Zurich Open Repository and Archive
University of Zurich
University Library
Strickhofstrasse 39
CH-8057 Zurich
www.zora.uzh.ch

Year: 2012

Flexible, lightweight requirements modeling with FlexiSketch

Wüest, Dustin ; Seyff, Norbert ; Glinz, Martin

DOI: https://doi.org/10.1109/RE.2012.6345826

Posted at the Zurich Open Repository and Archive, University of Zurich
ZORA URL: https://doi.org/10.5167/uzh-72267
Conference or Workshop Item

Originally published at:
Wüest, Dustin; Seyff, Norbert; Glinz, Martin (2012). Flexible, lightweight requirements modeling with FlexiSketch. In: 20th IEEE International Requirements Engineering Conference, Chicago, USA, 24 September 2012 - 28 September 2012. IEEE, 323-324.
DOI: https://doi.org/10.1109/RE.2012.6345826

Flexible, Lightweight Requirements Modeling with FlexiSketch

Dustin Wüest
University of Zurich
Zurich, Switzerland
wueest@ifi.uzh.ch

Norbert Seyff
University of Zurich
Zurich, Switzerland
seyff@ifi.uzh.ch

Martin Glinz
University of Zurich
Zurich, Switzerland
glinz@ifi.uzh.ch

***Abstract*—Early stage requirements models are often documented using paper and pencil-based approaches. In our current research, we are exploring lightweight modeling tools and approaches that could provide a beneficial alternative. We have developed the FlexiSketch tool prototype which combines support for free-form sketching with lightweight metamodeling capabilities. This creates the possibility for an automatic transcription of the documented information in later modeling stages. The tool is designed to be used on tablet devices.**

***Keywords*-requirements modeling; free-form sketching; lightweight meta-modeling; sketch recognition**

I. Introduction

Paper and pencil-based tools seem to be the most widespread support for early requirements modeling activities. Requirements engineers use flip charts, white boards, or just a simple sheet of paper to document early ideas. Although these tools have a lot of benefits and allow for free-form sketching, there are negative effects as well. As the documented information is not available in a structured form, changes and post-processing opportunities are limited. In order to allow for further modifications, a time-consuming and costly re-modeling of the early sketches is necessary. This can be done using software tools for desktop platforms that often follow more formal state-of-the-art modeling approaches such as UML (e.g. [1]). In several projects, those models can be seen as drivers for later software design and development activities.

With the availability of mobile tablet devices and digital whiteboards, we envision a more seamless integration of early RE activities in the software development process [5]. Our research focuses on methods and tools that combine the benefits of paper and pencil-based approaches with more formal software modeling methods to support early RE sketches. We have therefore developed a conceptual solution that supports lightweight, sketch-based RE modeling [4]. It focuses on free-form sketching support for requirements engineers, but also allows for a seamless transformation of the sketched information such that it can be imported into well-established modeling tools. In this regard, our solution is thought to be a combination of tools for free-form sketching, e.g. [6], and tools that recognize certain diagram types, e.g. [7]. So the requirements engineer can choose in which situations she wants to focus on sketching and at which points she provides the meta-model information needed for further processing. We have identified three key aspects regarding our conceptual solution:

Sketching: Free-form sketching with the help of paper and pencil based approaches is widely used and accepted [2,3]. Such approaches do not limit requirements engineers in any way. They can draw whatever they feel is appropriate in order to document the relevant information. Limitations, for example, the fact that documented information can’t be modified quicky, are accepted as the price for flexibility.

Allowing for free-form sketching is an important requirement regarding our envisioned approach. From our point of view, it is the main reason why people stick to paper and pencil-based approaches. To further support the paper and pencil metaphor, we also aim for a solution that can be used anytime, anywhere.

Metamodeling: While drawing a sketch, people often focus on the actual information the model should convey and often ignore more formal modeling conventions. Individual differences, when modeling certain information, lead to different models. These sketched models sometimes might only be readable and understandable by the creator herself. In most cases, information defining the underlying meta-model or syntactical information is missing.

In order to allow for the later transformation of the modeled information, lightweight metamodeling capabilities are needed that support the requirements engineer to define model elements and provide syntactical information.

Sketch recognition: As soon as the user adds a meaning to a certain element, it is necessary to detect all other appearances of that element type in order to allow automatic processing of drawn models.

We consider that our solution provides sketch recognition support in an interactive manner. The user is encouraged to take part in the recognition process in order to avoid misleading definitions. However, this will happen in an unobtrusive way which does not distract the requirements engineer.

II. The FlexiSketch Tool

The FlexiSketch tool prototype we have developed is based on our conceptual solution. It is available for the Android OS and is recommended to be used with tablet devices sized 7 inches and larger. In order not to break the paper and pencil metaphor, we consider using a stylus for working with FlexiSketch. However, the tablet screens are optimized for multi-touch finger gestures. Figure 1 presents a screenshot of FlexiSketch.

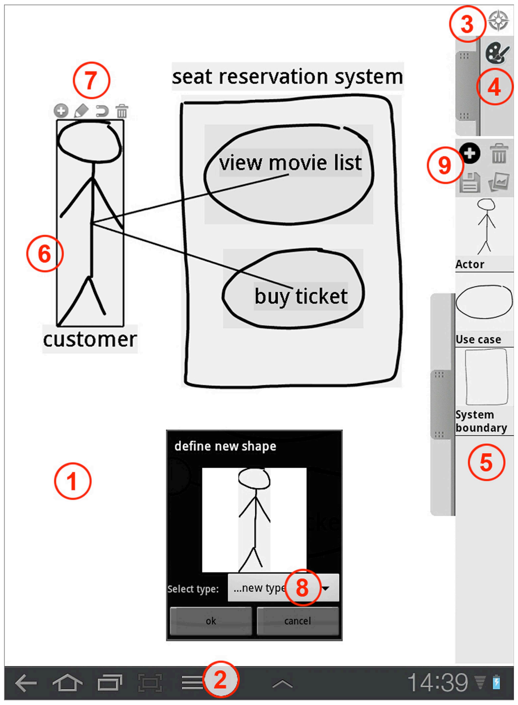

Figure 1. FlexiSketch Tool Prototype

The main aspects of the tool are highlighted with numbers (1-9). Similar to using paper for sketching, the user sees a white drawing canvas after starting FlexiSketch (1). A menu bar (2) providing some general options for saving and clearing the drawing space is presented at the bottom. Furthermore, there are two menu bars (4, 5) that are folded up by default and can be pulled out on demand so as not to distract the user from sketching. An additional button (3) in the top right corner activates scrolling. The drawing canvas can also be zoomed with the usual two-finger zooming gesture.

After starting FlexiSketch, the user can use the white canvas to sketch freely. The upper FlexiSketch menu bar (4) provides a list of simple geometric shapes and a palette for changing the drawing color and thickness of a stroke. Once a particular drawing step is finished and the user lifts the finger for a predefined amount of time, the drawing is converted into a distinct symbol. This symbol can then be selected by tapping (6), and a context menu appears (7). A selected symbol can be dragged around, deleted, and resized. The user can label symbols by adding text boxes using the context menu (7). For example, the stickman shown in Figure 1 is labeled *customer*. Those labels basically move together with their parent symbol, but can also move independently in order to place the label where it is needed. The context menu also allows starting lightweight metamodeling. In the *type* dialog (8), the user can define the meaning of a symbol by assigning a type. This can be done using the virtual keyboard to input a new type or by selecting one of the already defined types from a drop-down list.

Defined symbols can be re-used. As soon as the menubar (5) is expanded, it highlights all the defined symbol types. For example, Figure 1 shows that the user already defined the stickman as being of the type *actor*. Those definitions facilitate further diagram sketching in two ways. First, the user can get copies of defined symbols by dragging them from the container to the drawing canvas (instead of drawing them by hand). Second, as soon as the user draws a symbol similar to a defined one, the sketch recognition algorithm detects the symbol. It unobtrusively shows a popup box at the bottom of the screen with type suggestions for the symbol. The user can then tap on a suggestion to confirm the type. This also allows the user to replace the current drawing with the defined symbol of the same type in order to make all instances of a particular type look the same. Confirmed symbols also help to improve sketch recognition as they are taken into account for further sketch analysis.

As soon as a requirements engineer has defined all symbols needed for a particular type of diagram, it is also possible to store them permanently. The container menu (9) presents this option, and in addition, it allows the loading of defined types. Furthermore, types can be modified, particular symbols can be deleted, and new symbols can be added. This strengthens the re-use of individually created diagram types.

III. Conclusions and Future Work

Informal feedback from requirements engineers trying FlexiSketch is highly encouraging. However, also some problems emerged. Future work will address usability improvements. We also plan to extend the tool's meta-modeling capabilities. We will investigate how much meta-modeling can be done by requirements engineers, and how information relevant for metamodeling can be inferred by the tool itself. Future work will also focus on exporting and re-using models and meta-models made with FlexiSketch. This will eventually enable us to semi-automatically transcribe the documented information in order to support a more seamless integration of early RE sketches in the software development process.

References

[1] H. Ossher et al., "Flexible modeling tools for pre-requirements analysis: conceptual architecture and research challenges," in Proc. ACM Int. Conf. OOPSLA, 2010, pp. 848–864.

[2] M. D. Gross and E. Y.-L. Do, "Ambiguous intentions: a paper-like interface for creative design," in Proc. ACM Symposium on User Interface Software and Technology, 1996, pp. 183–192.

[3] S. Branham, G. Golovchinsky, S. Carter, and J. T. Biehl, "Let's go from the whiteboard: supporting transitions in work through whiteboard capture and reuse," in Proc. Int. Conf. on Human Factors in Computing Systems, 2010, pp. 75–84.

[4] D. Wüest, "Bridging the gap between requirements sketches and semi-formal models," in Doc. Symp. IEEE Int. Req. Eng. Conf. (RE), 2011. [Online]. Available: http://dx.doi.org/10.5167/uzh-55675

[5] N. Seyff, F.Graf, and N. Maiden, "Using mobile RE tools to give end-users their own voice," in Proc. IEEE Int. Req. Eng. Conf. (RE), 2010, pp. 37–46.

[6] N. Mangano, A. Baker, M. Dempsey, E. Navarro, and A. van der Hoek, "Software design sketching with Calico," in Proc. IEEE/ACM Int. Conf. on Automated Software Engineering, 2010, pp. 23–32.

[7] Q. Chen, J. Grundy, and J. Hosking, "SUMLOW: early design-stage sketching of UML diagrams on an e-whiteboard," Softw. Pract. Exper., vol. 38, no. 9, pp. 961–994, 2008.