Scenario-Based Interactive UI Design

Koki Kusano **Momoko Nakatani** **Takehiko Ohno**
NTT Service Evolution Laboratories
1-1 Hikari-no-oka, Yokosuka-Shi, Kanagawa, 239-0847 Japan
{kusano.kouki, nakatani.momoko, ohno.takehiko}@lab.ntt.co.jp

ABSTRACT
Clearly picturing user behavior is one of the key requirements when designing successful interactive software. However, covering all possible user behaviors with one UI is a complex challenge. The Scenario-based Interactive UI Design tool is designed to support the characterization of user behavior based on scenarios and then using the information in UI design. Scenarios make it easy to understand and share user behavior even if we have little design knowledge. However, they have two big weaknesses; 1) integrating several scenarios in one UI is difficult, even if we can create appropriate scenarios, 2) maintaining the links between scenarios and the UI is a heavy task in iterative design. Our tool solves the above problems through its hierarchical scenario structure and visualized overview of scenarios. It enhances the designer's skill in writing scenarios and designing UIs smoothly and easily.

Author Keywords
User interface design; Scenario; Design tool; Traceability

ACM Classification Keywords
H5.2 [Information interfaces and presentation]: User Interfaces. - Graphical user interfaces.

INTRODUCTION
A good understanding of user behavior is essential when designing interactive software that must provide high usability and user experience. Scenarios, which describe how people accomplish tasks, are a popular design tool; they are one way of making the user image concrete and extracting user requirements in Scenario-Based Design (SBD) [1]. In the human-computer interaction field, scenarios are also utilized by designers when designing UIs [2]. Scenarios that are written in a natural language are comprehensible even if the writer has insufficient experience in considering user-computer interaction. Hence, scenarios help a designer to facilitate agreement and consistency with stakeholders (ex. developers, clients).

However, managing scenarios becomes more difficult as the number and complexity of scenarios increases [3]. Therefore, scenarios should offer an overview of their contents by using some form of graphical representation to assist in the understanding of user behavior [4]. This is a critical requirement if the designer has to find common points and trade-offs among the scenarios and integrate all points into one UI [3]. In addition, updating the links between scenarios and UI is very tough when the design is iterated [5]. When a designer revises a UI based on the results of usability testing, he/she has to refer to not only the UI but also the original scenarios at the same time; this process should be iterated to reach the required quality. This difficulty renders scenarios less useful.

Therefore, designers must be able to overview, understand and manage scenarios easily and smoothly when designing a UI. Unfortunately, no tool well supports this requirement, and efficiency strongly depends on the designer's skill. In this paper, we solve these problems with the Scenario-based Interactive UI Design Tool (see Figure 1): we especially focus on 1) managing scenarios and 2) visualizing the relationships among multiple scenarios. First, our tool supports the designer to reorder the sentences in each scenario to more clearly express the hierarchical structure of the scenario. In addition, a designer can attach a tag that expresses a target user requirement to any sentence. Second, the tool automatically visualizes the scenario structure (overview) by using the tags. The designer can directly alter the UI via the overview. The overview provides easy understanding by focusing on the key common points and conflicts among the scenarios. In addition, the tool maintains the links among sentences, tags, visualization and UI. The above features allow a designer to write and manage scenarios and design a UI easily and smoothly in an iterated design process.

RELATED WORK
Most SBD methods iterate the following four steps: 1.field study, 2.create user image and scenarios, 3.design UI, 4.evaluate UI.

First, the field specialist observes target-users conducting interviews that reveal their actions in the target situation. Second, the designer creates a concrete user image and writes scenarios as needed based on the field data. Each scenario contains one user goal, situation, and behaviors to achieve the goal. Scenarios don't contain UI-components or functions to avoid the technology constraint trap. Next, the designer extracts user-requirements and functions, and then details the scenarios that contain UI-components. Third, the

CHI 2013, April 27–May 2, 2013, Paris, France.

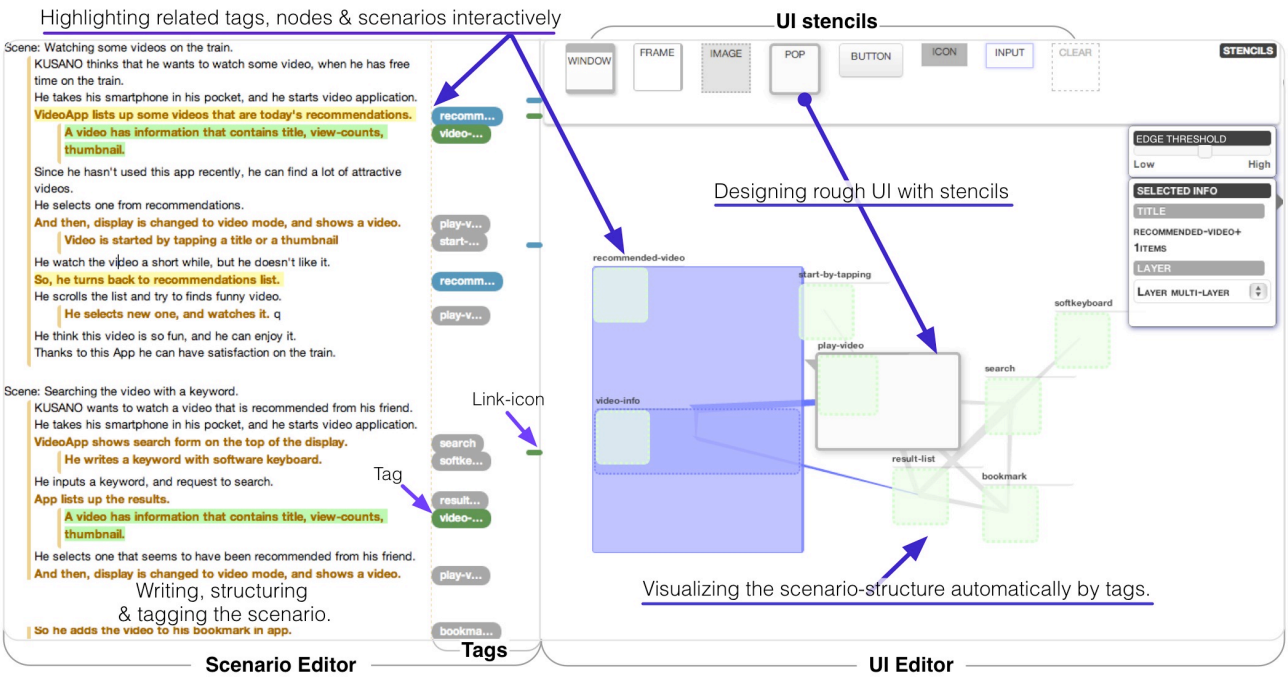

Figure 1: Scenario-Based Interactive UI Design Tool

designer designs the UI based on the scenarios. This process should be iterated with the scenario detailing step because the designer can realize which UI-components or functions are needed by observing the UI. Fourth, the designer evaluates the UI through usability testing and then revises the UI based on the test-results.

The process between step-two and step-three contains a problem; it is difficult to understand and manage scenarios and UI. Candidate solutions include structured-scenarios [6] or graphics and flow-charts [4]. These proposals certainly help the designer to understand the user requirements, however, they fail to support the management of the links among scenarios, requirements, and UI.

The task of maintaining traceability between requirements and program-codes has been well studied [7]. The model-driven UI automatic generation approach has been studied for many years [8]. This approach makes it easier to manage traceability. However, the model is difficult to understand without special knowledge. This characteristic is not suitable for designing UIs since there are various stakeholders who don't have specialist knowledge as regards UI design. In the field of SBD, to maintain the links between requirements and scenarios for software development, scenario-browser [9] or agile development process with scenario [10] has been proposed. However, they have not focused on managing the links among scenarios, user requirements and UI.

Finally, various prototyping tools have been proposed to design and evaluate UIs efficiently, and some of them have been utilized in real projects [11]. Prototyping tools obviously support the designer by allowing the design appearance of the UI to be more rapidly settled. However, current prototyping tools are unable to integrate all scenario information into one UI.

SCENARIO BASED INTERACTIVE UI DESIGN TOOL

The scenario based Interactive UI design tool provides support for scenario management and UI design based on scenarios. Furthermore, it offers traceability between scenarios and UI such that a designer can efficiently iterate the UI design-evaluation process. Figure 1 shows the GUI of the proposed tool; a browser-based application developed in HTML5 and JavaScript. Its key parts are the Scenario Editor and UI Editor. The designer is able to smoothly switch between them depending on the situation.

Support of Scenario Management.

The Scenario Editor has three steps for creating and managing scenarios: 1.Writing a scenario, 2.Structuring the scenario and 3.Tagging the scenario.

First, the designer writes, like a novel, a scenario that contains user behavior when using the software (ex. background, motivation, goal, steps).

Second, the designer creates hierarchies of sentences based on sentence detail. This operation is similar to the use of the outline-view. For the example shown in Figure 1 left, the first hierarchy layer holds scene titles. The second layer holds user behavior with no reference to UI expressions. The third layer holds concrete UI expressions. The hierarchy clearly shows the details of each sentence (does it contain UI components or not). Traditional SBD separately writes scenarios with and without UI components. The drawback of traditional scenarios is they make it hard to

maintain the links because the two types of scenarios are written separately. On the other hand, our tool uses hierarchy to integrate them. However, we feel that users need to understand the adjustment rules that will yield the proper hierarchy. Thus, our tool shows tooltips as guides for hierarchy adjustment. For example, if there are UI-component expressions (ex. window, button, icon) in the second layer, the tooltip recommends moving them to the third layer.

Third, the designer extracts tags, text labels, from the scenarios. The designer selects a sentence that contains requirements that impact UI design and manually creates a tag for the sentence. A floating tag box is used to input tags as shown Figure 2. The designer can quickly reuse registered tags since this box support incremental tag search. The tool automatically maintains each tag and links between the sentence and the tag.

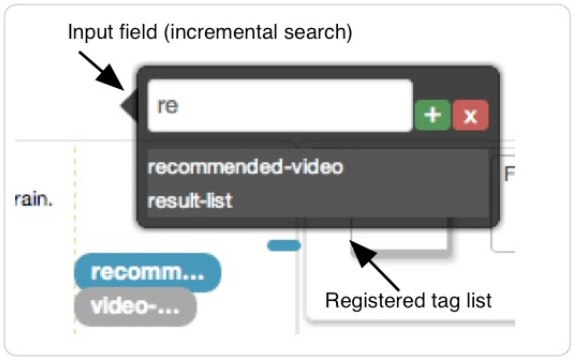

Figure 2: Floating Tag Box

Support of Designing UI

The UI Editor automatically visualizes the relationships of sentences by the tags (see Figure 1 right); this helps the designer to overview the scenarios and create a rough UI. The relationship is visualized as a graph that uses nodes (translucent square with dashed border) as tags and edges (line) as the relationships between tags, which is based on the tag position in the scenario hierarchy. The visualized-node positions are calculated by the spring-model. Node-size increases with frequency of usage in the scenario. Edge-width is calculated from the distance between tags in the scenario structure (sentences on same layer and next/previous sentences have strong ties). An edge is automatically created if tag distance is under a distance threshold (edge-threshold).

Graphs are very effective for overviewing relationships. In particular, using a graph to visualize the scenario structure helps the designer to visually consider groups of requirements and possible conflicts. For example, a node that has many edges is visually obvious and logically important in the graph because this node (tag) is related to many scenarios and can be the cause of user confusion. The current prototype focuses on several tens of nodes, and the default edge threshold is optimum for up to ten scenarios and tens of nodes. Hence, it provides only plain interaction (no zooming and filtering) with the graph. However, users can manually adjust the threshold for the number of nodes by slider operation; the graph is automatically updated.

The designer can create UI-components directly on the graph by selecting a stencil (see Figure 1 top) and dropping it on a node; nodes can be moved and layouts created. Figure 1 right shows the middle part of the UI design process (intermediate between graph and UI). It shows that our tool can maintain the links among scenarios, tags, graphs and UI. This feature shortens the design time. Even if the designer sets a UI-component to a node, both the visualized data and the UI-component can be easily recognized due to the overlaying of visualized data. The designer easily turns off the graph with single click. Incidentally, the current prototype does not focus the creation of a click-through prototype. This functionality can be added, however, it is well supported by various prototyping tools.

Provides Traceability among Scenarios, Tags and UI

Our tool always maintains the data of scenarios, scenario-structures, tags, UI-components and the links created by the designer. Because of this, it allows scenario construction to become a environment in which to envisage the design at a rough scale and then detailing that design by re-linking design choices back to earlier versions.

Figure 1 shows the linkage between highlighted nodes, tags, and scenarios. Highlighted nodes and edges have different color and tone. The designer understands quickly the number of tags used and their source sentences, since the scenario editor shows link-icons (see Figure 1). If the designer clicks a link-icon, the scenario editor immediately shows the linked sentence and the tag in question. For example, after setting a UI-component on each node and laying the nodes out, if the designer selects a scenario to better understand the links between the scenario and its UI-components, our tool automatically highlights only those UI-components related to the sentence in question. In addition, our tool warns users when they delete a sentence that has a tag; the node that was linked to the deleted sentence changes its color.

The above features offer high traceability and interactivity between scenario writing and UI designing. They encourage the designer to concentrate on imagining user behavior and designing the UI. Additionally, this feature strongly supports iterative-design, since it allows quick reference to data related to the issue. In iterative-design, the designer has to repeatedly evaluate the UI, find an issue and fix it. This means that the designer should refer to not only the UI, but also the related data of scenarios and tags. It is obviously a heavy task without our tool.

Furthermore, the designer must create test scenarios for usability testing. First, the designer decides which part of

the UI should be evaluated; he/she can quickly refer to the related scenarios and choose to create an abstract scenario or a concrete scenario depending on the evaluation objective. This feature strongly supports iterative design.

USER STUDY

We conducted a preliminary user study to get user feedback. Three participants who were unfamiliar with SBD used a prototype of our tool. We requested them to design one UI suitable for two situations; Scene 1, user uses a smartphone application to browse a movie website, and Scene 2, user wants to search and watch a movie recommended by a friend via a smartphone application. We mainly observed design activities related to writing, structuring, and tagging scenarios and designing a rough UI.

The participants commented that writing and structuring the two scenarios was useful for clarifying user behavior and for finding vague points. This result confirms the effectiveness of the hierarchy structure. However, they also commented that it was difficult to set appropriate hierarchies at first. One solution is to show scenario samples or tutorials.

Furthermore, they noted the benefit of the graph-based form of visualization. They could understand quickly which tags were key and which were related tags. Our proposed visualization and interaction method is simple, but we could confirm its effectiveness for understanding relationships between tags. Note that this benefit strengthens with scenario size. Thus, we plan to extend visualization by adding some of the interaction techniques proposed in the research field of visualization that will enhance its scaling performance and conduct further experiments to confirm this strength.

In the UI design stage, when the participants identified some shortfall in the UI, they could quickly and easily trace and revise the related sentences and tags. This means that the participants performed the iterated design process, while smoothly tracing the data relationships identified by highlighted sentences, tags and UI components. On the other hand, when participants wanted to detail the UI, a new sentence had to be created to anchor the new UI-component. This seems to be somewhat inefficient. This case study revealed that implementing an annotating-function that does not involve the addition of sentences would further enhance the iterative process. The above results show that this tool is effective for managing scenarios and designing UIs.

CONCLUSION

We have presented the Scenario-based Interactive UI Design tool; it supports designers in structuring scenarios and maintains the links among scenarios, tags, and UI. We also have shown that it enables designers to concentrate on design-tasks and allows the iterative design process to be conducted smoothly and efficiently by providing highly effective support. A user study gathered positive feedback from all participants. In the future, we will conduct examine bigger scenarios to more fully evaluate the effectiveness of interactivity and traceability among scenarios, tags and UI. We also plan to put our tool into commercial use (in-house) to observe the resulting design activities for a detailed evaluation of its practicality.

REFERENCES

1. Carroll, J. M. *Making Use: Scenario-Based Design of Human-Computer Interactions.* The MIT Press, 2000.
2. Cooper, A., Reimann, R., and Cronin, D. *About face 3: the essentials of interaction design.* John Wiley & Sons, Inc., New York, NY, USA, 2007.
3. Hertzum, M. Making use of scenarios: A field study of conceptual design. *International Journal of Human-Computer Studies 58* (2003), 215–239.
4. Gough, P. A., Fodemski, F. T., Higgins, S. A., and Ray, S. J. Scenarios-an industrial case study and hypermedia enhancements. *In Proceedings of the Second IEEE International Symposium on Requirements Engineering*, IEEE Computer Society (1995), 10–17.
5. Nielsen, J. Iterative user-interface design. Computer 26, 11 (1993), 32–41.
6. Yanagida, K., Ueda, Y., Go, K., Takahashi, K., Hayakawa, S., and Yamazaki, K. Structured Scenario-Based design method. In Human Centered Design, vol. 5619 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg (2009), 374–380.
7. Winkler, S., and von Pilgrim, J. A survey of traceability in requirements engineering and model-driven development. *Software and Systems Modeling 9* (2010), 529–565.
8. Sousa, K., Mendonca, H., Vanderdonckt, J., Rogier, E., and Vandermeulen, J. User interface derivation from business processes: a model-driven approach for organizational engineering. *In Proceedings of the 2008 ACM symposium on Applied computing*, ACM (2008), 553–560.
9. Rosson, M. B., and Carroll, J. M. Integrating task and software development for object-oriented applications. *In Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM Press/Addison-Wesley Publishing Co. (1995), 377–384.
10. Obendorf, H., and Finck, M. Scenario-Based usability engineering techniques in agile development processes. *In CHI '08 extended abstracts on Human factors in computing systems*, ACM (2008), 2159–2166.
11. Tang, L., Yu, Z., Zhou, X., Wang, H., and Becker, C. Supporting rapid design and evaluation of pervasive applications: challenges and solutions. *Personal Ubiquitous Comput. 15, 3* (2011), 253–269.