3D digital prototyping and usability assessment of user interfaces based on User Interface Description Languages -Lessons learned from UsiXML and XAML-

Satoshi Kanai[1]

[1]Division of Systems Science and Informatics, Graduate School of Information Science and Technology, Hokkaido University
Kita-14, Nishi-9, Kita-ku, 060-0914, Sapporo (Japan)
kanai@ssi.ist.hokudai.ac.jp

ABSTRACT

Usability-conscious design while shortening the lead time has been strongly required in the manufactures of information appliances in order to enhance their market competitiveness. Prototyping and user-test of their user interfaces at early development stage are the most effective method to fulfill the requirements. However, fabricating the physical prototype costs much, and they become available only in late stage of the design. To solve the problem, advanced tools for UI prototyping were proposed and developed where a UI-operable 3D digital prototype can be fabricated in a less-expensive way based on user interface description languages (UIDLs), and where the user test and usability evaluation can be performed in more systematic and efficient way than in the physical prototype. In the tools, two conventional UIDLs were adopted; UsiXML and XAML. And their specifications were expanded to support not only declarative description of static structures and dynamic behaviors of the UI, but 3D geometric model of appliance housings and physical UI objects placed on them. Case studies of the automated user tests, usability assessments and UI redesigns utilizing our developed tools are shown.

Keywords

Prototyping, usability-conscious design, UIDL, UsiXML, XAML, user test, information appliances.

General Terms

Design, Experimentation, Human Factors, Verification.

ACM Classification Keywords

D2.2 [**Software Engineering**]: Design Tools and Techniques – *Modules and interfaces; user interfaces*. D2.m [**Software Engineering**]: Miscellaneous – Rapid Prototyping; reusable software. H5.2 [**Information interfaces and presentation**]: User Interfaces – *Ergonomics; Graphical user interfaces (GUI); Prototyping*. J.6 [**Computer Applications**]: Computer-aided Engineering.

INTRODUCTION

With stiffer global market competition of information appliances, usability-conscious design while shortening the lead time have been required in the manufactures. The manufactures are placing a premium on increasing efficiency and consciousness of usability in the UI software development of their appliances. The "usability" is defined as "the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use".

There are several methods of evaluating usability [6]. Among the methods, the "user test" is recognized as the most objective and effective one where end users are directly involved in the evaluation. In the user-test, designers make end users operate a working "prototype" of the appliance, observe the user's operational situation, and closely investigate ergonomic issues of the UI design.

However in the current UI software development for the prototype, its specifications are still described by written documents, and the software is implemented based on the documents. This makes the prototype implementation process inefficient if a redesign of the UI is needed after the user test.

Moreover, the "physical" prototypes of the appliances are

Figure 1. User test and a physical prototype

mostly used in the user-tests. However, fabricating the physical prototypes costs much. For example, as shown in Figure 1, a working physical prototype of a compact digital camera costs a few thousand dollars which is around one hundred times more expensive than the final product. These prototypes also become available only in late stage of the design.

Results of the user-test must be analyzed manually by the usability engineers, and a long turnaround time is needed before finding major usability problems. If problems of the UI design appear at this time, it is often too late for changes within their development schedule.

To solve the problems, digital prototyping of the UI has been introduced in the user-test. A digital prototype is software substitute where its UI functions can work almost in the same way as those in the physical prototype while it can be built in much inexpensive way.

RELATED WORKS

2D and 3D digital prototypes

So far, as shown in Figure 2, both 2D and 3D digital prototypes have been proposed and used for simulation and user-test of UI operations in the information appliances.

Commercial digital prototyping tools have been already available such as [17, 21, 22] as shown in Figure 2-(a). And those for conceptual UI design were also studied in [14, 15].

However, since 2D digital prototypes could only be built in these tools and its UI simulation were only limited to 2D and lacked reality, the user performance obtained from these prototypes were not necessarily the same as those of physical prototypes. Former studies including ours [11, 20] showed that operation time and missed operation patterns in a 2D digital prototype were very different from those of the physical prototype and serious usability's problems were overlooked in 2D case.

On the other hand, "3D" digital prototypes allows users to manipulate UI elements placed on 3D housing models of the appliances and to perform more realistic UI simulation than 2D ones. In our former study [11], the missed operation patterns in a 3D digital prototype were also highly correlative to those of the real product.

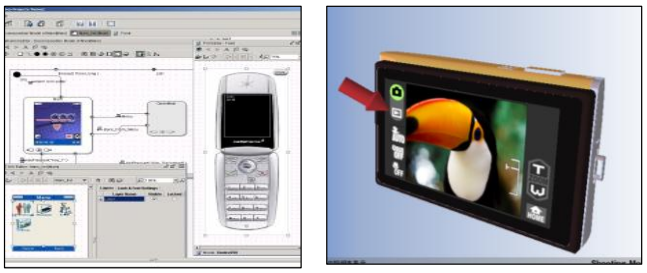


Figure 2. 2D and 3D digital prototypes

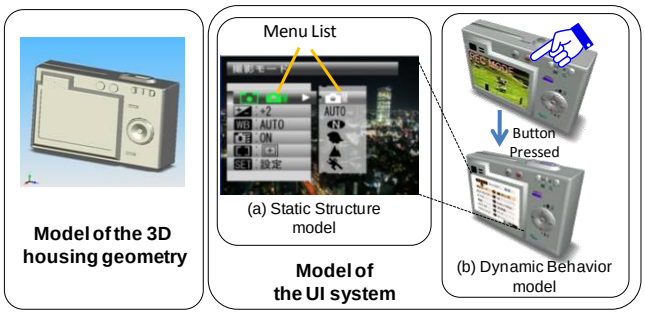


Figure 3. Modeling constituents of the UI operable 3D digital prototype

Unfortunately, there have been few dedicated tools of 3D digital prototyping for information appliances [8, 11, 20]. In [8], they added a logging function to a VRML player and applied it to the user test of mobile phones. In [20], they developed a tool for 3D augmented prototyping for some hand held information appliances with state-based UI simulation.

Issues of current 3D digital prototypes

To assure reliability of the user test results to some extent in the early design stage, 3D digital prototypes are more likely to be suitable for testing and evaluating the logics of the UI, and for clarifying the weaknesses and what needs improvement in the UI design.

As shown in Figure 3, the modeling of UI operable 3D digital prototypes consists of two parts; the model of the UI system and of the 3D housing geometry. Moreover the model of the UI system is divided into the static structure model and the dynamic behavior model. The static structure model of UI describes graphical properties of individual 2D components displayed on the UI screen such as menu-list, button, list-box, slider and image component, and also describes containers for the component layout such as window and tabbed dialog-box. While the dynamic behavior model of UI describes how graphical properties of the 2D components of the UI change in interaction and enables us to simulate the state change of the UI part in the appliance.

Conventional UI operable 3D digital prototypes were built and run using the Web3D authoring tools and their players [8, 25, 26, 27]. However, the following technical issues remain when we use the Web3D as the UI operable 3D digital prototype for user test and usability assessment;

(1) Lack of the static structure model of the UI

The static structure of the 2D components displayed on a UI screen such as menu list or icon placements cannot be directly modeled in the Web3D formats. So a huge number of digital image files representing snapshots of the UI screen must be built using the 2D drawing or painting tools before the UI simulation and the user test. This pre-

paratory work makes the simulation turn-around very slow.

(2) Lack of the dynamic behavior model of the UI

The Web3D formats usually provide script-based control function which enables 3D kinematic animations, change of the graphical properties of 3D objects and texture mapping etc. But the function cannot simulate the dynamic behaviors of the 2D components displayed inside the UI screen. The We3D formats do not also provide any declarative dynamic behavior model of the UI system which is based on state-based or event-based formalisms. These formalisms of the UI fit to the screen transition diagrams in early UI design stage [13, 4], and are indispensable to the specification. The lack of the declarative dynamic behavior model forces UI designers to code the behavior using programming language. But the designers are not necessarily programming professionals, and the task makes the cost of UI simulation and user testing expensive.

(3) Lack of user test and usability assessment functions

The Web3D formats do not provide any functions of user test execution and usability assessment based on the operational logs. Doing these works manually on the digital prototype makes the usability assessment time-consuming and the assessment results inaccurate.

To solve these issues, the dedicated functions of modeling the static structure of the UI-screens, of modeling the event-based or state-based dynamic behavior of the interaction, and of supporting the computer-aided test execution and the usability assessment must be added to the traditional Web3D authoring tools and players.

To achieve them, our research group has been developing advanced tools for UI prototyping were proposed and developed where a UI-operable 3D digital prototype can be fabricated in a less-expensive way based on user interface description languages (UIDLs), and where the user test and usability evaluation can be performed in more systematic and efficient way than in the physical prototype. In the tools, two conventional UIDLs were adopted for UI specification and implementation in the final development stage; UsiXML [24, 28] and XAML [18, 30]. And their specifications were expanded to support not only declarative description of static structures and dynamic behaviors of the UI, but 3D geometric model of appliance housings and physical UI objects placed on them.

In the following sections, the functions and features of the two developed tools each of which is respectively based on UsiXML or XAML are introduced. Case studies of the automated user tests, usability assessments and UI redesigns utilizing our developed tools when applied to UI prototyping of digital cameras on the market are shown.

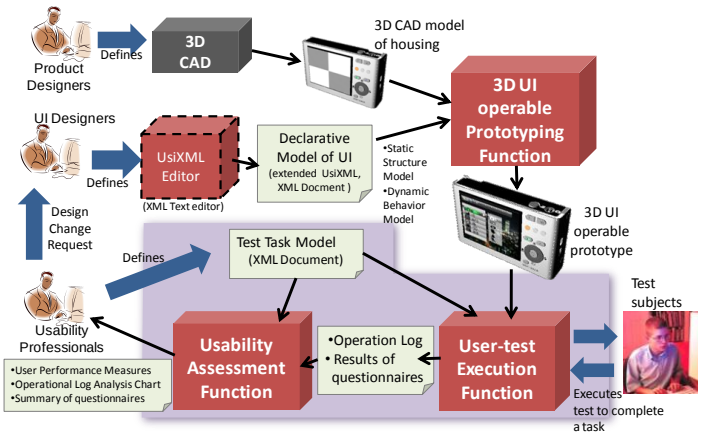


Figure 4. An overview of the UsiXML-based prototyping, user-test and usability assessment tools

UsiXML-BASED 3D DIGITAL PROTOTYPING AND USABILITY ASESSMENT

An Overveiw

As the first approach, the 3D digital prototyping and usability assessment tools based on Usi-XML were developed by our group[11]. Figure 4 shows the functional overview of the tools. The features of the tools are summarized as follows;

(1) The model-based specification of UsiXML, which is one of the XML-compliant UI description languages, was extended to enable the UI designer to specify the declarative model not only of the logical UI elements displayed on the screen but of the physical UI elements such as buttons and dials placed on appliance's housings.

(2) 3D UI prototyping and simulation functions were developed where the extended UsiXML was combined with the 3D CAD models of the housings and where the UI interaction were simulated based on the declarative model of the UI behavior described by the UsiXML.

(3) The automated usability assessment functions were developed which in such a way that they were tightly connected to the declarative model of the UI and to the simulation functions.

(4) An example of the usability assessment and the UI redesign using the 3D digital prototype of a digital camera using our tool was shown the effectiveness and reliability of our proposed tool.

UsiXML and its extensions

UsiXML

Several XML-based user interface mark-up languages have been recently proposed to make UI prototyping of PC applications reduced and structured: UIML[23], XUL[29], and UsiXML[24]. Each of them specifies models for defining the UI and allows us to describe the UI model in declar-

ative terms. They enable the UI designers or UI programmers to specify what elements are to be shown in the UI and how should they behave in XML documents. This concept becomes an advantage for defining 3D digital prototypes of handy information appliances from the following standpoints:

(1) The static structure of the 2D component objects displayed on the UI screen is explicitly and declaratively modeled and described by the XML document. The snapshot of the UI screen can be automatically drawn in the simulation based on the static structure model if we realize the real-time rendering function for the model. It can eliminate the preparatory work of the UI simulation, and makes its turn-around efficient.

(2) The dynamic behavior of the UI interaction has to be described by script or ordinary programming language in most the of UI mark-up languages (UIML, XUL and XAML). However, in the UsiXML, the behavior can also be explicitly described as an event-based model. The model can eliminate the coding of UI dynamic behavior simulation if an execution function of the behavior model in the simulator of the 3D digital prototype is realized.

(3) The user test and the usability assessment can be automated if the static structure and the dynamic behavior models of the 3D digital prototype are reused for analyzing the property of the subject's operations in the usability assessment functions. It can make the cycle of prototyping-testing-redesigning very efficient.

Therefore, we introduced UsiXML to our study, because it can describe the dynamic behavior model of the UI in a declarative way and model the UI at a different level of abstraction. UsiXML was originally proposed by Vanderdonckt et al [16, 28]. It aims at expressing a UI built with various modalities of interaction working independently. The UI development methodology of UsiXML is based on the concept of MDA (Model Driven Architecture).

Issues of the UsiXML from the aspect of 3D digital prototypes.

The concept and specification of UsiXML is advanced in UI prototyping, but it has still the following issues when we directly use it for developing UI operable 3D digital prototypes and for the usability assessment;

(1) As shown in Figure 5-(a), the CUI of UsiXML specifies the static structure model of 2D UI component objects displayed on the UI screen, but the current CUI only specifies the static structure model for WIMP (Windows-Icons-Menus-Pointers)-type GUI. In UsiXML, there are no specifications for the physical UI elements such as buttons, dials, lumps and dials placed on the information appliance's housing shown in Figure 5-(b), which are essential for modeling the UI operable 3D digital prototypes.

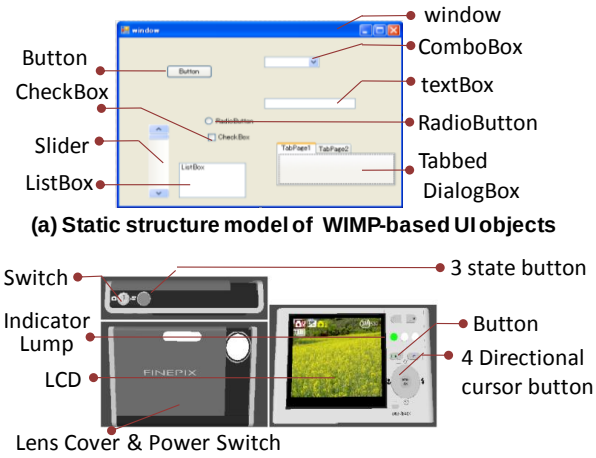


Figure 5. Types of UI component objects modeled in the static structure model

(2) Many experimental automated tools have been developed for UsiXML. However, there is no 3D UI prototyping and simulation tool available to UsiXML at present. The event-based dynamic behavior model is specified in the CUI, but it has not been reported yet how the dynamic behavior of the UI is described concretely, nor how the model of the CUI should be implemented on a particular Web3D format.

(3) UsiXML has been originally developed for UI prototyping, but at present there is no specification and no supporting tool concerning user testing and usability assessments which utilize the UI prototype. Therefore, we cannot incorporate the functions of user testing and usability assessment into UI prototyping based on UsiXML.

Extensions of the CUI model of UsiXML

The current specifications of the CUI in UsiXML mainly consist of the static structure model of the objects displayed on the UI screen and the dynamic behavior model of the interactions of the UI. The static structure model further consists of the UI object model and the object container model. The UI object model expresses the individual GUI component object displayed on the UI screen such as buttons, listboxes, image components, etc., while the object container model does the whole-part relationships among the multiple GUI objects such as a window and a tabbed dialog box. The dynamic behavior model consists of the event-based rewriting rules of the UI screen in interaction and of the procedures and variables to refer to the internal data of the device.

In this research, we extended this static-structure-model part of UsiXML so as to fit it to the UI operable 3D digital prototyping. Figure 6 indicates the UML class diagram and its example which expresses a part of the original CUI model structure in the UsiXML. In the structure, *gra-*

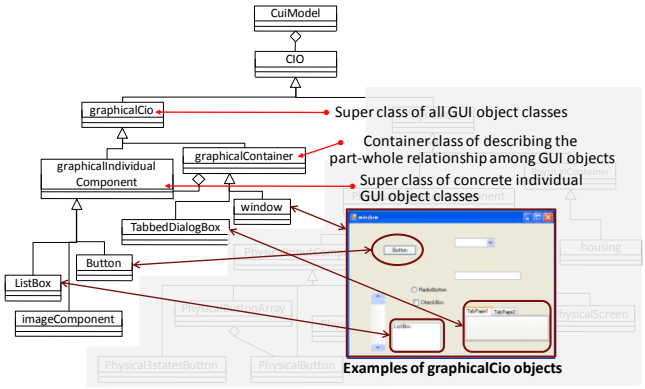


Figure 6. UML class diagram and its example of the original CUI model of UsiXML

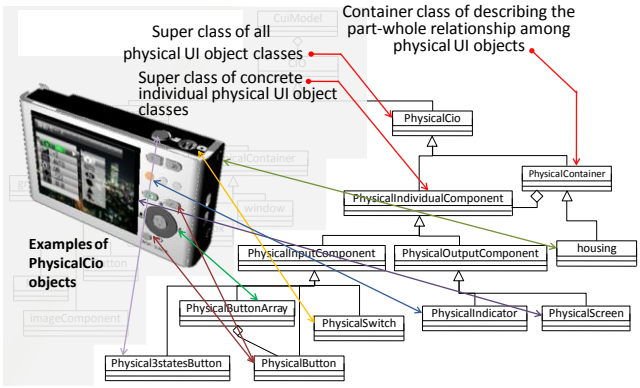


Figure 7. UML class diagram and its example of our extended CUI model of UsiXML

phicalCio is the super class of all GUI object classes in the model, and *graphicalContainer* is the container class for the GUI objects. The concrete classes of the GUI objects and the object containers of UI are defined as a subclass of these two classes.

On the other hand, Figure 7 indicates the class diagram and its example of our extension of the CUI model. We extended the class structure of the CUI model to express the physical UI objects such as physical buttons, dials and lumps placed on the 3D geometric model of the appliance' housing. First, we added a new class *physicalCio* to the same class hierarchy level as one of the *graphicalCio* class. Then we further created two new classes of *PhysicalIndividualComponent* and *PhysicalContainer* as subclasses of the *graphicalCio*. The *Physical-IndividualComponent* class expresses the one for modeling each the physical UI object, and the *PhysicalContainer* class does the physical housing of the appliances which play a role of the virtual container in aggregating the physical UI objects. Moreover, as the subclasses of *PhysicalIndividualComponent*, we added a *PhysicalButton* class and *PhysicalScreen* class to the subclasses of *PhysicalIndividualComponent* in order to express concrete buttons and LCDs placed on the housing. Figure 6 shows the correspondence between the physical UI objects in a digital camera and the classes describing them.

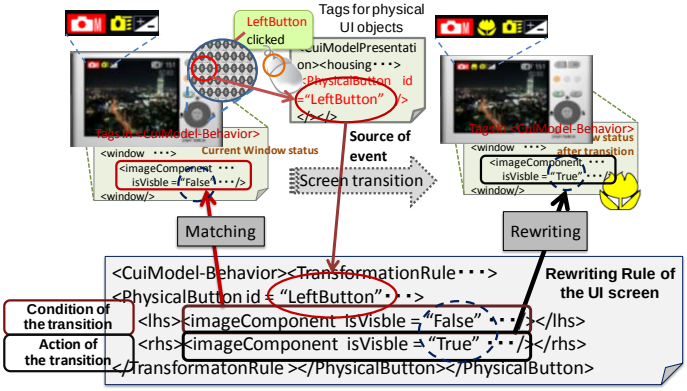


Figure 8. UI screen transition rule described in XML document of the extended UsiXML

Design of the XML document structure of the extended CUI model

The current version of the UsiXML does not specify the explicit XML encoding rule of the CUI model. Therefore, we specified a tag structure of the XML document of our extended CUI model independently. This tag structure is imported to the 3D UI operable prototyping functions and is used for the 3D UI simulation. Figure 8 shows an example of the tag structure in XML document and their presentations in the UI screen image.

The structure consists of a <*CuiModel-Presentation*> tag and a <*CuiModel-Behaviour*> tag. The former represents our extended static structure model of the CUI which express the objects displayed in the UI screen, while the latter does the dynamic behavior model which corresponds to the UI screen transition. And concrete CUI objects are described inside these two tags in our XML document.

To describe the UI screen transition, we set up a <*TransformationRule*> tag inside the <*CuiModel-Behaviour*> tag which describes the general graph rewriting rule mechanism defined in the original UsiXML specification. As shown in Figure 8, in the <*CuiModel-Behaviour*> tag, we put the pair of a condition tag <*lhs*> and an action tag <*rhs*> together by each tag corresponding to the subclass of *PhysicalInputComponent* class. The condition tag expresses a condition where the screen transition occurs because of an event coming from the physical UI objects. And the action tag expresses the state of the UI screen after the screen transition occurs.

In the user-test execution function, if an event occurs in an object belonging to any subclass of the *PhysicalInputComponent* class, the function tries to find a <*TransformationRule*> tag which has the same tag id as the source of the event from all tags inside the <*CuiModel-Behavior*> tag. And if the current tag and its attribute values in the <*Cui-*

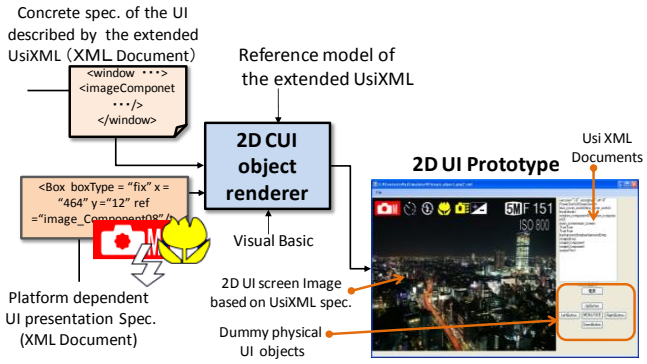


Figure 9. Functions of 2D CUI object renderer

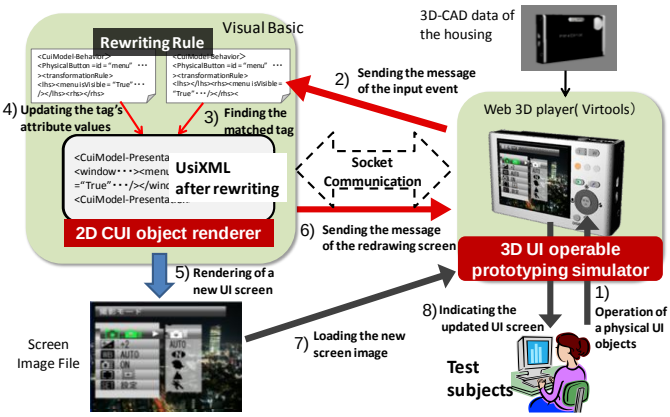


Figure 10. 3D UI simulation function based on UsiXML

Model-Presentation> tag are exactly identical to the ones written in the <*lhs*> tag in the <*TransformationRule*> tag, then these attribute values are overwritten as the one in the <*rhs*> tag. This overwriting mechanism implements the UI screen transition on the 3D UI operable prototype based on UsiXML.

3D UI operable prototyping function

2D CUI object renderer

We developed a 3D UI operable prototyping function where the extended UsiXML is combined with the 3D CAD models of the housings and the UI interaction were simulated based on the dynamic-behavior model of the UsiXML. The 3D UI operable prototyping function consists of the 2D CUI object renderer and the 3D UI operable simulator which is a remodeling of Web 3D player (Virtools [27]).

The 2D CUI object renderer is a VisualBasic application developed by ourselves. Figure 9 shows the function of the renderer. The renderer interprets the XML document of the extended UsiXML and accepts another XML document which defines the platform dependent UI presentation specification such as the concrete position of each GUI object and object containers on the UI screen. It renders the 2 dimensional UI screen image on the fly according to the UI screen transition rule described in the XML document. It also renders the dummy physical UI objects on the same 2D UI screen. So the renderer also enables the UI designer to do the 2D UI software prototyping when the renderer is used alone. The 2D CUI object renderer is executed during the 3D UI simulation cooperating with the Web 3D player to provide the main function for the UI simulation.

The 2D CUI object renderer enables UI designers to eliminate their preparatory works of generating a huge number of snapshot images of the UI screen, and makes the turnaround time of the 3D UI simulation short.

3D UI operable prototype simulator

Figure 10 shows the 3D UI operable prototyping function. The function consists of the 2D CUI renderer and the 3D UI operable prototype simulator which is a commercial Web 3D player (Virtools[27]). We remodeled the players so that the Web 3D player runs with the CUI renderer simultaneously, and they exchange events via socket communication during the 3D UI simulation for the user-test.

In the preliminary step of the 3D UI simulation, 3D CAD data of the housing is imported to the Web 3D player from 3D-CAD systems (CATIA, Solidworks, etc.) in the format of 3D-XML[1]. The 3D-XML is a universal lightweight XML-based format for the sharing of 3D CAD models. The format includes the assembly and each part has its own unique part-name. In the model, a 3D object which is the source of an event or the destination of an action is modeled as a single part. A button, a switch knob, an LCD screen and an LED are typical examples of these objects.

In the Virtools player, a UI designer links an event and an action described in the <*lhs*> tag and the <*lhs*> tags in the <*TransformationRule*> tag in the dynamic behavior model of UsiXML to messages of the Virtools player. A message consists of a unique event-name, part-name and event-type. For an example, an event of "*button_1_pushed*" in the UsiXML model is tied to a message consisting of "*message-1*" (event-name), "*button-part-1*" (part-name) and "*on_clicked*" (button-type). Consequently this linking operation builds all logical links between the messages in the Virtools and events or actions in dynamic behavior model of UsiXML.

As shown in the processing sequence in the Figure 10, the 3D UI simulation is executed as the following procedure:

(1) The user manipulates the 3D housing model of the appliances and operates a physical UI object such as a 3D button placed on the housing model using the mouse on the player.

(2) The operation of the physical UI object is recognized as an event in the player and it sends a unique message of the event to the 2D UI object renderer via socket communication.

(3) The renderer analyses the incoming message to pick up the event, and tries to find a <*TransformationRule*> tag which has the same tag id as the source

of the event from all tags in the <*CuiModel-Behavior*> tag.

(4) If the current tag and its attribute values in the <*CuiModel-Presentation*> tag are identical to the ones written in the <*lhs*> tag in the <*TransformationRule*> tag, then these attribute values are updated to the new ones according to the content of the <*rhs*> tag.

(5) The renderer then redraws a new UI screen image after the screen transition in a scratch file according to the updated attribute values.

(6) The renderer sends another message of the screen redraw event to the player.

(7) The player loads the new screen image from the updated scratch file.

(8) The texture rendered on the face in the 3D housing model which corresponds to the UI screen changes to the new screen image.

By repeating the procedure, the 3D UI simulation on the 3D housing model which cooperates with the 2D UI simulation is realized in the prototyping function. The UI simulation rule is completely described in the XML document of UsiXML in declarative way, and the simulation execution is completely managed in the developed renderer.

User test execution function

Test task and task model

In the user test, a subject is asked to complete a set of tasks by operating the UI, and actual operations for the task are analyzed to evaluate the usability. In our tools, we designed a "test task model" and made a logical link between the task model and the dynamic behavior model of the UI to automate the usability assessment. Figure 11 shows the test task model. This task model is originally developed for the "state-based" dynamic behavior model of the UI screen [9].

A task consists of a start state, goal state and a list of task routes. And a task route consists of a list of checkpoints. A checkpoint is a state where a correct sequence of UI operation must pass. A start state and a goal state refer to the state in the UI behavior model.

Generally multiple correct operations of the UI exist in order to achieve a goal. Therefore multiple task routes can be allowed for one task in the model. Moreover, lower and upper bounds for the number of operations or an allowable elapsed time between every two neighboring checkpoints can be also defined. If the elapsed time of a subject's operation stays within these bounds, the operation is judged to be correct. In this way, usability professionals can adjust a range of correct operations in the task when determining the number of error operations and the task completion.

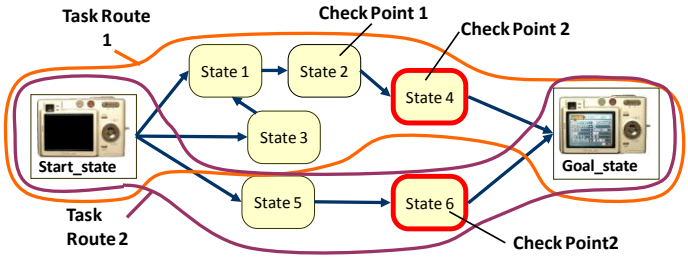


Figure 11. Test task model

State evaluation and operation logging

The dynamic behavior model of the UsiXML is expressed by a set of transformation rules which describe how the attribute values of the objects displayed in the UI screen have to be changed in response to the input event. Therefore it is the "event-based" dynamic behavior model, and the model does not have the explicit notion of "state" of the UI.

However, in the user test, the task model was originally designed for the "state-based" dynamic behavior model of the UI system [9]. And the notion of the state is indispensable for defining test task, recording user operation logs and identifying missed operation. Without the notion of the state, it is very difficult for the usability professionals to capture the situations of user operations.

To solve the problem, we added the state evaluation function in the user test execution function. In this function, a set of conditions which describes attribute values of a CUI object to be taken in a particular "state" are defined in an XML document in advance. And the function always evaluates whether the condition holds or not in the UI simulation of the user test. If a set of attribute values of the displayed CUI object is perfect match for the condition, then the function reports that the UI transits to the certain predefined state.

In some cases, there might be several different modes with a same set of attribute values in the behavior model. In this case, our state evaluation function cannot identify these modes as different states. Inserting an extra attribute for distinguishing one mode from others into the original set of attributes can solve this problem.

Using this mechanism, the user test execution system can recognize which state the UI is in during the simulation. We also developed the operation logging function based on the state evaluation function. The logging function records all subject's operations in the form of the combination of a time, a previous state, a next state and an input event coming from the user interaction. The function saves these logs in a XML document. We integrated the state evaluate function and the operation logging function with the user test execution function, and enabled the usability professionals to manage the user operation log for every task during the user test.

Test execution

Shown in an example of Figure 12, at the beginning of every new test session, the user test execution function indi-

Figure 12. 3D digital prototype under user-test

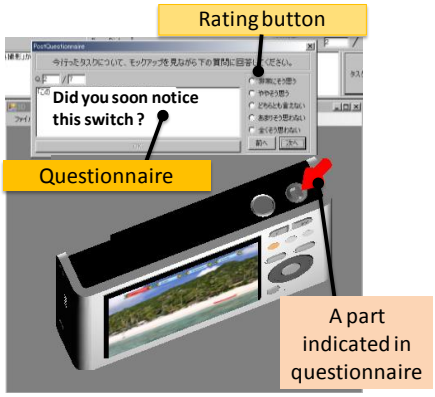


Figure 13. Digital questionnaire on the prototype

cates a goal of each task in the form of an imperative sentence. "Switch shooting mode from still to video" and "set the self-timer for 20sec" are the typical examples of the goal. The goal is indicated on the other window just above the 3D digital prototype.

The subject is asked to complete the task by operating the UI of the 3D digital prototype in the session. The subject manipulates the 3D prototype by rotating, translating and enlarging it and operates the physical UI objects on the prototype by clicking or dragging them with a mouse in the virtual space. If the physical UI objects, such as buttons, are located on a different surface that cannot be seen from the 2D UI screen such as LCD, the subject have to rotate the prototype so as to make these objects face him/her.

During the UI operation, the operation logging function records a sequence of state-transitions of the UI as a list of combinations of state and event together with the time stamps.

At the end of the every test session, the operation logging function compares the actual sequence of state-transitions with all pre-defined task routes, allowable number of operations and elapsed time between checkpoints. Then the logging function judges whether the subject's operation of the session for this task was correct or not, and identify which checkpoint state the subject made mistakes in his/her operation.

If the operation is judged to be wrong, a set of digital questionnaires are progressively popped up on another screen at the end of the test session. One questionnaire is displayed corresponding to one checkpoint state at which the subject made a mistake. A portion on the UI object in the 3D digital prototype related to each questionnaire is automatically pointed by the tool as shown in Figure 13.

The subject is asked to answer to each questionnaire by choosing his/her impression from five grades. For example, when the questionnaire is "Did you soon notice this button?", the rating is the one of "Strongly agree: 5", "Agree: 4", "Yes and No: 3", "Disagree: 2" and "Strongly disagree: 1". The subject answers this rating only by clicking one of the radio buttons placed on the questionnaire as shown in Figure 13. The rating is stored to clarify what needs improvement in the design in the usability assessment function. The detail of this digital questionnaire is explained in the next section.

Usability assessment function

User performance measures

The usability assessment function investigates the operation log data by comparing it with the test task and the dynamic behavior model of the UI. The function outputs the measures of usability assessment as a result of the analysis. The analysis can be automatically processed, and the function outputs measures of the user performances.

We adopted the following three measures based on three basic notions of usability (effectiveness, efficiency and satisfaction) defined in [7];

1) The number of user events inputted in each task and in each subject,

2) Elapsed time in each task and in each subject,

3) Personal task achievement ratio, and

4) Scores for SUS questionnaire [2].

Operational log analysis chart

An actual sequence of operations is compared with one of the correct task routes defined in the test task, and the result

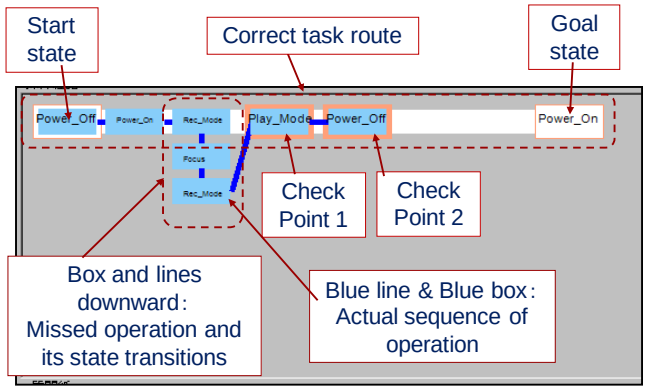


Figure 14. Operational log analysis chart

of the comparison is graphically displayed in the form of "operational log analysis chart" on the lower part of the screen.

Figure 14 shows the notation of this operational log analysis chart. Each rectangle shows a state, and a line between two rectangles does a transition between states. A left-most rectangle in the chart indicates a start state, and a right-most rectangle does a goal state. The upper most horizontal white straight line indicates transitions on a correct task route, and every rectangle with orange edges on this line except both ends corresponds to each checkpoint. While the blue rectangles and blue lines indicate actual operation sequence of the subject. If a subject does UI operations whose elapsed time or number of events between two neighboring check points exceeds the predefined bounds, the tool judges that a subject did a wrong operation on the UI, and draws additional blue rectangles and blue lines in downward direction corresponding to these wrong operations.

Therefore, as the depth of the chart becomes larger, the usability professionals can easily recognize at a glance which states the subject did more missed operation in the task.

Digital questionnaires

The operational log analysis chart can clarify particular states in the UI screen transition where many subjects made errors in the UI operations. However, the chart cannot provide enough information to enable UI designers to find the cause of missed operations and to redesign the UI in order to improve the user's performance.

To solve this problem, the digital questionnaire execution function has been developed to identify the causes of the missed operations. The structure of the digital questionnaires is built based on an extension of the cognitive walk-through method which is dedicated by the HCI (Human Computer Interaction) model.

To construct the digital questionnaires, we first introduced the extended HCI (Human-Computer-Interaction) model. The extended HCI model [3] is an extension of the CW method whose questionnaire is formulated based on an extended model of Norman's HCI[19] that explicitly distinguishes between object and action, and between perceiving and understanding. The questionnaires were originally used for the usability evaluation of Web sites. The questionnaires based on the extended HCI model are easier to understand for end users. A list of the questionnaires is shown in Table 1.

However, as shown in Table 1, the questionnaires based on the extended HCI model still have abstract expressions. We further make them more straightforward and concrete when using them in the user tests of information appliances so that the end users can understand them more easily as shown in Table 1. For example, a questionnaire of the extended HCI model which examines the perception of the object to be manipulated is expressed as "Will users be able

Table 1. Extended HIC model and the questionnaires

Cognition Process of Extended HCI Model	Items of questionnaires based on Extended HCI [Hori & Kato 2007]	Items of questionnaires used in our system
Formation of Intension of manipulation	Does the user try to accomplish the correct action ?	Did you easily understand what you should do for the appliance by reading the task ?
Perception of objects	Can the user perceive the object to be manipulated ?	Did you soon notice this [*input element name*]?
Interpretation of objects	Can the user understand that the perceived object is the correct object ?	Did you soon understand that you should operate this [*input element name*] ?
Perception of actions	Can the user perceive his/her actions of manipulation ?	Did you soon understand how you should operate this [*input element name*] ? (by pushing, sliding etc.)
Interpretation of actions	Can the user come up with actions of manipulation which should be applied to the object ?	N.A.
Execution of actions	Can the user certainly execute the correct action?	N.A.
Perception of the effect	Can the user notice the change of the state ?	Did you soon notice that the state of this [output element name]changed by your operation?
Interpretation of the results	Can the user understand what state the system is after the state change ?	Did you easily understand how the state of the appliance changed as a result of the operation by observing the state change of [*input element name*] ?
Evaluation of the results	Can the user understand that he/she advances toward the solution of the task by observing the system state ?	Did you soon understand whether your operation is correct or not by observing the state change of [*input element name*] ?

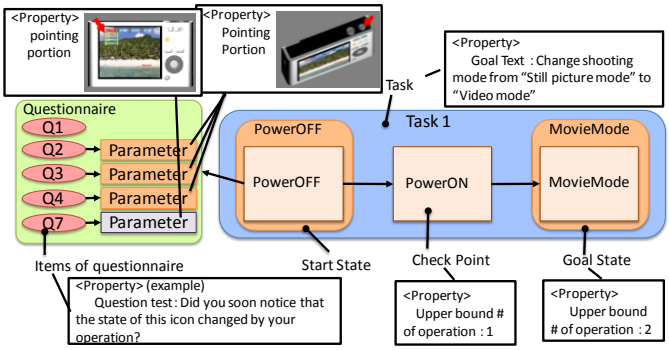


Figure 15. Association of the digital questionnaire with the test task model

to perceive the object to be manipulated?". We changed it to more plain one; "Did you soon notice this button?" when applying it to the test of a digital camera.

Moreover, we also implemented a function that automatically points to a 3D object which corresponds to "this button" or "here" on the 3D digital prototype when indicating a certain portion in the questionnaire as shown in Figure 14. This function enables end users involved in the test to understand each item of the questionnaire more easily, and also enables usability professionals to save a quite bit of manpower for constructing the questionnaire.

Figure 16. The appliances for the user test

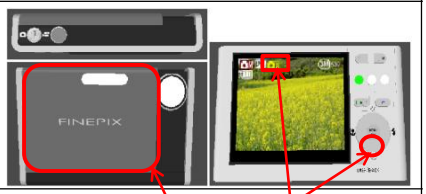
Task goal	Setting a self timer to10 sec from power-off state.
Minimum # of operation	2
Physical UI objects to be operated	
Objects indicated in Questionnaires	• Front lens cover • Downward cursor button • Self time icon displayed in LCD

Figure 17. The task of the user test

The proposed digital questionnaire enables many end users to take part in the cognitive walkthrough evaluation and to answer the questions by actually manipulating the 3D digital prototype whose UI can work as same as the final appliance does. This feature can greatly increase the reliability of the user test's results.

For defining the digital questionnaire, as shown in the Figure 15, the usability professionals assign one questionnaire to a check point in the test task model. The professional also specifies a particular portion of the UI screen image or the 3D model of the housing which should be indicated on the 3D prototype. The standard template of the questionnaire is predefined as a stencil with standard properties in the Visio, and the usability professionals can paste the stencil to the particular task, which is graphically displayed in the Visio, and input the sentence of a question in the property value [10].

A case study of usability assessment and redesign

Task Setting

A case study was done which consisted of the user test, the usability assessment and the UI redesign based on the results of the assessment. A compact digital camera (Fuji FinePix-Z1) on the market shown in Figure 16 was selected for the user test.

The goals of the case study were:

- to investigate whether the UI operable 3D digital prototype and our tool can clarify the weaknesses in the UI design where many subjects often make mistakes in their UI operation,
- to investigate whether the digital prototype and the tool can clarify why many subjects often make mistakes in the operation and what needs improvement in the UI design,
- to investigate how small the differences in the assessment results are between the digital prototype and the real product, and
- to evaluate how efficiently the redesign of the UI can be done using the UsiXML and our tool.

A UI operable 3D digital prototype of this camera was built as shown in Figure 12 and used for the assessment. For the digital prototype, we modeled the dynamic behavior model of the camera UI which has 34 states and 224 transitions including neighboring transitions of correct operation sequences of the task. A real product was simultaneously used for the assessment, and the result was compared with that of the 3D digital prototype.

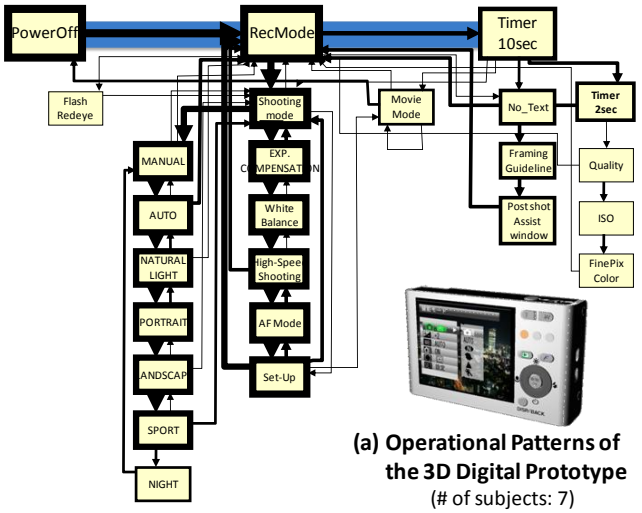
The prototype was operated by 14 subjects (male and female students of age 20-29) who had not used the same model. 7 subjects took part in the test using the UI operable 3D digital prototype, and the other 7 subjects used the real camera.

We defined the task of "Setting a self-timer to 10 seconds from power-off state" in the test which is one of the occasionally-used operations. In the task, as shown in Figure 17, first the user has to turn on the power switch by sliding the front lens cover, and then to switch the mode from the manual shooting to the self-timer setting with 10 seconds by pushing a downward cursor button once. If the camera reaches to this goal state (self-timer 10 seconds), a small circular white icon which symbolizes the self timer appears on the top of the LCD. The subject has to notice that this icon indicates the goal state and that he/she completed the task.

Analysis of operational patterns

To investigate the subjects' actual operational patterns both of the 3D digital prototype and the real product, actual sequences of operations including missed operations of each subject are put together. The sequences are schematically drawn as a "summarized operational log analysis chart".

This chart can be made by superimposing an operational log analysis chart for one subject shown in Figure 14 onto ones of the other subjects. In this summarized chart, the notation of correct and missed operations is the same as the one in the personal version described in the previous section. But the width of a directed path on the chart is proportional to the number of subjects who passed over the transi-



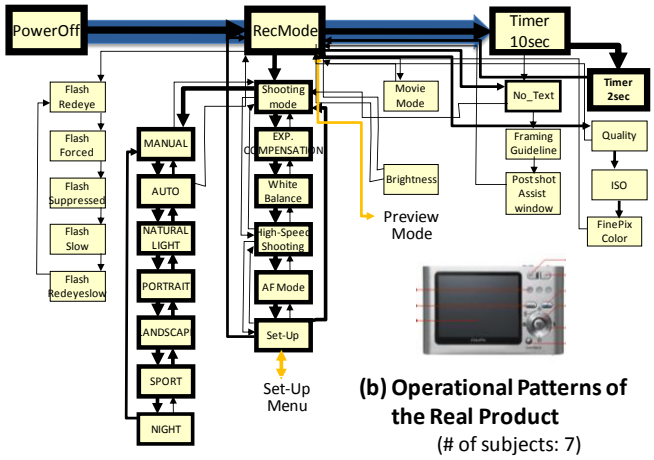


Figure 18. Comparison of the summarized operation log analysis charts of the user test

tion corresponding to the line. Therefore, a wider directed path indicates that more subjects passed over the routes of operation to complete the task.

Figure 18 shows the two summarized operational log analysis charts; the chart for the subjects who operated the 3D digital prototype (Figure 18(a)) and the other for the ones who did the real product (Figure 18(b)).

Both analysis charts show the clear facts that;

-at the "Rec_Mode" state, many users stepped into the wrong path aiming to the "Shooting mode" state instead of the correct path to the "Time_10sec" state,

-even at the "Timer_10sec" state which is a goal of the task, many users went past the state and continue operating to reach the "Time_2sec" state, and

-the pattern of missed operations of subjects who operated the 3D digital prototype is very similar to that of the real product.

From this comparison, the differences of the operational patterns were small between the UI operable 3D digital prototype and the real product. It was also shown that the 3D digital prototype could find the weakness in the UI design where many subjects often take missed operations similar to the those missed operations performed by the ones using the real product.

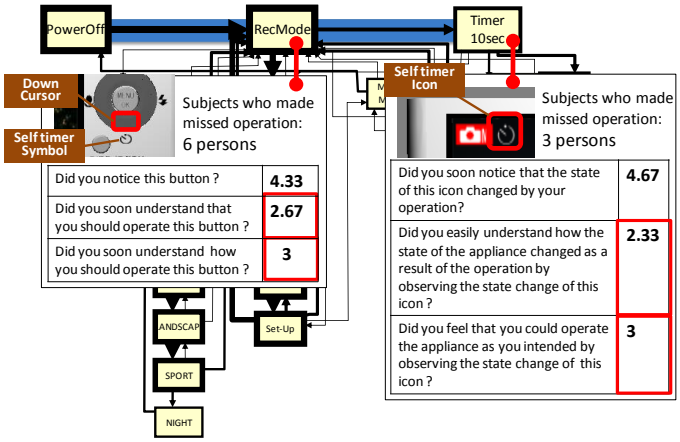


Figure 19. Average rating for the digital questionnaires at 2 states in question in the original UI design

Analysis of digital questionnaires

When reading only from the summarized operational log analysis chart, we could not discover the reasons why so many subjects made mistakes in those particular states and what needs improvement in the UI design. So we further analyzed the rating from the subjects in the digital questionnaires indicated on the 3D digital prototype.

Moreover, the ratings obtained in the digital prototype were compared with those in the real product. To the subjects who used the real product, the questionnaires were manually indicated to them, and the ratings were written by themselves.

The average ratings from the subjects for the digital questionnaires indicated at the "Rec_Mode" state and the "Timer_10 sec" state were shown in Figure 19. "Rec_Mode" state means that the camera is in the manual shooting mode, and "Timer_10 sec" state that it is in the self-timer setting mode with 10 seconds and is the goal state.

The ratings at the "Rec_Mode" state from the 3D digital prototype suggest that many subjects could recognize a down cursor button itself, but could not notice that they could move to the self-timer setting mode from the manual shooting mode by pushing this cursor button. Therefore, from the rating analysis, we found that the small symbol indicating the self-timer printed on the housing surface near the cursor button needs to be changed to a new one which can give us the self-timer function at a glance.

On the other hand, the ratings at the "Timer_10sec" state suggest that the subjects could notice the change of the system status caused by their operations, but could not understand what occurred in the camera and whether they correctly accomplished the task. This means that this white

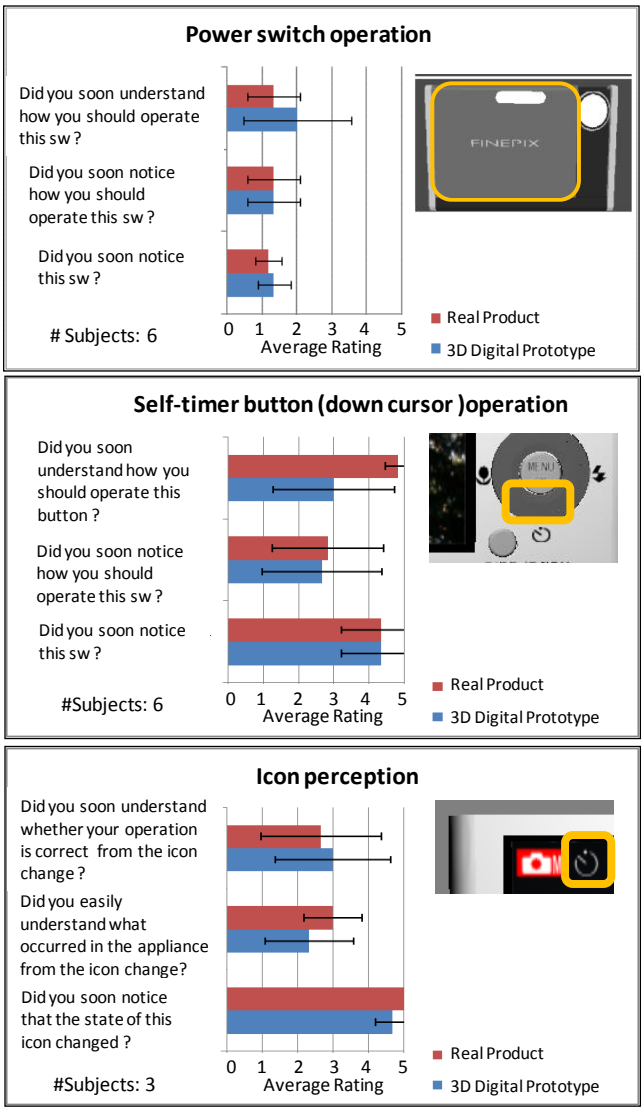


Figure 20. The differences in the ratings in three questionnaires between the 3D digital prototype and the real product

icon displayed on the LCD in Figure 19 could be noticed by many subjects, but did not enable them to notice that the self-timer settings had already been set to 10 seconds. Therefore, from the rating analysis, we finally found that the small timer icon indicated on the LCD in the original UI design need to be improved to the new on which can give us the setting value of the self-timer at a glance.

Figure 20 shows the difference in the ratings in three questionnaires between the subjects who used the UI operable 3D digital prototype and the ones who used the real product. There are strong correlations of the ratings between the digital prototype and the real product in all three cases. Therefore, the combination of the UI operable 3D digital prototype and the digital questionnaires enables the UI designers to reveal what needs improvements in the UI and

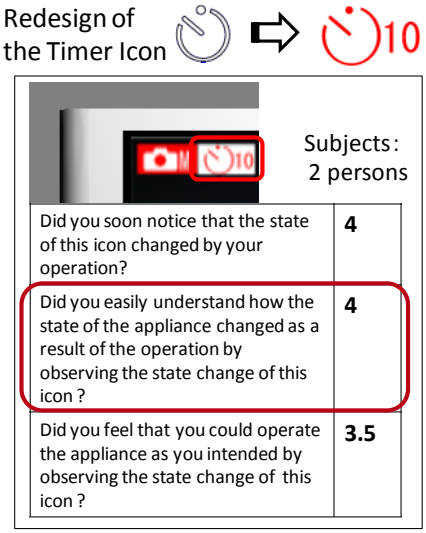


Figure 21. Improved average rating for the digital questionnaires at "Time 10sec" state in the modified UI design

how they should be improved as minutely as a real product does.

UI redesign based on the digital questionnaires

The rating analysis in the digital questionnaires revealed that there are two candidates which should be redesigned in the original UI design; a small symbol indicating the self-timer printed on the housing, and the small self-timer icon indicated on the LCD shown in Figure 19.

Based on the analysis, we evaluated how efficiently the re-design of the UI can be done using the UsiXML and our tool. In this study, only the second redesign candidate was implemented.

As shown in the Figure 21, we redesigned the shape and color of the timer icon to the new ones so that the background color becomes conspicuous and the timer setting value is explicitly drawn in the icon. An additional test was executed for the new four subjects who used the 3D-digital prototype with the redesigned icon. The result of the test showed that two of four subjects could complete the task without wrong operations. And the rest could also complete the task with small wrong operations. Moreover, the result of the ratings of the digital questionnaire of the redesigned icon indicates that more subjects could easily find that the self-timer settings had already been set to 10 sec.

In this redesign work, it only took 10 minutes to redraw the icon image and 1 minute to rewrite a small part of the tag contents in the XML document of the UsiXML.

From the whole results of the case study, we obtained the following conclusions;

- The summarized operational log analysis chart based on the 3D digital prototype enabled UI designers to

discover the weakness in the UI design where many subjects make mistakes, and also that the digital questionnaires enabled them to clarify what needs improvement and who they should be improved in the design.

- There was a strong correlation of the operational log analysis chart and the ratings in the questionnaire between the 3D digital prototype and the real product. Therefore, the UI operable 3D digital prototype could replace a real product or a physical prototype while keeping the ability to discover the usability problems of the UI logic.
- The UI operable 3D digital prototype based on the UsiXML and the automated usability assessment functions can complete the works of prototyping-test-redesign more efficient than the current manual based assessment can.

XAML-BASED 3D DIGITAL PROTOTYPING AND USABILITY ASESSMENT

Issues in UsiXMl-based prototyping

Usi-XML-based 3D prototyping and usability assessment tools in the previous sections enabled us to achieve the realistic simulation fidelity of the UI, to declaratively and explicitly describe the static structure and dynamitic behaviours and to execute the very efficient and systematic usability assessments.

However, in these tools, there were still the following technical issues to be improved in terms of prototyping;

(1) The UI simulation environments of 2D and 3D were not fully integrated. Structure of 2D UI components such as menus and icons could not be directly rendered in the UI simulation function. As a result, every UI screen had to be rendered on-the-fly as a texture-mapped image on the 3D prototype, and huge number of UI screen snapshots had to be rendered every time the UI screen changes. This causes the simulation execution very inefficient.

(2) Due to the texture-mapping and the limited image resolution, the appearance of the UI screen in the 3D became much degraded when the 3D model is zoomed up. It might let the test subjects feel unmotivated for the simulation-based user test.

(3) An expensive commercial Web3D player (Virtools) was needed for the 3D UI simulation. This forced the manufacturers to make an additional investment for the prototype and hindered widespread use of the proposed technology.

(4) So far, there is no sophisticated or commercial visual authoring tool or editor, which can help UI designers easily build and modify the extended UsiXML models in a visual way.

(5) The tools did not support simulation of touch sensitive interface which becomes very common in recent appliance UIs such as i-Pod and digital cameras.

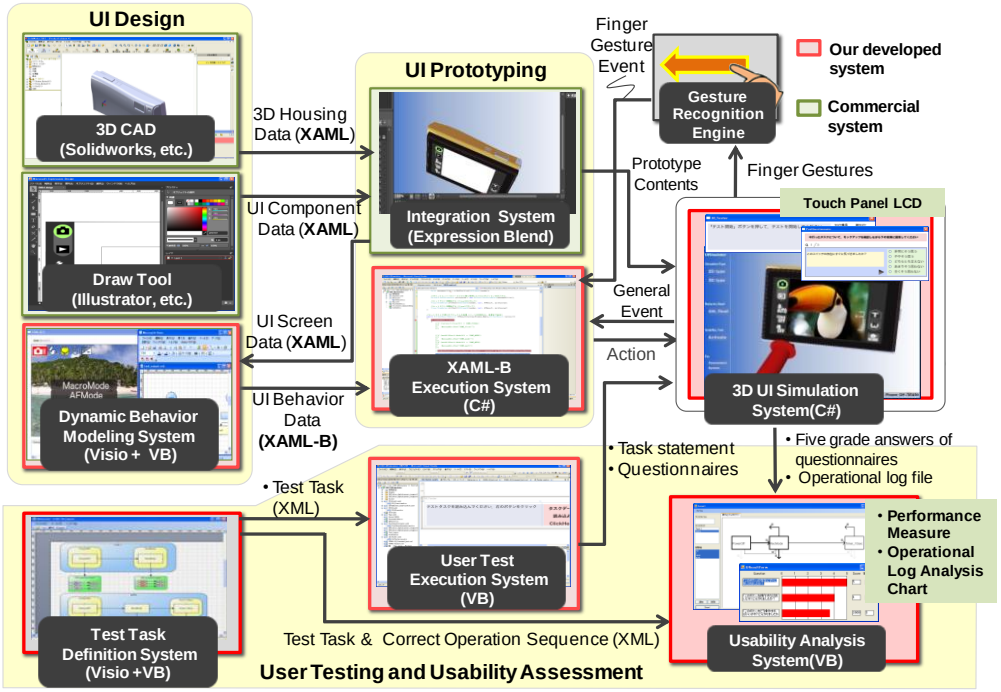


Figure 22. An overview of the XAMLS-based prototyping, user-test and usability assessment tools

In order to solve these problems, our group developed the second version tools for user interface prototyping and usability assessment[12]. As shown in Figure 22, the new systems enabled 3D digital prototyping of information appliances with touch sensitive interfaces and also enable automated user test and usability assessment. The 3D digital prototype is defined by the combination of XAML and XAML-B which is our original extension of XAML.

The technical features of the proposed systems are summarized as the followings;

- XAML allows UI designers to declaratively describe static structures both of 2D UI components and 3D housing, and its vector-graphic rendering engine can generate high-quality UI images on-the-fly. So, generation of snapshot images of UI screen becomes unnecessary during the execution.
- The proposed XAML-B enables UI designers to declaratively describe dynamic behaviors as a set of event-based rules. It can eliminate their programming works and state explosion in the UI behavior modeling.
- Several commercial tools were already available for defining and integrating the 2D UI static structure. They could be used even for 3D prototyping.
- A standard PC environment is only needed for 3D UI simulation. Any special commercial Web3D player is unnecessary.
- The processes of the user test and the usability assessment are fully automated along with the 3D digital prototype.
- Gesture recognition function enables the users to manipulate touch-sensitive UI on the 3D digital prototype.

XAML-based 3D UI Prototyping

XAML

XAML (eXtensible Application Markup Language) was developed by Microsoft [18,30] as a UI specification targeted for Windows applications. XAML is an XML based mark-up language which specifies the static structure of a UI running on the WPF (Windows Presentation Foundation). WPF is a UI framework to create applications with a rich user experience, and combines applications UIs, 2D graphics and 3D graphics and multimedia into one framework. Its vector-graphic rendering engine makes the UI implementation fast, scalable and resolution independent. WPF separates the appearance of an UI from its behavior. The static structures of 2D UI screen appearances is declaratively specified in XAML, while the behavior has to be implemented in a managed programming language like C#.

XAML-B for dynamic behavior modeling

Modeling Concept of XAML-B

We extended XAML specification so that the UI designers can declaratively define dynamic behavior only by writing a XML document with simple syntax. This extension part of the XAML is named XAML-B (XAML-Behavior).

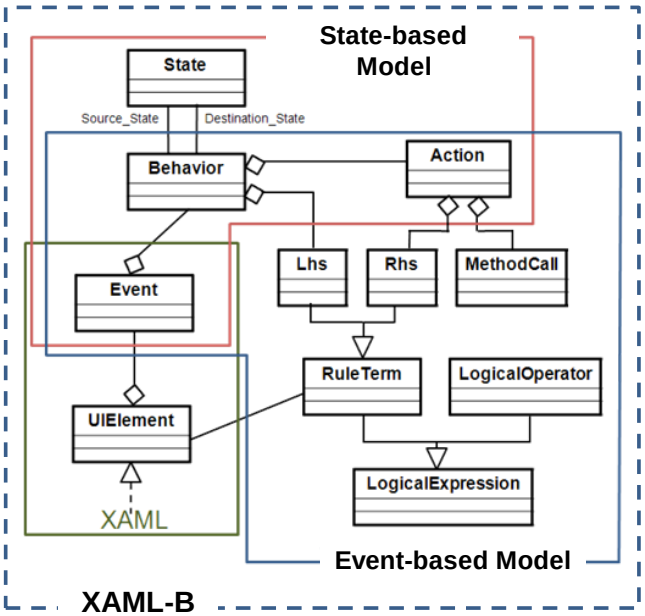


Figure 23. UML class diagram of XAML and XAML-B

We also proposed a two-stage modeling process of dynamic behaviors both of which is supported by XAML and XAML-B.

In these stages, two sub-models of XAML-B were respectively used to define the dynamic behavior; the state-based model for early design stage and the event-based model for detailed design stage. The UML diagram describing class structures of XAML-B is shown in Figure 23. The specification of the XAML-B includes both state-based model, event-based model and the reference to the original XAML specifications. The details of these two models are explained as the following sub-sections.

State-based model for early design stage

At the early design stage, the dynamic behavior of the UI is modeled as a state-based model, because the number of states is relatively small and the state-based model enables UI designers to capture a flow of user interactions at a glance. Rough interaction flows of the UI are initially captured as a state-based model.

Figure 24-(a) shows an example of the state-based model in case of the power-on UI behavior in a digital camera. A set of the classes of XAML-B included inside the state-based model in Figure 23 is used. The state-based model consists of the classes of *State*, *Event*, *Behavior* and *Action*.

A *State* expresses unique combination of attribute values which expresses an appearance of the UI. An *Event* is an incoming phenomenon to the UI such as “button-pushed” or “icon-tapped”. An *Action* is a process triggered by an event that causes a state change such as change in an icon on the screen or mechanical motion of the appliance. A *Behavior* means a notion of state-transition composed of *Source_State*, *Destination_State*, *Event* and *Action*.

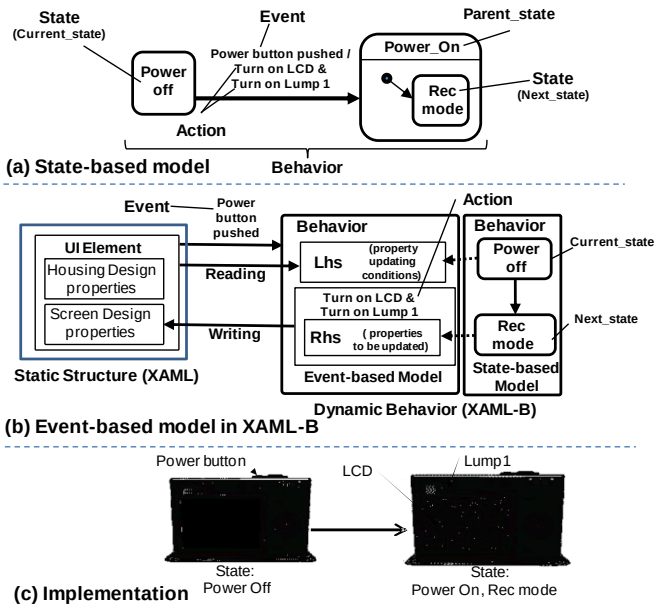


Figure 24. Examples of state-based modeling and event-based modeling of UI behaviors in XAML-B

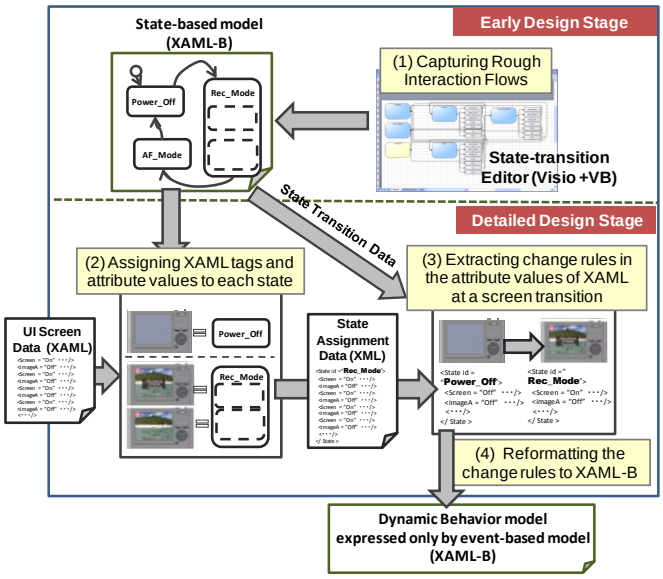


Figure 25. Dynamic behavior modeling process in the XAML-based system

Event-based model for detailed design stage

Once the UI design enters the detailed design stage, the number of states tends to explode when using the state-based model. So, an event-based model is used at the detail design stage. The event-based model enables the designers to describe the detail control of UI components indicated on the screen whose notion is originally included in XAML specification.

Figure 24-(b) shows an example of the event-based model which is described based on the state-based model example of Figure 24-(a). The event-based model was made up on the basis of UsiXML [16, 28], which is the other XML-compliant UI description language and has declarative description of event-based dynamic behavior. A set of the classes of XAML-B included inside the event-based model in Figure 23 is used.

The event-based model consists of *Behavior*, *Lhs*, *Action* and *Event*. Notions of *Behavior*, *Action* and *Event* are the same as those in the state-based model. An *Lhs* expresses the conditions under which each *Action* become executable. An *Action* consists of the combination of *MethodCall* and *Rhs*. An *Rhs* specifies how the attribute values of XAML should change corresponding to the change of UI appearance. A *MethodCall* specifies an external procedure such as an animation clip or a sound clip. A pair of an *Lhs* and an *Rhs* describes a state-transition rule. And a *RuleTerm* expresses a condition that attribute values in XAML must fulfill before and after an *Action* occurs.

Dynamic behavior modeling system and the modeling process

We developed a prototype of dynamic behavior modeling system which was implemented by the combination of Visio and Visual Basic. Figure 25 shows the functions of the modeling system. It can support both of the state-based and event-based modeling processes.

The modeling flow of the system is also shown in Figure 25. In the early design stage, as in the upper part of Figure 25, rough interaction flows are initially captured as a state-based model. In order to support efficient modeling work in this stage, a state-transition editor was implemented. In the editor, state, event and action can be graphically created and edited on the Visio by UI designer. This modeling result is stored in a XAML-B document.

In the detailed design stage, as shown in the lower part of Figure 25, first a set of tags and their attribute values of one UI screen which has been modeled as XAML document in the integration tool are assigned to one state. A new state tag is created in the XAML-B document, and a set of the XAML tags and their attribute values representing the state of the UI is packed inside the state tag. Two different sets of these tags and the attribute values each of which is assigned to a source or a destination state are then compared to each other. Taking XOR between these tags and attribute values automatically makes the change in attribute values of an Action tag in the event-based model. Finally reformatting the rule of change to an XML document makes a final XAML-B description corresponding to this state-transition.

Building process of 3D digital prototype

According to the process described in the previous section, a 3D digital prototype is built in the following processes shown in Figure 22;

(1) A 3D housing model is created in a commercial CAD system (Solidworks 2008) and is exported to a model integration system (Expression Blend) as a XAML document.

(2) Each graphical component (text, icon, etc.) of the UI is defined in a draw tool (Illustrator etc.) and is exported to an integration tool as a XAML document.

(3) In the integration tool, a set of the graphical components are combined to make one XAML document which defines each UI screen. And the 3D location of the UI screen is also specified onto the 3D housing model by this XAML document.

(4) The XAML document is imported to a dynamic behavior modeling system. The behavior of the UI is defined by a UI designer according to the process described in 2.3, and is outputted as a XAML-B document from the system.

(5) Finally, the XAML-B execution system reads this document and drives UI simulation on the 3D housing model responding to input events coming from the user of the digital prototype.

Gesture recognition function

Gestural interfaces become available in current information appliances with touch sensitive interfaces. So a gesture recognition function is installed in our prototyping system. Several types of user finger gestures inputted from a UI screen of a 3D digital prototype can be recognized as events, and can be processed in the XAML-B execution system. In the XAML-B specification, several gesture types that the function can recognize are described in the "*Gesture_Type*" attribute placed inside the "*Event*" tag.

Real-time gesture recognition is needed for smooth operation of UI simulation. InkGesture engine [5] is being used in the system. Four types of finger gestures of "Leftward", "Rightward", "Upward" and "Downward" can be recognized as Events in our system. The recognized gesture can then be processed as one of the events in XAML-B execution system.

This recognition function enables the users to operate touch sensitive interfaces modeled on the 3D digital prototype using not only mouse gestures but direct finger gestures.

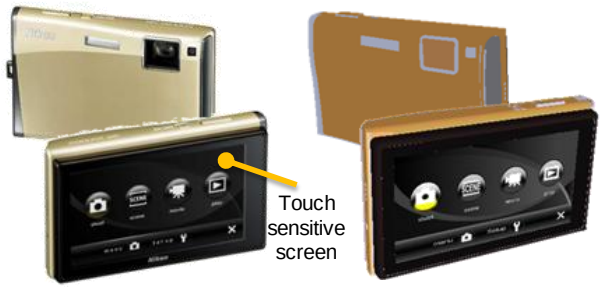
User test and usability assessment systems in XAML-based system

In our system, user test and usability assessment can be done on the 3D digital prototype. The test and assessment processes are as same as the ones described in the previous sections of the UsiXML-based assessment system.

A case study of user test

An example of the 3D digital prototype

A digital camera (Nikon-S60) with a touch-sensitive screen shown in Figure 26-(a) was adopted as an example of prototyping, user test and usability assessment. As shown in Figure 26-(b), the 3D housing model of the camera was



(a) Real Product (b) 3D Digital Prototype

Figure 26. A digital camera and its digital prototype

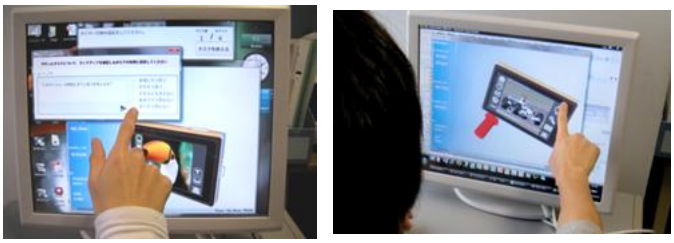


Figure 27. User-test situation with touch sensitive screen where a digital prototype is displayed

modeled in Solidworks 2008. Over 60 states and 100 state transitions were modeled in the dynamic behavior model described by XAML-B. The sound of finger touch is also emulated in the digital prototype to avoid missed-operations.

As shown in Figure 27, a 19-inch touch sensitive LCD monitor shown was used in the test, and the user can directly input by finger touch and finger gesture on the touch screen displayed as a part of the 3D digital prototype which is displayed on the touch sensitive LCD monitor. This enabled the users to operate the UI on the 3D digital prototype as realistically as the one on the real camera.

User test settings

The user test was done using the 3D digital prototype and the operational log analysis chart and five grade evaluation results were compared to those obtained from the one using the real camera. 35 subjects who had no experience of using this camera attended the test. Among them, 17 subjects operated the 3D digital prototype and 18 subjects the real camera.

Two tasks shown in Figure 28 which include the basic UI operations were given to the subjects. In task 1, they asked to set the self-timer duration for 10 sec from the power-off state. In task 2, they asked to enter the preview mode state from a power-off state and to indicate the first picture by turning over the pictures indicated on the touch screen. In the task 2, two correct sequences of operations exist as shown in Figure 28; the one of pushing an up-arrow or a down-arrow icon indicated at the corner of the touch screen, or the one of directly sliding the finger leftward or rightward on the touch screen.

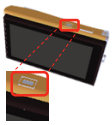
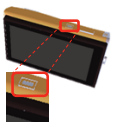
	Task description	# of correct routes	Correct operation sequence and states at which digital questionnaires appear				
Task1	Set the self-timer duration for 10 sec from power-off state	1	State	Power off	Shooting	Timer Select	Timer_10s
			Correct operation	Push power button 	Push timer icon 	Push 10s icon 	
Task2	Enter preview mode from a power-off state, and display the pre-specified picture on the touch screen using finger gesture	1	State	Power off	Shooting	Play_Mode	Play_First
			Correct operation	Push power button 	Push playback icon 	Slide a finger horizontally 	
		2	State	Power off	Shooting	Home	Play_Mode
			Correct operation	Push power button 	Push home icon 	Push playback icon 	Slide a finger horizontally 

Figure 28. The test tasks and their correct operations

The five grade evaluation results for the questionnaires were also obtained from the subjects who took a missed operation.

Usability assessment results

The operational log analysis charts and 5 grade evaluation results for the questionnaires at a state of missed-operation in case of the real product and the digital prototype in task 1 are shown in Figure 8.

From Figure 29, it was found that many subjects (12/18 in case of real camera, and 9/17 persons in case of digital prototype) took missed operations at the "Shooting" state. In this state, they should push the small timer icon indicated lower left of the touch screen, but they found themselves lost deep in the menu hierarchy toward incorrect states. Two missed-operation patterns in Figure 29-(a) and (b) were very similar to each other. Moreover, the results of the five grade evaluation for the questionnaire at the "Shooting" state showed that most of the missed subjects did not notice the icon to be pushed and did not understand that they should operate it. This figure also showed that same tendencies of the evaluation results were observed both in the digital prototype and in the real camera.

While in case of the task 2, as shown in Figure 30, most of the subjects (8/10 in the real camera and in the digital prototype) did not complete the task using the finger gestures, but could it by taking alternative correct operation (pushing the icons). The five grade evaluation results for the questionnaire at the "Play_Mode" state also showed that the subject who took this alternative operation ("Play_Mode" -> "Play_Detail" -> "Play First") did not notice that the

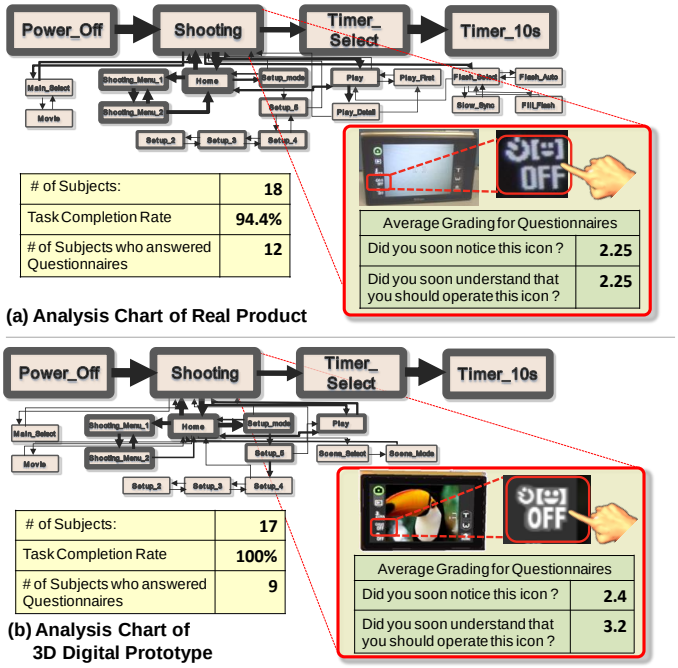


Figure 29. The operational log analysis charts and five grade evaluation results in Task1

camera could accept direct finger gestures and that they should make them to complete the task.

These test results showed the following clear facts:

- -In the "Shooting" state and the "Play_Mode" state, the icon indicated on the UI screen could not ade-

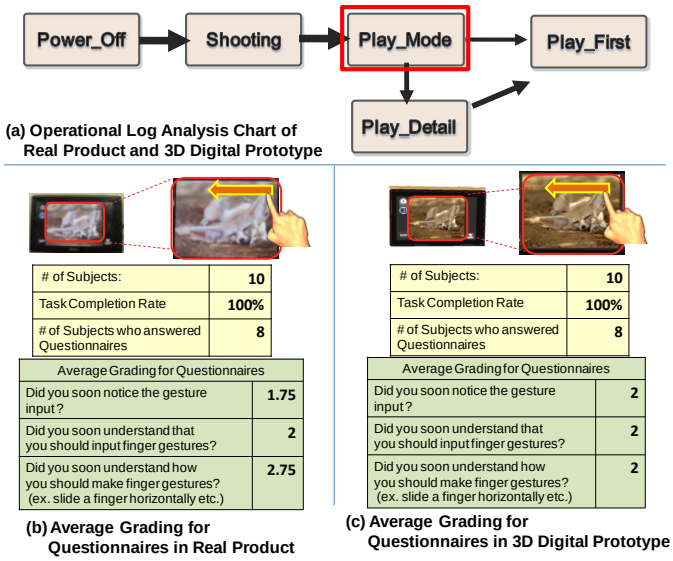


Figure 30. The operational log analysis charts and five grade evaluation results in Task2

quately make most users think of the correct input operations intended by the designers. So, these icons should be strongly redesigned to improve usability.

- There were strong correlations of user operation sequence and the ratings of the questionnaires between the 3D digital prototype and the real product. This suggests that the 3D digital prototype with touch sensitive interface could replace a physical prototype while keeping the ability to find usability problems from the prototype.

The effect of UI redesign on the digital prototype

The result of task 2 revealed that the icons indicated on the UI screen at "Play_Mode" state could not make the users think of direct finger gesture to turn over the indicated pictures. To solve the problem, a reciprocal motion animation of a finger icon was newly added on the screen when entering this state.

After this redesign, an additional user test for the task 2 was done by 7 new subjects, and the results were analyzed. Figure 31 shows the operational log analysis charts and five grade evaluation results for this new test. Six of the seven subjects could take direct finger gestures to turn over the pictures this time. Also from the grade evaluation, one subject who took a missed operation could even notice the finger gesture input at this state and actually inputted the gesture.

Only 10 minutes were needed for creating and inserting this animation file name into the original XAML-B document in the redesign. This fact showed that our proposed systems enabled UI designers to realize very rapid turnaround of design-test cycle compared to the one of physical prototypes.

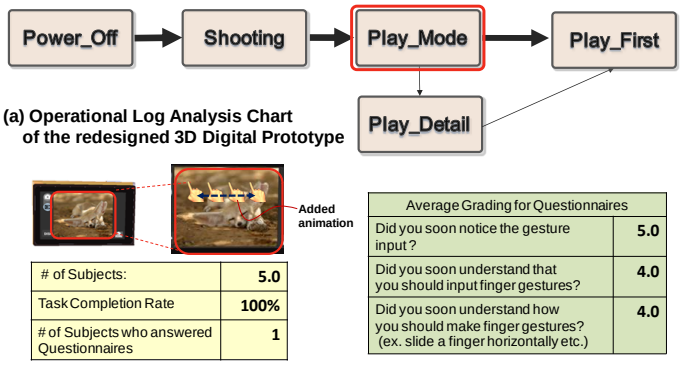


Figure 31. The operational log analysis charts and five grade evaluation results in Task2 after UI redesign

CONCLUSIONS

Usi-XML-based and XAML-based systems of prototyping, user testing and usability assessment were proposed which enabled 3D digital prototyping of the information appliances. To declaratively describe the static structure and the dynamic behavior of the user interface of the appliances with physical user interface elements, Usi-XML and XAML were originally extended, and its UI simulation system were developed. Gesture recognition function enabled the subjects to manipulate the touch-sensitive UI of the 3D digital prototype in the user test. User test execution and analysis of missed operations could be fully automated. The results of the user test and usability assessments for the digital camera showed that major usability problems appearing in real product could be fully extracted even when using the digital prototype, and that the proposed systems enabled rapid turnaround time of design-test-redesign-retest cycle.

The user test and usability assessment could be fully automated by the proposed system. But there are still open-problems to be solved in our research. The major one is whether the proposed two-stage modeling process of UI behaviors is actually understandable and accessible for most interaction designers compared to the current prototyping tools like Flash.

As a result of the development of our two XML-based computer-aided prototyping and usability assessment tools for UI, we are concluding that, so far, it is the best way to combine the UsiXML-based model-driven hierarchical development framework with XAML-based implementation. UsiXML provides clarity in capturing and describing the UI system ranging from the conceptual design to the concrete stage in declarative way, while XAML does excellent ability and fidelity of the 2D and 3D integrated UI simulation in much inexpensive environments.

At this moment, the dynamic behavior modeling system is still a prototype phase, and "usability" of the system itself is still not fully considered and improved. Therefore our future research should include the usability evaluation on the proposed dynamic behavior modeling method and system

by interaction designers themselves. Statechart-based modeling process [4] which was adopted in our research can inherently offer interaction designers an ability of state-based step-by-step hierarchical modeling process of UI. The process also enables a good compatibility of UI code generation process. Confirming the effectiveness of this concept by experiments will be included in our future research.

ACKNOWLEDGMENTS

We gratefully acknowledge the supports of the Grant-in-Aid for Scientific Research under the Project No.19650043 (Term: 2007-2008) and No.21360067 (Term: 2009-2011) funded by the Japan Society for the Promotion of Science, and of the Grant-in-Aid for Seeds Excavation (Term: 2009) funded by Japan Science and Technology Agency.

REFERENCES

1. 3D XML: www.3ds.com/3dxml.
2. Broak, J., "SUS: a 'quick and dirty' usability scale", Usability Evaluation in Industry. Taylor and Francis, 1996
3. Hori, M. and Kato, T., "A Modification of the Cognitive Walkthrough Based on an Extended Model of Human-Computer Interaction (in Japanese)", Trans. Information Processing Society Japan, 48(3): 1071-1084, 2007.
4. Horrocks, I., "Constructing the User Interface with Statecharts", Addison-Wesley, Harlow, 1999.
5. Microsoft.Ink Gesture class, available at http://msdn.microsoft.com/en-us/library/microsoft.ink.gesture(v=VS.85).aspx
6. ISO13407, 1999, "Human-centred design processes for interactive systems". 1999.
7. ISO9241-11, "Ergonomic requirements for office work with visual display terminals (VDTs) – Part 11: Guidance on usability" , 1998
8. Kuutti,K., et. al., "Virtual prototypes in usability testing", Proceedings of the 34th Hawaii International Conference on System Sciences, 5, 2001.
9. Kanai,S., Horiuchi,S., Shiroma,Y. and Kikuta,Y., "Digital usability assessment for information appliances using User-Interface operable 3D digital mock-up", Research in Interactive Design, 2, Springer: VC_HUCEID2006_p235, 2006
10. Kanai,S., Horiuchi,S., Shiroma,Y., Yokoyama, A. and Kikuta,Y., "An Integrated Environment for Testing and Assessing the Usability of Information Appliances Using Digital and Physical Mock-Ups", Lecture Notes in Computer Science, 4563: 478-487, 2007
11. Kanai, S., Higuchi, T. and Kikuta Y., "3D digital prototyping and usability enhancement of information appliances based on UsiXML", International Journal on Interactive Design and Manufacturing, 3(3):201-222, 2009.
12. Kanai, S., Higuchi, T. and Kikuta Y., "XAML-Based Usability Assessment for Prototyping Information Appliances with Touch Sensitive Interfaces", Research in Interactive Design, 3, Springer: PRIDE-P189, 2010.
13. Kerttula, M., Tokkonen, T. K, "Virtual design of multi-engineering electronics systems", IEEE Computer, 34(11): 71-79, 2001
14. Landay, J.A. et al., "Sketching Interfaces: Toward more human interface design", IEEE Computer, 34(3): 56-64, 2001
15. Lin, J., et al., "DENIM: Finding a tighter fit between tools and practice for web site design", Proceedings of Human Factors in Computing Systems: CHI 2000, 2(1): 510-517, 2000
16. Limbourg, Q., Vanderdonckt, J., Michotte, B., Bouillon, L., Víctor López Jaquero, "UsiXML: a Language Supporting Multi-Path Development of User Interfaces", Lecture Notes in Computer Science, 3425: 200-220, 2005
17. Cybelius Maestro, available at http://www.nickom.co.jp/product_English.html
18. MacVittie, L.A., "XAML in a Nutshell", O'Reilly Media, 2006.
19. Norman, D. A., "Cognitive engineering" In D. A. Norman and S. W. Draper (Eds.), User Centered Systems Design: New Perspectives in Human-Computer Interaction,:31-61, Hillsdale, NJ: Lawrence Erbaum Associates, 1986
20. Park,H., Moon,H.C. and Lee, J.Y., "Tangible augmented prototyping of digital handheld products", Computers in Industry, 60(2), 114-125, 2009.
21. Protobuilder, http://www.gaio.co.jp/product/dev_tools/pdt_protobuilder.html.
22. RAPID PLUS, http://www.e-sim.com/products/ rapid_doc/index-h.htm.
23. UIML(User Interface Markup Language) v3.1 Draft Specification, http://www.uiml.org/, 2004
24. UsiXML, http://www.usixml.org/
25. Viewpoint, http://www.viewpoint.com/pub/technology/
26. VRML97, "Functional specification and VRML97 External Authoring Interface (EAI)", ISO/IEC 14772-1:1997 and ISO/IEC 14772-2, 2002
27. Virtools: www.virtools.com
28. Vanderdonckt, J, "A MDA-compliant environment for developing user interfaces of information systems, Lect. Notes Comput. Sci., 3520, 16-31 (2005)
29. XML User Interface Language (XUL) 1.0, http://www.mozilla.org/projects/xul/xul.html , 2001
30. Extensible Application Markup Language (XAML), http://msdn.microsoft.com/en-us/library/ms747122.aspx, 2008.