
<td>Pediatric_WUIML_required_modeling_item</td>
<td>The correctly modeled required modeling items in WUIML for the Pediatric Medical Profile Login WUI.</td>
<td>Proposition 1</td>
</tr>
<tr>
<td>Pediatric_WUIML_reuse</td>
<td>The correctly modeled required modeling items via reuse in WUIML for the Pediatric Medical Profile Login WUI.</td>
<td>Proposition 2</td>
</tr>
<tr>
<td>Pediatric_WUIML_personHours</td>
<td>The person-hours spent to model the Pediatric Medical Profile Login WUI using WUIML.</td>
<td>Proposition 1 and Proposition 2</td>
</tr>
<tr>
<td>Pediatric_UML_modelingTool</td>
<td>The modeling tool used to perform the modeling in UML for the Pediatric Medical Profile Login WUI.</td>
<td>Proposition 1 and Proposition 2</td>
</tr>
<tr>
<td>Pediatric_UML_modelsFilename</td>
<td>The name of the software copy of the resulted models in UML for the Pediatric Medical Profile Login WUI.</td>
<td>Proposition 1 and Proposition 2</td>
</tr>
<tr>
<td>Pediatric_UML_required_modeling_item</td>
<td>The correctly modeled required modeling items in UML for the Pediatric Medical Profile Login WUI.</td>
<td>Proposition 1</td>
</tr>
<tr>
<td>Pediatric_UML_reuse</td>
<td>The correctly modeled required modeling items via reuse in UML for the Pediatric Medical Profile Login WUI.</td>
<td>Proposition 2</td>
</tr>
<tr>
<td>Pediatric_UML_personHours</td>
<td>The person-hours spent to model the Pediatric Medical Profile Login WUI using UML.</td>
<td>Proposition 1 and Proposition 2</td>
</tr>
<tr>
<td>Adolescent_WUIML_modelingTool</td>
<td>The modeling tool used to perform the modeling in WUIML for the Adolescent Medical Profile Login WUI.</td>
<td>Proposition 1 and Proposition 2</td>
</tr>
<tr>
<td>Adolescent_WUIML_modelsFilename</td>
<td>The name of the software copy of the resulted models in WUIML for the Adolescent Medical Profile Login WUI.</td>
<td>Proposition 1 and Proposition 2</td>
</tr>
<tr>
<td>Adolescent_WUIML_required_modeling_item</td>
<td>The correctly modeled required modeling items in WUIML for the Adolescent Medical Profile Login WUI.</td>
<td>Proposition 1</td>
</tr>
<tr>
<td>Adolescent_WUIML_reuse</td>
<td>The correctly modeled required modeling items via reuse in WUIML for the Adolescent Medical Profile Login WUI.</td>
<td>Proposition 2</td>
</tr>
<tr>
<td>Adolescent_WUIML_personHours</td>
<td>The person-hours spent to model the Adolescent Medical Profile Login WUI using WUIML.</td>
<td>Proposition 1 and Proposition 2</td>
</tr>
<tr>
<td>Adolescent_UML_modelingTool</td>
<td>The modeling tool used to perform the modeling in UML for the Adolescent Medical Profile Login WUI.</td>
<td>Proposition 1 and Proposition 2</td>
</tr>
<tr>
<td>Adolescent_UML_modelsFilename</td>
<td>The name of the software copy of the resulted models in UML for the Adolescent Medical Profile Login WUI.</td>
<td>Proposition 1 and Proposition 2</td>
</tr>
<tr>
<td>Adolescent_WUIML_required_modeling_item</td>
<td>The correctly modeled required modeling items in UML for the Adolescent Medical Profile Login WUI.</td>
<td>Proposition 1</td>
</tr>
<tr>
<td>Adolescent_UML_reuse</td>
<td>The correctly modeled required modeling items via reuse in UML for the Adolescent Medical Profile Login WUI.</td>
<td>Proposition 2</td>
</tr>
<tr>
<td>Adolescent_UML_personHours</td>
<td>The person-hours spent to model the Adolescent Medical Profile Login WUI using UML.</td>
<td>Proposition 1 and Proposition 2</td>
</tr>
</tbody>
</table>
of the required modeling items that are correctly modeled via reuse when the modeling was done in standard UML; m5) person-hours spent to generate the WUIML models; m6) person-hours spent to generate the standard UML models; m7) the total number of required modeling items.

The Case Study Procedure for both Case Studies

0. For those SMEs that need to perform the modeling in WUIML, teach them WUIML.
1. Identify the required modeling items from the software requirements.
2. Provide SMEs who are to conduct the modeling using WUIML with the software requirements.
3. Provide SMEs who are to conduct the modeling using standard UML with the software requirements.
4. Collect the results (WUIML models and the data from the completed case study worksheets) generated from step 2.
5. Collect the results (standard UML models and the data from the completed case study worksheets) generated from step 3.
6. Perform data analysis on the results and data collected from step 4 to find out the following: a) number of the correctly modeled required modeling items when the modeling was done in WUIML (the measure of m1); b) number of the correctly modeled required modeling items via reuse when the modeling was done in WUIML (the measure of m3); c) person-hours spent to generate the WUIML models (the measure of m5).
7. Perform data analysis on results and data collected from step 5 to find out the following: a) number of the correctly modeled required modeling items when the modeling was done in standard UML (the measure of m1); b) number of the correctly modeled required modeling items via reuse when the modeling was done in standard UML (the measure of m3); c) person-hours spent to generate the standard UML models (the measure of m5).
of m2); b) number of the correctly modeled required modeling items via reuse when the modeling was done in standard UML (the measure of m4); c) person-hours spent to generate the standard UML models (the measure of m6).

8. Evaluate the outcome from step 6 to determine whether or not the proposition 1 is supported or rejected.

9. Evaluate the outcome from step 7 to determine whether or not the proposition 2 is supported or rejected.

RESULTS AND ANALYSIS

WUIML is developed to improve SPL software engineering paradigm in modeling user interfaces for Web-based SPLs. The improvement provided by WUIML should decrease the work needed by the SPL engineers in the WUI modeling perspective.

In the first case study, each SME has to model the WUIs for three product lines: Pediatric Medical Profile Login, Adolescent Medical Profile Login, and Adult Medical Profile Login. One SME is asked to model each WUI using WUIML while the other SME is asked to model the WUIs using standard UML. Each WUI is from a different member product (Pediatric Medical Management System, Adolescent Medical Management System, and Adult Medical Management System) of a medical product line (Medical Management Systems).

Table 3 shows the results for modeling Pediatric Medical Profile Login WUI using standard UML and WUIML respectively. The last row in Table 3 shows the ratio of the number of correctly modeled required modeling items to the total number of required modeling items in standard UML and WUIML respectively.

Note that in each case study, there are two SMEs. One SME is asked to model using standard UML while the other SME is asked to model using WUIML. In the following tables (Table 4, 5, 6, and 7), the numbers shown in the second column from the left were derived from the models created by the SME who was asked to model using standard UML. The numbers shown in the third column from the left were derived from the models created by the SME who was asked to model using WUIML. The sample size is irrelevant in case study research method because case study research method is based on analytical generalization instead of statistical generalization (Yin, 2003). However, the results shown in these tables must replicate (or be replicated by) the results from the second case study in order to lead to an analytical generalization.

Table 4 shows the ratio of the number of correctly modeled required modeling items to the total number of required modeling items in modeling the three WUIs and the person-hours spent on each approach. Notice that the person-hours spent for modeling the Pediatric Medical Profile Login WUI was more than the person-hours spent for Adolescent Medical Profile Login WUI and Adult Medical Profile Login WUI respectively in WUIML approach. This is because to model the WUIs for a member product (such as the Pediatric Medical Profile Login WUI), one must first develop the WUI models for the SPL.

Table 5 shows the number of correctly modeled required modeling items per person-hours for the models in UML and WUIML for the three member products’ WUIs respectively.

Table 6 shows the ratio of the number of correctly modeled required modeling items via reuse (or product line reuse when WUIML is used) to the total number of reusable required modeling items in modeling the three WUIs and the person-hours spent on each approach.

Table 7 shows the number of correctly modeled required modeling items per person-hours via reuse for the models in UML and WUIML respectively.

The data in Table 3 show that 18 out of 25 required modeling items were correctly modeled
Table 3. Results for modeling the Pediatric Medical Profile Login WUI

<table>
<thead>
<tr>
<th>Required modeling items for Pediatric Medical Profile Login WUI</th>
<th>Modeled correctly in Standard UML</th>
<th>Modeled correctly in WUIML</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page title: “Pediatric Medical Profile Login”</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Label 1: “Profile ID:”</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>A textbox for label 1.</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Label 2: “Password:”</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>A textbox for label 2.</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Label 3: “Role:”</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>A radio button.</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>The radio button must default to be checked.</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>A label for the radio button: “Parent/Guardian”. (Only parent or legal guardian who are previously registered with the health provider can login for the child.).</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>A submit button with name “Login”.</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>The Page title must placed on the top of the page.</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>The Profile ID and its related textbox must be immediately next to each other.</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>The Password and its related textbox must be immediately next to each other.</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>The Role and its related radioButton must not be immediately next to each other.</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>Profile ID must be placed on top of the Password and its related textbox.</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>The Role and the radio button must be on a line that is on top of the Login button.</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>The Login button must be placed at the lower left hand side of the page.</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>The activity diagram should include an action node: Pediatric Medical Profile Login.</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>The activity diagram should include an action node: Login Error.</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>The activity diagram should include an action node: Parent Welcome.</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>The activity diagram should include an action node: Customer Service.</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>The activity diagram should include a start node.</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>The activity diagram should include an end node.</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>The activity diagram should include a decision node: whether validation is successful.</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>The activity diagram should indicate the condition that “Profile ID, Password, and Role” values are available.</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>18/25</td>
<td>24/25</td>
</tr>
</tbody>
</table>

for the Pediatric Medical Profile Login WUI using standard UML. On the other hand, the data in Table 3 show that 24 out of 25 required modeling items were modeled for the same Pediatric Medical Profile Login WUI using WUIML. This result in Table 3 indicates that a SME was able to correctly model more required modeling items using WUIML than using standard UML. Table 4 shows
the correctly modeled required modeling items and the approximate person-hours for modeling the Pediatric, Adolescent, and Adult Medical Profile Login WUI respectively in standard UML and WUIML. Table 5 shows the calculated value of modeled items per person-hour based on the data from Table 4. The data in Table 5 shows that when using WUIML, a SME was able to correctly model about four times more required modeling items per person-hour (i.e. 9.08) than that in standard UML (i.e. 2.37). Thus, this result indicates WUIML is more efficient than standard UML in modeling user interfaces for Web-based SPL, therefore, this result supports proposition 1.

Table 6 shows the number of correctly modeled required modeling items via reuse (or product line reuse) and its corresponding person-hours used. For example, using standard UML to model the WUI

### Table 4. Ratio of correctly modeled required modeling items to the total number of required modeling items

<table>
<thead>
<tr>
<th>WUI</th>
<th>Standard UML (Person-hours)</th>
<th>WUIML (Person-hours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pediatric</td>
<td>18/25 (8)</td>
<td>24/25 (7)</td>
</tr>
<tr>
<td>Adolescent</td>
<td>23/30 (10)</td>
<td>29/30 (4)</td>
</tr>
<tr>
<td>Adult</td>
<td>18/25 (7)</td>
<td>24/25 (2)</td>
</tr>
</tbody>
</table>

### Table 5. Number of correctly modeled required modeling items

<table>
<thead>
<tr>
<th></th>
<th>Standard UML model</th>
<th>WUIML model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pediatric</td>
<td>2.25</td>
<td>3.4</td>
</tr>
<tr>
<td>Adolescent</td>
<td>2.3</td>
<td>7.25</td>
</tr>
<tr>
<td>Adult</td>
<td>2.57</td>
<td>12</td>
</tr>
<tr>
<td>Average</td>
<td>2.37</td>
<td>9.08</td>
</tr>
</tbody>
</table>

### Table 6. Number of correctly modeled required modeling items via reuse

<table>
<thead>
<tr>
<th>WUI</th>
<th>Standard UML (Person-hours)</th>
<th>WUIML (Person-hours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pediatric</td>
<td>0/10 (8)</td>
<td>10/11 (3+4)</td>
</tr>
<tr>
<td>Adolescent</td>
<td>8/10 (10)</td>
<td>10/11 (4)</td>
</tr>
<tr>
<td>Adult</td>
<td>8/10 (7)</td>
<td>10/11 (2)</td>
</tr>
</tbody>
</table>

### Table 7. Number of correctly modeled required modeling items via reuse (or product line reuse) per person-hours

<table>
<thead>
<tr>
<th></th>
<th>Standard UML model</th>
<th>WUIML model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pediatric</td>
<td>0</td>
<td>1.43</td>
</tr>
<tr>
<td>Adolescent</td>
<td>0.8</td>
<td>2.5</td>
</tr>
<tr>
<td>Adult</td>
<td>1.14</td>
<td>5</td>
</tr>
<tr>
<td>Average</td>
<td>0.65</td>
<td>2.98</td>
</tr>
</tbody>
</table>
Modeling Variant User Interfaces for Web-Based Software Product Lines

Pediatric Medical Profile Login WUI, a SME was not able to reuse any model items out of the ten reusable items in the WUI model. For approximate eight person-hours, the SME has to model without reuse. This is because the Pediatric Medical Profile Login WUI was the first WUI for a SME to model. There is nothing to reuse yet. Notice that the number of reusable required modeling items is only a sub-set of the total number of required modeling items in a WUI model because some items are unique to a particular WUI and cannot be reused.

On the other hand, using WUIML, across all three WUIs (Pediatric Medical Profile Login, Adolescent Medical Profile Login, and Adult Medical Profile Login.), as shown in Table 6, number of correctly modeled required modeling items via reuse are consistent. This is because in WUIML, a SME is to first develop the product line WUI model. The WUI model for the SPL encapsulated all model items for its member product WUI models. Therefore, reusing the items in the product line WUI model creates the WUI model for a member product. As shown in Figure 7, the average number of correctly modeled required modeling items via reuse per person-hours using WUIML (i.e. 2.98) is about four times to that when using standard UML (i.e. 0.65). This result indicates that WUIML enables higher level of reuse than standard UML, thus it indicates increased effectiveness when using WUIML than using standard UML. Therefore, this result supports proposition 2.

The results of the second case study are not shown due to the limitation of the space in the paper. The second case study was conducted by another two SMEs. The units of analysis for the second case study were the software requirements for the Pediatric Exercise Record WUI, the Adolescent Exercise Record WUI, and the Adult Exercise Record WUI. The results from the second case study show that using WUIML to model the Exercise Record WUIs for the three member products of the SPL increases the modeler’s efficiency and effectiveness in modeling. This result is similar to the result from the first case study. The results of the two case studies literally replicate. Based on the results, the analytical generalization that WUIML increases modelers’ efficiency and effectiveness in model WUls for SPL is drawn in this study.

The generalization thus provides a positive answer to the study’s research question that Web user interface modeling for a Web-based medical SPL using WUIML increases SMEs’ modeling efficiency and effectiveness (thus decreases work).

CONCLUSION AND FUTURE WORK

In standard UML, there is no standard way to model user interface and no standard way to model variant features of user interfaces for a SPL. WUIML defines elements to propose a standard way to model user interfaces as well as variant user interface features of Web-based SPLs. WUIML improves UML in terms of modeling of user interfaces and use interfaces for SPL.

Case study research method has been applied to investigate whether WUIML method increases modelers’ efficiency and effectiveness on modeling a Web-based medical SPL when compared to standard UML method. The results indicate that modeling WUIs for SPL using WUIML is more effective and efficient than using standard UML.

There are many ways to model software using UML. Because UML is not formally defined, a human is needed to make the “best” judgment on the correctness of a UML model. In this empirical study, the way to determine whether a required modeling item is modeled correctly or not relies on a human’s modeling experience. Since the same human is using his same modeling experience to make the judgment on all models, the bias due to human judgment should be insignificant.
Since the SMEs are of similar backgrounds and technical experience, the impact on modeling efficiency and effectiveness due to individual SME’s capability should be insignificant. Since the same requirements are given for both approaches (WUIML and standard UML), the reuse opportunity base on application logics (according to the requirements) should be the same for all SMEs.

Future work is to develop a tool to support using WUIML to model WUIs for SPLs and to extend WUIML beyond the Web platform.

REFERENCES


This work was previously published in Int. Journal of Information Technology and Web Engineering, Vol 1, Issue 1, edited by G. Alkhatib and D. Rine, pp. 1-34, copyright 2006 by IGI Publishing (an imprint of IGI Global).