2010 34th Annual IEEE Computer Software and Applications Conference

A Usability-pattern-based Requirements-analysis Method to Bridge the Gap between User Tasks and Application Features

Sang-Hyun Lee, In-Young Ko, Sungwon Kang, Dan-Hyung Lee
Department of Information and Communications Engineering
Korea Advanced Institute of Science and Technology
Daejeon, South Korea
{jeffapril, iko, sungwon.kang, danlee}@kaist.ac.kr

***Abstract* – In software development, it is important to mediate various concerns coming from user experience (UX) designers and application developers. In Agile User-Centered Design (Agile-UCD), there is a special role called specialist who is dedicated to implement application features as well as to monitor user experiences. However, the specialist normally has difficulty in linking user tasks to be accessed via a user interface (UI) into application feature entities. In addition, the specialist may also have some unsettled usability risks that might result in the failure of meeting certain usability criteria and passing acceptance tests. To alleviate these difficulties of the specialists in Agile-UCD, we propose a usability-pattern-based requirement-analysis method. This method uses standardized and common representations of requirements specification to bridge the gap between user tasks and related application features of a UI. It also provides a guideline to allow the specialist to reduce usability risks in an early stage by reflecting usability factors of UI design patterns to an application design. A case study has been conducted to show how users can effectively specify user tasks and application features on UI workflows. It also shows how easy and practical it is to understand the common representations as well as to apply to usability patterns.**

Keywords – Agile User-Centered Design, Specialist, User Task, Application Feature, Requirements Analysis Method, Usability Pattern

I. INTRODUCTION

In the current maturing software development of rich interactive applications, there are two things to be considered. First, quality of user experience [1] needs to become an application differentiator as a key success factor in modern software development. Therefore, the user-centered design (UCD), as a complementary design process utilizing human-centered perspectives, addresses the major usability concerns for software applications [2]. Second, rapid launch of the application features needs to be done just-in-time, meeting frequently changing needs. Therefore, the Agile methods, as feature-centric software development methodologies, supply a fashion of working software providing various application features [3]. Later, upon endeavoring to add user experience (UX) and application features, a rapid user-context-based software development process is created as the Agile User-Centered Design (Agile-UCD) [4]. By linking the Agile methods and UCD practices, the Agile-UCD can better suit and complement one another while conserving both basic values and techniques.

The Agile-UCD has mainly two parallel tracks of development for UX designers and application developers independently [5]. These parallel development tracks result in additional roles and activities for bridging the development tasks of UX designers and application developers. Therefore, many Agile-UCD based software development projects enhance the role of the Agile-UCD specialist (AUS) who governs and controls the above issues [6]. However, in many references about Agile-UCD case studies and interviews with AUSs, several issues in the requirement specification phase of Agile-UCD have been raised [6][7][8][9][10][11]. Most of those issues are identified by questioning AUSs about their goals and activities. An AUS is supposed to investigate communication and collaboration between UX designers and application developers. However, he/she has difficulties in informing them with understandable common representations of user tasks and application features. Moreover, it is hard for the AUS to seamlessly bridge and separate application features from user tasks via a user interface (UI). Despite of skillful work that UX designers perform, the AUS has suffered due to the difficulties in consolidating various usability factors of functionalities in early stages.

In this paper, we present a usability-pattern-based requirement-analysis method to help AUSs do their tasks of requirements specification. It is accomplished by formalizing the representations of extended Unified Modeling Language (UML) notations. Our approach also provides a guideline to transform user tasks into a set of application features on a UI page, and then to integrate requirements analysis results with probated usability factors – usability properties and patterns.

With exploratory requirement-analysis case study using several AUSs, we show in this paper that a level of agility and usability can be maintained by using the proposed requirement-analysis method. In addition, our evaluation reveals that the proposed requirements analysis method bridges gaps between user tasks and application features without degradation of characteristics of the Agile methods and UCD practices.

The rest of the paper is organized as follows. Section II presents related work. Section III presents considerations of the usability-pattern-based requirement-analysis method. In

0730-3157/10 $26.00 © 2010 IEEE
DOI 10.1109/COMPSAC.2010.39

section IV, we explain our proposed method in detail. Section V describes our evaluation and experiment results. Finally, we discuss the conclusion and possible future work in Section VI.

II. Related Works

With the aim of bridging gaps between UX designers and application developers for rapid software developments with the consideration of end users, several object-oriented modeling techniques and UI design patterns have been attempted.

In Human-Computer Interaction (HCI) fields, with a unified framework, object-oriented modeling for UI designs has been regarded as a major approach to develop interactive software systems. The Wisdom method has extended UML notations so that it helps designers and developers to share common representations with each other [12]. It supports shared semantics of communication or outcomes for designers and developers. In addition, it is advantageous in breaking down development works concerning user tasks analysis and application system task analysis. However, it assumes that architectural issues can only be identified with misconceptions of user requirements by developers. Therefore, it has not applied the UX design patterns with architectural perspectives of usability attributes in the early stage of requirement engineering.

The Another Dimension of Information (ADOI) approach provides several modeling representations that are also extensions of UML notations for user and application system tasks [13]. It is useful for architecture designers to decide application design tactics with UCD practices. However, the ADOI approach requires a preparation of a sizable analysis model. In addition, it does not prompt the visual checkpoints of user tasks with high level of feature operations, notwithstanding its approach of simple analysis methods of modeling. Moreover, without coordinated usability design activities, it tackles and resolves only limited aspects of the application system's usability.

In software engineering, extreme context of user-concerned methodologies are emerged with simple and small user experience design deliverables as well as the pattern library information. The Agile usability process considers multiple design perspectives as a Central Design Record (CDR), which assimilates with the text-editorial requirements management tool [14]. However, the CDR does not include specific regulations of visual representations for rapid feedbacks. In addition, it does not explicitly implicate technical constraints in the business process automation. This means that it has different representations of user tasks and application features. On the other hand, the guidelines of usability functionalities assist application developers to determine whether and how usability features apply to the application sub-domains [15]. It leads to improve usability of the final application to be implemented by application developers. However, it is hard to achieve common understanding between UX designers and application developers about usability of user and system tasks. Moreover, the guidelines of usability functionalities include a checklist of usability risks and competences, which is not a checklist of feature completion for user tasks on a UI page [16].

III. Considerations

To overcome the limitations of the related works, we have developed a common modeling technique for the requirements analysis in Agile-UCD. Our approach provides visual checkpoints of user task completion, and uses UI design patterns in the perspective of software architecture in specifying requirements.

Having formulated representations yielding the standardized requirements analysis, this paper approaches a visual notation of agile modeling that decomposes of a user task into the application system entities on a UI page. This enacts the synchronous records of UI-based application features on a user task. Additionally, it applies UI design patterns of architecture design level into user tasks, which establishes the entry criteria of the usability functionalities for several usability concerned tests.

With discussing an approach of model-based mediation, which bridges the gaps in Agile-UCD between user tasks by UX designers and application features by application developers. Therefore, the approach of this paper can take advantage of:

- Enhancement of communication and collaboration, both internal and towards external stakeholders
- Domination of development works easily able to break down user and application system tasks
- Flexibility of application developers to be encouraged to take initiative and not to be obliged to follow UX designers' outcomes
- Improvement and creativity of usability risks, and management fostering competitiveness of usability criteria in fast and early stages.

IV. Approach

The approach in this paper is mainly to bridge user tasks and application features. This provides assurance to all stakeholders and commonly identifies user tasks and relative application system tasks [17]. Then, it specifies essential task flows of interactive applications. Therefore, it is possible for the AUS to successfully translate user tasks and relative application features into UML construction. Essential use cases are the natural language scenarios from user tasks and application system tasks on a UI page. Therefore, the requirements analysis method identifies several usability properties and patterns, which are applied into formalized application features of the control and boundary conditions. When an approach is named as a usability-pattern-based requirement-analysis method, it is a conjoint analysis coupling between: a visual notation model analysis bridging user tasks and relative application system entities; an abstract usability pattern analysis eliciting and applying architecturally high level of UI design patterns occurs [18].

In detail, the proposed usability-pattern-based requirement-analysis method has been divided into two analysis activities and related sub-analysis activities. The first one is the Contextual Link Model Analysis, which is a visual notation modeling technique decomposing and linking UI-based user tasks and relative system entities. To identify and structure user tasks and application system tasks, it has two sub-analysis activities. First, the User Tasks Analysis distinguishes user tasks from UI stretches. The second one is the Internal System Analysis that classifies application features separately from user tasks on the User Task Analysis. The second analysis activity is the UI Usability Pattern Analysis that is an elicitation of usability objects (i.e. properties and patterns) by the usability framework [19]. The UI Usability Pattern Analysis then applies usability objects into the Contextual Link Model Analysis. This also has two sub-analysis activities, the first of which is the Internal System Analysis. It is an activity confirming whether identified application features are able to be weaved and are serviceable or not. Then, the second sub-analysis is the Usability Framework Analysis, which links architecturally sensitive usability patterns into the application feature logics and outward aspects.

A. Contextual Link Model Analysis

The Contextual Link Model Analysis is a formalized visual communication modeling method. It intermediates development tasks between UX designers and application developers to capture significant requirement elements. It provides a specification of interactive facilities to be used by users while invoking core functionality of the application features. Moreover, it has the effect of a feasibility driver of the evaluation in design phase of the Agile-UCD. The Contextual Link Model Analysis displays several requirement differences between user tasks and application features. Therefore, it makes minimum measurable application features from context of use on user experiences. The Contextual Link Model Analysis identifies and structures core operations of the users and interactions with relative functionalities, including interaction styles and techniques. This can be pervasively used as the model-driven requirements specification inquiry that creates and simplifies elicited requirements for UX designers and application developers.

1) Principles of the Contextual Link Model Analysis:

The proposed Contextual Link Model Analysis is a new extended UML model to accommodate encompassment of the information, dialogue, and presentation dimensions shown in Figure 1. While maintaining the desired separations of requirements concerns, it clearly maps the prerequisite steps of the conceptual architecture design modeling in the interactive application developments.

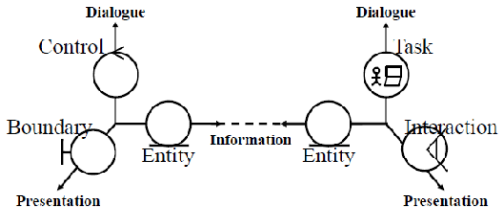

Figure 1. A dimension of the Contextual Link Model Analysis

With reference to a given UML analysis framework as well as introduction of two additional dimensions – the presentation and dialogue dimensions – by the Wisdom [12], the Contextual Link Model Analysis redefines classes and associations of user tasks and relative application features. The Contextual Link Model Analysis detaches the <<Interaction>> and <<Boundary>> of user interfaces from user and system tasks. And then, it regulates sequences of user task levels as a <<Task>> and system task levels as the <<Control>>. In addition, the <<Entity>> is shared among both the user interface specific dimensions and the internal application dimensions. This <<Entity>> class postpones domain adaptation to other models at lower levels of abstraction. Next, the Contextual Link model has two associations – <<Communicate>> and <<Subscribe>> - among interfering classes so that they denote directions of interaction between directly relative classes. While two associations are used with whole classes, the << Subscribe >> has a one-way route. On the other hand, the << Communicate >> has a two-way route. Each class and notation has been presented in Table 1.

TABLE I. CLASS AND ASSOCIATION STEREOTYPES OF THE CONTEXTUAL LINK MODEL ANALYSIS

Class	Notation	Description
Boundary		Used to model the interaction between the application system and external ones (including human), which corresponds with the high level of usability patterns
Control		Used to represent the coordination, sequencing, and control of system objects, which corresponds with the high level of usability properties
Entity		Used to show abstract level of feature classes of the logical domain behavior reflecting perdurable user tasks
Task		Used to represent the structure of dialogue for a user with a meaningful set of actions reflecting a user goal achievement on a user interface
Interaction		Used to model the space within a user interface of an application system corresponding to user experiences
Communicate	———	Used to denote direction of interaction between classes. The direction of the Communication is to be one- or two-way
Subscribe	---------	Used to denote direction of interaction between source class (subscriber) and target class (publisher); subscriber is notified when a particular event occurs in objects of the publisher. The direction of the Subscribe is the one way

2) Workflows of the Contextual Link Model Analysis:

As the first part of the usability-pattern-based requirement-analysis method, the activity diagram for the analysis workflow of the Contextual Link Model Analysis is illustrated in Figure 2. The topmost flow is the User Task Analysis that structures the interaction model in terms of task and interaction space classes. It is how classes are organized to support the user interface of the application system. It is divided into two concurrent flows that support the dialogue and presentation dimension of user and application system tasks. The bottommost flow reflects the Internal System Analysis and how analysis classes are organized to realize the different essential use cases with user stories. It facilitates information dimension of application tasks into the application feature entities.

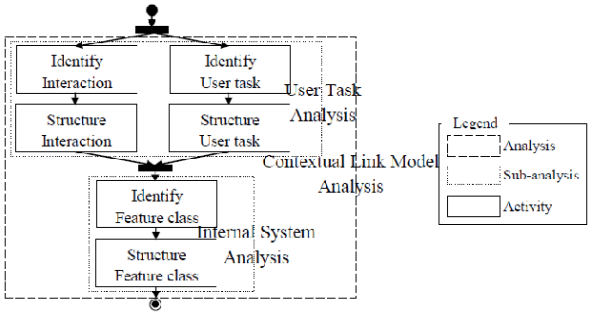

Figure 2. The activity diagram of the Contextual Link Model Analysis

The User Task Analysis activity mainly concerns the external conceptual design of an application, which is responsible for the user interface supporting users' envisioned tasks. There are two sub-activities in this Internal System Analysis: identify and structure interaction classes - <<Task>> and <<Interaction>>. The <<Task>> class stereotype is the element of the dialogue component. And, the <<Interaction>> class stereotype is the element of the presentation component. To identify both <<Task>> and <<Interaction>> classes, the essential task flow of the User Task Analysis is referred from user stories based essential use cases. The <<Task>> has usually mapped to the task steps in the task flows. The major concern for user tasks identification is to distinguish a set of tasks that support the essential task flows while ensuring robustness and reuse of the user interfaces. Regarding the presentation component, identifying and structuring the <<Interaction>> is a settlement between a user interaction and a user task of the transitions. This produces several versions of the usability concerning the UI design and acceptance test conditions with usability of quality attributes.

The Internal System Analysis activity facilitates the identifying and structuring of the functional requirements in a mode of the <<Entity>> classes. It advances application developers in understanding and management of the implementation works of the application system. It focuses on analyzing abstractions of domain concepts captured in the requirements workflow with postponing handling quality attributes. As with the User Task Analysis companion activities, the Internal System Analysis has two sub-activities: identification and structuring of general <<Entity>> analysis classes. At first, the application system tasks and objects are extracted with a light-weight version of the CRC card and relative methods. The next sub-activity follows to structure and distribute the identified <<Entity>> classes into analysis stereotypes giving responsibilities.

B. UI Usability Pattern Analysis

The UI Usability Pattern Analysis is a way to capture solutions to common usability problems in a specific context. It aims to provide ease of understanding an interface and accomplishing application system works with ensuring high usability performances for all stakeholders. The UI design patterns are good solutions to standard user interface design problems over effectiveness of usability. And they ensure verified implementation strategies of efficiency of usability. Therefore, UI design patterns have become a method for the AUS to use as a product and process management technique.

1) Principles of the UI Usability Pattern Analysis:

The proposed UI Usability Pattern Analysis identifies usability patterns of application system logics and relative outward aspects in the usability framework. Therefore, it applies usability properties and patterns into the Contextual Link model of the Internal System Analysis. The UI Usability Pattern Analysis refers to the usability framework of usability properties and patterns, as in Figure 3 [19]. The usability property is the concept of linking architecturally sensitive usability patterns to requirements considered to have a direct influence on application feature logics. In addition, the usability pattern is a technique that can be applied to the design of the architecture of an application system in order to address a need identified by a usability property.

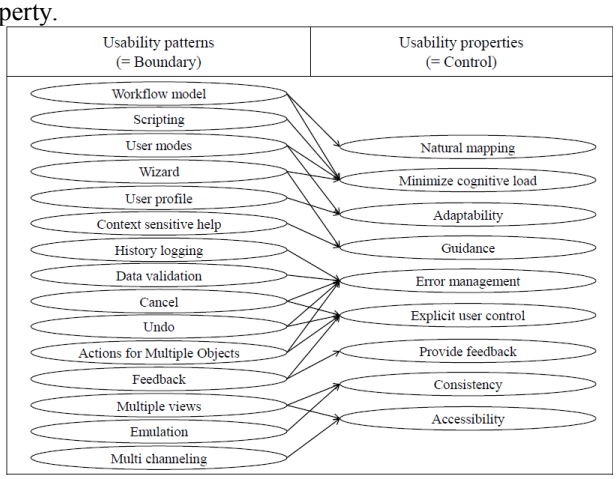

Figure 3. The UI patterns in the UI Usability Pattern Analysis

The UI Usability Pattern Analysis specifies usability factors to guarantee high quality of usability promoted activities for both UX designers and application developers. Furthermore, as the probated manner for UX designers and application developers, it does step forward usability risks management in early stages, and mitigates usability risks against limits of usability of functionalities.

2) Workflows of the UI Usability Pattern Analysis:

As a second part of the usability-pattern-based requirement-analysis method, the activity diagram for the analysis workflow of the UI Usability Pattern Analysis is illustrated in Figure 5. The topmost flow is the Internal System Analysis that checks pre-defined user task entities for whether they can operate application system tasks. It is also divided into two activities that re-identify and re-structure the application feature entities. Then, the bottommost flow is allowed to reflect usability concerned factors (usability properties and patterns). By identifying pertinent usability properties, the Usability Framework Analysis applies usability concerned UI design patterns into outcomes of the Internal System Analysis. By complying with the usability framework, the Usability Framework Analysis determines usability properties and then applies relative usability patterns.

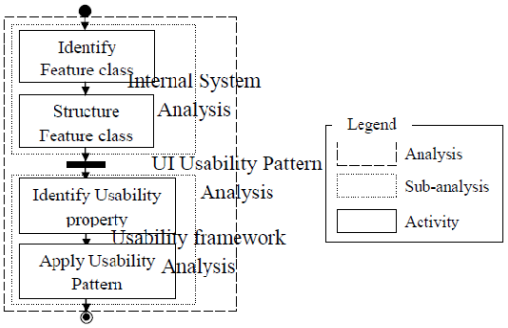

Figure 4. The activity diagram of UI Usability Pattern Analysis

The Contextual Link Model Analysis of notations in the requirement analysis model represents the high level of dialogue and presentation classes. Therefore, the UI Usability Pattern Analysis can be mapped into the analysis model stereotypes, i.e. <<Control>> and <<Boundary>>. As the Internal System Analysis activities are completed, several application feature entities are validated or newly extracted from the second application system tasks analysis. Therefore, the AUS ultimately identifies the <<Control>> and <<Boundary>> classes of attributes. These are intuitively mapped from the context of the application domain and the context of use.

After achieving identification of architecture levels of usability properties and patterns from the usability framework, the UI Usability Pattern Analysis applies a usability property and pattern into the <<Control>> and <<Boundary>> class from the Contextual Link Model Analysis. At this time, the usability property is supposed to be confirmed with the Contextual Link Model Analysis of notations. This is because the usability property is able to have multiplex concepts that are able to set several alternative usability patterns. The AUS has completed the confirmatory reaction of usability property into the <<Control>> classes. Then, the UI Usability Pattern Analysis is finalized by transforming the <<Boundary>> classes into the usability patterns on.

V. EXAMPLE OF THE APPROACH

As an example of the proposed usability-pattern-based requirement-analysis method, a simple mobile hotel reservation application is illustrated [21]. Even though the domain of this example is for the mobile application, it is very similar with the Web application of the hotel reservation. Therefore, an essential use case of hotel reservation application – MakeReservation – is explored in Table II.

TABLE II. A GENERAL ESSENTIAL USE CASE OF THE MOBILE HOTEL RESERVATION APPLICATION

Step	Description
1	The customer requests to check a hotel room booked available in near.
2	The system searches a hotel and shows a feedback of hotel reservation available.
3	The customer selects a room type of hotel, and enters start and end dates to stay.
4	The system informs availability for the room type on the dates specified.
5	If an appropriate room is available, the system computes the deposit charges, and prompts the customer to confirm the reservation.
6	The customer confirms to book a hotel.
7	The system requests the customer's credit card payment and personal details.
8	The customer submits the customer's credit card payment details and personal details.
9	The system creates a new reservation customer in the reservation database, assigning a unique reservation number.
10	The customer shows the reservation details including a reservation number.

According to the workflows of the approach method, identifications of user tasks and application system tasks of objects per a UI page are preceded in Figure 5. At this time, user tasks in an essential use case are supposed to be identified by a UI page. In the domain of a modern mobile application, a task is defined as a single task that can be stated in a single concise question or statement on a UI page [21]. As a composition of 5 statements in general, a task is clearly matched into the 5 classes of the Contextual Link Model: Boundary–Control–Entity–Task–Interaction [22].

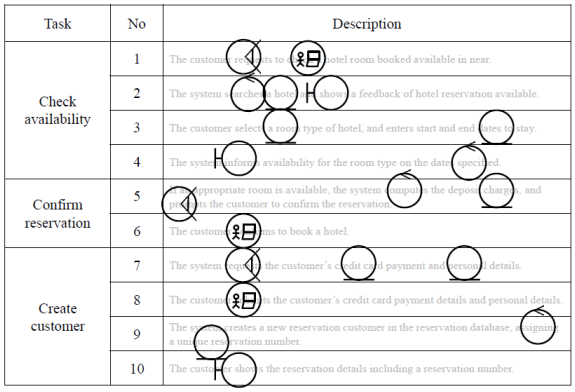

Figure 5. Contextual Link Model Analysis (Part I) for a mobile hotel reservation application

Next, organization (i.e. structure) of identified results of the user and application system tasks is followed by interconnecting each neighboring class with relative associations – <<Communicate>> or <<Subscribe>>. The results are in Figure 6.

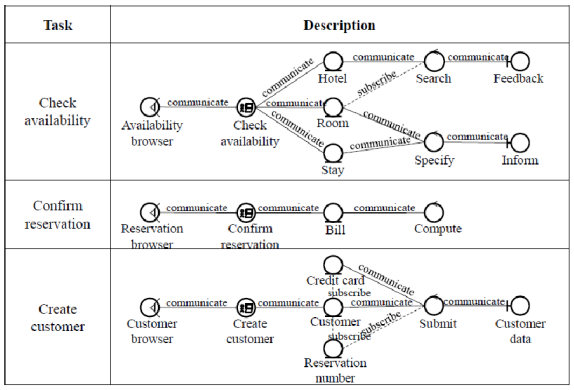

Figure 6. Contextual Link Model Analysis (Part II) for a mobile hotel reservation application

After completing a set of structure actions in the Contextual Link Model Analysis, the Internal System Analysis in UI Usability Pattern Analysis is executed. It validates the application system entities whether they are able to be developed in proper manner or not. Having neutralized between user and application system tasks, the <<Entity>> classes are verified as to their feasibilities at the Contextual Link Model Analysis as well as validated by the UI Usability Pattern Analysis. Then, labeling of <<Control>> and <<Boundary>> classes is performed, being considered application system tasks.

The Usability Framework Analysis is the final step. The AUS searches the usability properties in terms of context of use of the <<Control>> class, which results in the usability pattern with mapping of the <<Boundary>> class in Figure 7. As we already mentioned, a usability property has multiplex concepts that are able to set several alternative usability patterns. Therefore, the AUS has completed the confirmatory readjustment of usability property. Then, the UI Usability Pattern Analysis is finished by transforming the <<Boundary>> classes into the usability patterns.

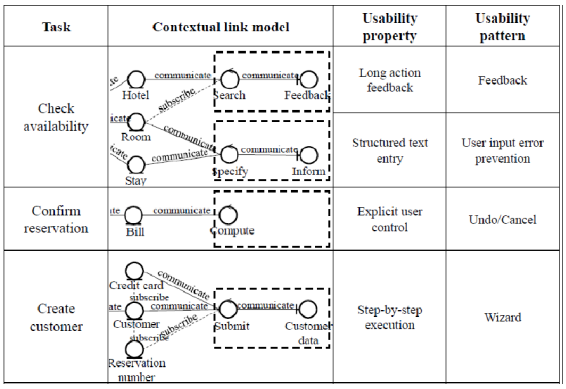

Figure 7. UI Usability Pattern Analysis for a mobile hotel reservation application

VI. EVALUATION

We used 10 senior level AUSs for this evaluation. 5 specialists are native to the application development track; on the other hand, the other 5 are native to the UX design track. They have, on average, experienced more than 4.40 years of implementation practices as well as 4.10 years of UX design. Furthermore, the examinees have, on average, experienced Agile-UCD projects about 2.80 times.

Meanwhile, the example throughout this evaluation is based on the simple problem of the essential use case of a mobile auction application illustrated in Table III [23]. Moreover, the essential use case is the Bid that is a real life situation where modern mobile applications were developed to support cooperative and interactive works between user tasks and application features.

TABLE III. AN ESSENTIAL USE CASE OF A MOBILE AUCTION APPLICATION

Step	Description
1	The bidder selects the bid option.
2	The system displays current bid and requests bidder to enter new bid.
3	The bidder enters bid.
4	The bid amount is higher than the current amount.
5	The bidder becomes the current highest bidder.
6	The bid amount becomes the highest bid amount.
7	The system updates and displays the bid.
8	The bidder enters bid.
9	The bid amount is not higher than the current amount.
10	The bid falls.
11	The system alerts bidder of amount being too low.

After the experiment of the mobile auction application case study, a questionnaire of the evaluation is used for identifying experimenters' practical tendencies with a Likert-scale. We defined five-point of Likert-scale for Agile practices, and each point has different numeric scores: severe Agile issue (-4), major Agile issue (-2), minor Agile issue (0), no Agile issue (2), and highly Agile (4). The five-point of Likert-scale for UCD practices has same numeric scores as Agile practices' one. The questions are mostly concerned with the issues of the AUS goals and activities from references and interviews. The questions are derived from the Agile Modeling principles [24] and usability of functionality process criteria [25], which aligns the extent of advantages in our approach; mentioned above section III. CONSIDERATION. Among nine questions for the evaluation, five of them, Question 1 through Question 5, are to learn the agility behaviors based on Agile methods. And From Question 6 to Question 9, four of the questions are to involve in the usability behavior of the UCD practices:

- Question 1: Do other AUSs understand the contents in it? – The usability-pattern-based requirement-analysis method promotes active stakeholder participation by being displayed publicly. It also assumes common awareness of the meaning.
- Question 2: Is it as simple a representation as the requirements specification artifacts? – The usability-pattern-based requirement-analysis method applies

the right artifacts to be used as the simplest tool of the requirements analysis. Furthermore, it depicts simple expressions to ease communication.

- Question 3: Does it conform to UML standards on the Agile Modeling? – The usability-pattern-based requirement-analysis method is needed to work for all stakeholders. It defines the notation and semantics for common object-oriented models
- Question 4: Do UX designers and application developers both understand the contents? – The usability-pattern-based requirement-analysis method creates a set of communication modes to develop a common vision between UX designers and application developers. Then, it promotes the test-driven usability and implementation evaluations.
- Question 5: Is it possible to contain unnecessary factors excluding user concerns? – The usability-pattern-based requirement-analysis method boosts the decision to synchronize requirements metaphors from user concerns. Then, it discards and formalizes the essential contract application features.
- Question 6: Does it define the user tasks on a UI page as a basic unit? – The usability-pattern-based requirement-analysis method is intuitive in terms of visualization of use, thereby having a low entry barrier for novices. It also easily explains user task goals on a UI page.
- Question 7: Does it define application operation objects in terms of user tasks? – The usability-pattern-based requirement-analysis method simplifies the application system operation sequences by executing user tasks. It covers somewhat complex sequences of the application system in a simple fashion.
- Question 8: Are the user tasks easily applied into usability patterns? – The usability-pattern-based requirement-analysis method supports various ways of making user tasks executions. Furthermore, it provides guidelines of assured usability configurations.
- Question 9: Is it possible to switchover the UI design of usability patterns for application features? – The usability-pattern-based requirement-analysis method makes the application features reflect usability manipulation styles and user needs well. It promotes the usability concerns into the functional support of user tasks.

Next, concerning problems of the AUS and resolutions of them, an evaluation validates modeling efficiencies, i.e. modeling completion time and compliances of modeling behaviors. In other words, this evaluation metric has been composed of four sub-category measures: the efficiency of modeling completion time; the efficiency of agile modeling; the efficiency in bridging user tasks and application features; and the efficiency of usability pattern.

A. *Modeling Time – Model Efficiency I*

The Modeling Time (i.e. Model Efficiency I; ME-I) is a verification of modeling completion time. As a use case is generally of 5 user tasks [26], a mobile application commonly has less than 10 use cases [27]. At this time, as a modeling completion in Agile software developments takes normally 2.5 hours [28]. Therefore, in Agile-UCD, a task is to be rapidly analyzed and modeled within 3 minutes. And, the Modeling Time of experiment from ten AUSs results are presented in Figure 8.

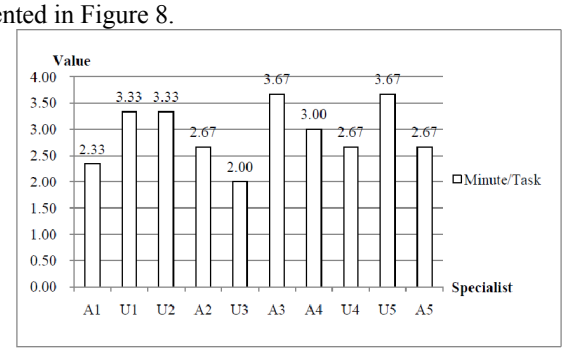

Figure 8. Evaluation results of the Model Efficiency I

All AUSs have successfully completed the tasks of the usability-pattern-based requirement-analysis method. They have specified correct user tasks and system tasks. They have completed the usability-pattern-based requirement-analysis method works at 2.90 minutes for a task. As compared with the general modeling time in Agile Modeling – 3 minutes, the evaluation result of the ME-I indicates that the time behavior of this requirement analysis model is more efficient. In the end, the proposed requirement analysis model has satisfied the efficiency of its time behavior for the AUSs to practically use it.

However, the evaluation results indicate the fact that several designer-native AUSs (i.e. U1, U2, and U5) are a little bit behind the standard value of modeling time compared to developer-native AUSs' one. The designer-native AUSs felt awkward with the model notations (and their actions) of the extended UML 2.0. Nevertheless, the clumsiness of the actions is expected to be reduced through training and orientation. As one of the designer-native AUSs, the U3 AUS has been examined two times for this evaluation at an interval of two weeks. At this time, the second examination result of modeling completion time (i.e. 2.00 Minute/Task) had been reduced by 45 % compared to the first examination result (i.e. 3.67 Minute/Task).

B. *Compliance of Agile Modeling – Model Efficiency II*

The Compliance of Agile Modeling (i.e. Model Efficiency II; ME-II) is a validation of modeling on agility criteria. It resolves the issue of poorly shared semantic interconnections between UX designers and application developers. Considering the context of the problem, it has a strategy of enhancement of communication and collaboration, both for internal and towards external stakeholders. Then, when reflecting on the strategy of the problem, the ME-II is supposed to be related to several Agile

Modeling practices - Active stakeholder participation, Displaying models publicly, Applying the right artifacts, Depicting models simply, Collective ownership, Modeling with others, and Considering testability. Next, these practices are applied into the criteria of the ME-II – Communicative, Simple, Standard, and Collaborative – with Question 1 through 4. Finally, the examinee AUS has measured how much the proposed approach enhances communication and collaboration for both UX designers and application developers as a simple standard representation.

Ten Specialists have voted on the level of the agility of the usability-pattern-based requirement-analysis method. Considering the meaning and impact of 4 criteria for the ME-II, eight AUSs, four AUSs, six AUSs, and nine AUSs have graded no Agile issue and highly Agile for Communicative, Simple, Standard, and Collaborative criteria. With analyzing each agility criteria of the ME-II from the average Likert-scale scores in detail, Communicative and Collaborative criteria have been satisfied with the Liker-scale point of no Agile issue (i.e. 2.0) However, Simple and Standard criteria have not penetrated the no Agile issues value in Figure 9.

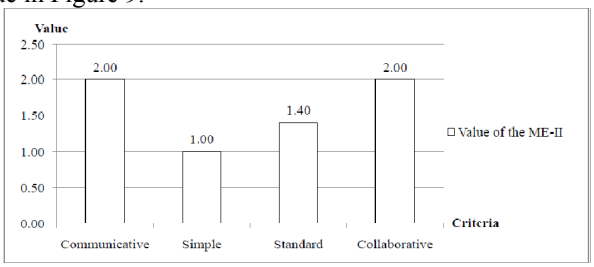

Figure 9. Evaluation results of the Model efficiency II

Several developer-native AUSs (i.e. A1, A3, and A4) had struggled with the new UML modeling and consolidation works for making the artifact. Moreover, they have considered this approach itself to be one of the additional requirements of engineering. They suggest the usability-pattern-based requirement-analysis method to be developed as a tool with simple representations while keeping its functionality. Regarding the Standard criteria, some AUSs have been confused about several legacy stereotype notations. They have considered the legacy UML notations to be used for whole application tasks, including user tasks and system tasks. In this situation, they thought the newly extended representation and their roles induce few difficulties for modeling works. However, after being given an explanation of this approach method, they have denied that the usability-pattern-based requirement-analysis method violates the UML standards.

C. *Compliance of Transforming Tasks to Features – Model Efficiency III*

The Compliance of Transforming Tasks to Features (i.e. Model Efficiency III; ME-III) is a validation of modeling on agility and usability criteria. It resolves a problem of hardness of driving prioritized application features from user tasks. With reference to the essentials of the problem, it has a set of strategies: domination of controlling development tasks easily able to break down user and system tasks; flexibility in the sense that application developers are encouraged to take initiative and are not obliged to follow UX designers' outcomes. By reflecting strategies of the problem, the ME-III is supposed to be related with two Agile Modeling practices: Discard temporary models and Formalize contract models. Furthermore, it covers UCD core practices – Visual support of task goals and Operation sequences for executing a task. These agility- and usability-concerned practices are then applied onto the criteria of the ME-III – Contractible, Task intuitive, and Sequential, which is based on Question 5 through 7. Eventually, the examinee AUSs have inspected how much the usability-pattern-based requirement-analysis method dominates allocation of development tasks easily able to break down user and application system tasks. And, they has also validated how our approach encourages the application developer to take initiative in UX design concerns, and not feel obliged to follow UX designers' outcomes.

Ten Specialists were also polled regarding the level of the agility and usability of the usability-pattern-based requirement-analysis method. For 3 criteria of the ME-III, seven AUSs have graded no Agile issue and highly Agile for Contractible criteria. And nine of them have given marks of no UCD issue and highly UCD for Task intuitive criteria. Besides, seven AUSs have valuated no UCD issue and highly UCD for Sequential criteria. Under analysis of the average Likert-scale point for Agile-agility and UCD-usability by poll results, the usability-pattern-based requirement-analysis method has been graded up with no Agile or UCD issue. And then, this result indicates that. usability-pattern-based requirement-analysis method has a pertinent feasibility of the contract-based modeling. Therefore, it has resolved the difficulty of driving prioritized application features from user tasks. By being specifically validated concerning the agility and usability in the ME-III, Contractible and Sequential criteria have minor issues in the Agile or UCD, as shown in Figure 10.

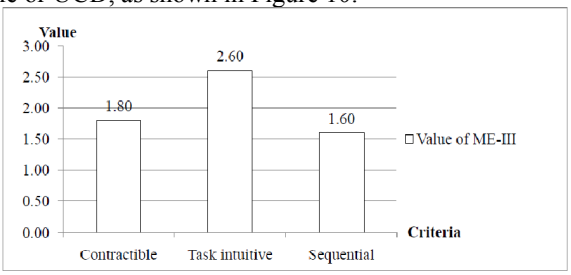

Figure 10. Evaluation results of the Model efficiency III

Most of the AUSs find the break down actions easier compared to essential use cases to specify and prioritize user tasks and application features on a UI page. However, several developer-native AUSs are dispositive of tasks decomposition too detailed to elicit detail design essentials, which resulted in useless outcomes for software architecture design perspectives, as well as being out of compliance the developments contract. For the Sequential criteria, some AUSs (i.e. A2 and A3) have complained about ontological system tasks that are not able to optimize the structure of

application features. One of them has decomposed user tasks into sub-tasks and relative operations. However, he has agreed to the fact that decomposition works are supposed only to be imposed by application developers.

D. Compliance of Usability Pattern – Model Efficiency IV

The Compliance of the Usability Pattern (i.e. Model Efficiency IV; ME-IV) is a validation of modeling on usability criteria. It settles a problem of high usability risks by limit of usability of functionality. Regarding the context of the problem, it has a strategy of improvement and creativity fostering competitiveness of usability criteria in fast and early stages. With the consideration of the problems, the ME-IV has coherence with UCD core practices, i.e. support of efficient interaction and Functional support of user needs. Furthermore, usability concerned practices are employed with the criteria of the ME-IV (Pattern executive and Transformative). The developer-native or design-native AUSs have carried out the evaluation of how much improvisatorially affects competitiveness of usability criteria in turbulent application domains.

Ten AUSs have completed a poll of usability degrees concerning the usability-pattern-based requirement-analysis method. With a set of criteria of the ME-IV, seven AUSs have graded no UCD issue and highly UCD for Pattern executive criteria. And eight of them have given marks of no UCD issue and highly UCD for Transformative criteria. They have evaluated this approach has various ways of promotion that make user tasks execute with well-defined design tactics. Moreover, it provides guidelines of assured usability configurations that make application features reflected in usability manipulation styles.

By analyzing each agility criteria of the ME-IV in detail, Pattern executive criteria is highly over the Liker-scale point of no UCD issue (i.e. 2.0). However, Transformative criteria has the minor issue of UCD has been noted down as the 1.80 – an average of Likert-scale point – in Figure 11.

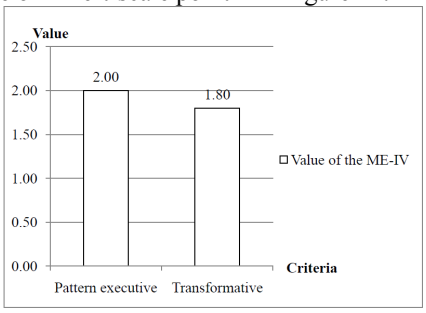

Figure 11. Evaluation results of the Model efficiency IV

Some of the developer-native AUSs are quite satisfied with the usability pattern execution, which reduces the deep insights of high level of architecture design as well as decreasing the usability jeopardy in early phase. Despite those advantages, several developer-native AUSs experience difficulty in elicitation of usability properties from the outcomes of the Internal System Analysis in Contextual Link Model Analysis. On the other hand, designer-native AUSs have disclosed a UI design of usability pattern difficulty to be applied into the detailed user tasks and system tasks. However, as they are using universal usability patterns, they have expected this approach to be used as an indication of task information and relative usability configurations.

VII. CONCLUSION

The modern interactive software systems have been closely operated with the demands of the user and rapid launch of a product rather than the complicated development technology [28][29]. This characteristic requires Agile methods and UCD practices to be combined into the Agile-UCD. At this time, the proposed usability-pattern-based requirement-analysis method becomes the key representation of the rich application software developments. It shows all the parts of responsibilities of the AUS to maintain coherent chunks of UX design and agile development activities. Due to its inclusion of Contextual Link Model Analysis and UI Usability Pattern Analysis, our method presents a lightweight requirements specification method in Agile-UCD encompassing three aspects, as follows:

- Strengthen communication and collaboration between both UX designers and application developers in parallel tracks of application designs and developments
- Develop appropriate representations of workflows for user and application system tasks on a UI basis by defining a set of compliant modeling notations
- Consider workflow-oriented usability criteria on the object-oriented model to ensure usability of functionalities

Our approach is complementary to the Agile-UCD that enhances communication and collaboration for all stakeholders. It seamlessly rationalizes the chaotic modes of semantic interconnections between UX designers and application developers. In addition, it controls the requirements specification that prevents the detail software design. This affects the AUS by boosting more communicative and collaborative appliance for the requirements engineering utilization. By promoting a UI-based task driven modeling language, this paper ensures high agility development progresses with simple standardized representations. However, a few AUSs request up front model analysis in detail for dialogue and presentation dimensions. Moreover, other AUSs revealed some difficulty with the extended UML representations in doing modeling works.

Even though our approach has successfully resolved the three main problems of the Agile-UCD in requirement specification phase, there is a need for modeling training to seamlessly apply the approach into practices. In addition, even though this approach satisfies the visual transforming and expressing the user tasks into application features, some of the user tasks are hardly ever supposed to convert right application feature of entities. Therefore, the level of application feature entities is to be defined in coherence with user tasks as well as application system tasks. In addition, as we define the scope of the Agile-UCD in the requirements

specification, the Agile-UCD itself sets the next cycle of requirement engineering until the time of the project end. Therefore, quantitative verifications of pre-implementation and post-implementation in each cycle are asked this work around. Therefore, we plan to investigate to ensure the reliability of our method in Agile-UCD. Consequently, verifications and validations in post-implementation could be applied in real project practices.

VIII. REFERENCES

[1] R. Beauregard, A. Younkin, P. Corriveau, R. Doherty, and E. Salskov, "Assessing the Quality of User Experience", Intel Technology Journal Vol.11 Issue 1, 2007

[2] Z. Hussain et al., "Agile User-Centered Design Applied to a Mobile Multimedia Streaming Application", USAB 2008, LNCS 5298, pp. 313-330, 2008.

[3] P. Abrahamsson et al., "Agile software development methods - Review and Analysis", ESPOO 2002, VTT Electronics, ISBN 951-38-6010-8, 2002

[4] Z. Hussain, et al., "Investigating Agile User-Centered Design in Practice: A Grounded Theory Perspective", USAB2009, LNCS 5889 pp.279-289, 2009

[5] L. Miller and D. Sy, "Agile User Experience SIG", CHI2009, ACM 978-1-60558-247-4/09/04, pp. 2751-2754, 2009

[6] D. Fox, J. Sillito, and F. Maurer, "Agile Methods and User-Centered Design: How These Two Methodologies Are Being Successfully Integrated In Industry", Agile 2008 Conference, IEEE 978-0-7695-3321-6/08, 2008

[7] J. Dickinson and D. Kumana, "Agile and UCD: Building the right thing, the right way", 2009

[8] S. W. Ambler, "Tailoring Usability into Agile Software Development Projects", Maturing Usability, Springer, London, 2008

[9] J. Dickinson and D. Kumana, "Getting Agile with User-Centered Design: Create software users can't live without", StickyMinds, 2008

[10] M. Budwing, S. Jeong, and K. Kelkar, "When User Experience Met Agile: A Case Study", CHI2009, ACM, 2009

[11] M. Federoff and C. Courage, "Successful User Experience in an Agile Enterprise Environment", Salesforce.com, HCII2009, 2009

[12] J. Belenguer et al., "HCI Designers and Engineers: It is possible to work together?", IFIP2003 - Closing the Gap: Software Engineering and Human-Computer Interaction, pp.20-25, 2003

[13] D. N. J. Nunes, "Object Modeling for User-Centered Development and User Interface Design: The Wisdom Approach", Universidade da Madeira, 2001

[14] Q. Hua et al., "A UCD Method for Modeling Software Architecture", IPSI, 2002

[15] J. C. Lee and D. S. McCrickard, "Towards Extreme(ly) Usable Software: Exploring Tensions Between Usability and Agile Software Development" AGILE 2007, ISBN:0-7695-2872-4, pp. 59-71, 2007

[16] N. Juristo et al., "Guidelines for Eliciting Usability Functionalities", IEEE Transactions on Software Engineering, Vol. 33, No. 11, 2007

[17] Ahmed Seffah , Taleb Mohamed , Halima Habieb-Mammar , Alain Abran, Reconciling usability and interactive system architecture using patterns, Journal of Systems and Software, v.81 n.11, p.1845-1852, November, 2008

[18] E. Folmer, M. Welie, and J. Bosch, "Bridging patterns: An approach to bridge gaps between SE and HCI", Information and Software Technology 48 (2006) pp.69-89, 2005

[19] E. Folmer, J. Gurp, and J. Bosch, "Software Architecture Analysis of Usability", University of Groningen, 2005

[20] P. Ng, "Hunting for Use Case Scenarios", The Rational Edge, Rational Software 2003, 2003

[21] Microsoft Corporation, "Microsoft Inductive User Interface Guidelines", Windows User Interface Technical Articles, 2001

[22] Max Möllers et al., "Towards Systematic Usability Verification", CHI2009, ACM 978-1-60558-247-4/09/04, pp. 4645-4650, 2009

[23] H. Motsi, "Wireless Applications for Virtual Communities of Practice", Tshwane University of Technology, 2004

[24] S. W. Ambler, "Lessons in Agility from Internet-Based Development", IEEE Software (2002) 0740-7459/02, 2002

[25] J. Heo, "A framework for evaluating the usability of mobile phones based on multi-level, hierarchical model of usability factors", Interacting with Computers 21 (2009), pp. 263-275, 2009

[26] Mobile Marketing Association, "Mobile Search Use Cases", Version87, 2007

[27] I. K. Ibrahim, "Agent-Based Mobile Auctions: The Flea Market Scenario", International Research Conference on Innovations in Information Technology (IIT2004), UAE university, pp. 89-98, 2004

[28] S. W. Ambler, "Agile modeling: Effective Practices for eXtreme Programming and the Unified Process", ISBN: 0-471-20282-7, Wiley & Sons, Ltd, 2003

[29] Z. Hussain, A. Holzinger, and W. Slany, "Current State of Agile User-Centered Design: A Survey", USAB2009, LNCS5889, Springer 2009