Bridging the gap between requirements sketches and semi-formal models

Wüest, Dustin

Bridging the Gap Between Requirements Sketches and Semi-Formal Models

Dustin Wüest
Department of Informatics
University of Zurich
Switzerland
Email: wueest@ifi.uzh.ch
http://www.ifi.uzh.ch/rerg/people/wueest/

***Abstract*—State-of-the-art requirements modeling tools rely on predefined notations. In contrast, requirements engineers and stakeholders often sketch requirements in arbitrary notations during early elicitation phases. Engineers must then manually transform the sketches into semi-formal models, which is a time-consuming and error-prone task. We propose to investigate how early sketching and the transformation of sketches can be supported by a semi-automatic method that allows engineers to assign meaning to the sketches on the fly. Our tool-supported contribution is supposed to bridge the gap between sketches and semi-formal models.**

***Keywords*-requirements; elicitation; modeling; sketch-based;**

I. Motivation and Research Problem

Many software tools support requirements elicitation and modeling activities. However, they are often not used during early stages of requirements elicitation. Whiteboards, as well as pen and paper, are still the dominant tools in these stages. The reason for this is that they allow for any kind of sketches. Sketching fosters creativity [1], [2] and can also be applied by stakeholders who do not know specific modeling languages with formal syntaxes.

Stakeholders can document their ideas on a whiteboard in any form they can think of, whereas when using a software modeling tool, they have to come up with a formalized version of their ideas that correspond to the modeling language supported by the tool. The process of transforming thoughts into a writeable form imposes a cognitive load on the human mind. The bigger this load is, the more do stakeholders get distracted from the creative task of coming up with requirements. The load is minimized when using a sketching device such as a whiteboard, because stakeholders can choose any syntax to draw requirements.

However, there are disadvantages that prevent whiteboards from being the perfect tool for requirements modeling. The most important drawback is the inability to store the sketches in an appropriate format. Participants of a requirements elicitation meeting can take photographs of the whiteboard to get digital copies of the sketches. But a single image containing all requirements sketches is not a proper source for requirements management, as e.g., it does not contain requirements as individual objects.

After sketching requirements on a whiteboard, the requirements engineer has to manually re-create the sketched models from scratch in a modeling tool in order to make the requirements amenable for further processing (requirements specification and management). This tedious task constitutes a *media break*: a person has to copy information manually because it is initially stored in a form that does not comply with the form needed for further processing of the information. The recreation process is time-consuming, error-prone, and can lead to a loss of information [3]. Furthermore, some intentions behind the sketches might already be lost at that point in time, which makes the sketches prone to misinterpretations. This leads to additional errors in the recreation process.

Our planned contribution enables an online, step-wise formalization of sketches. Engineers can assign meanings to sketched symbols while drawing them. These meanings help transform the sketches into semi-formal models (e.g., a class diagram or a statechart) that are amenable to further processing. In this paper, we use the term *sketch* primarily for graph-like diagrams consisting of nodes and edges, and diagrams consisting of hierarchically nested objects as in [4].

II. Related Work

Researchers have recognized the need to overcome the media break between sketches (that do not allow for further processing) and semi-formal models (that require the use of specific modeling languages) [5]. The challenge of bridging or reducing this gap can be tackled from two directions.

On one side, augmenting formal modeling tools with sketch recognition features has led to various prototypes that allow users to sketch diagrams, e.g. [6], [7]. Sketch recognition algorithms convert the sketches into models. The drawback of this approach is that the tools still rely on predefined notations. Users have to understand a specific modeling language that is supported by a tool. Sketched symbols that do not adhere to the language cannot be interpreted by the sketch recognizer. Therefore, users are still limited in their creativity and expressiveness. Moreover, sketch recognition errors distract users from the modeling task.

On the other side, we foresee that mimicking a whiteboard and extending it with formalization capabilities has high potential, but is hard to realize. In the past years, electronic

whiteboards and large multitouch displays became affordable for more people, which makes this approach more attractive. But as users gain more freedom in sketching, and therefore the set of possible sketches increases, it becomes more difficult for a tool to analyze what the sketches actually mean. There is only few research that implemented tool prototypes where users are independent of a specific modeling language. One example is the Calico prototype [8]. It provides mechanisms to structure sketches into different parts and to connect the parts with arrows. But the sketched symbols themselves are meaningless to the tool. Other tools like MetaEdit+ [9] and MaramaSketch [10] include a metamodeling editor. The editor allows to define a custom modeling language. Metamodeling must always be performed beforehand.

The open source project Sketch for Eclipse [11], currently under development, is an API that provides sketching capabilities and a trainable gesture recognizer. End-user metamodeling is not within the scope of the project, and it is not tailored to requirements engineering.

III. Research Goal and Objectives

Our goal is to develop and evaluate a tool-supported, semi-automatic method for assigning meanings to sketches. Sketches annotated with meanings can then be translated into semi-formal diagrams, e.g., class diagrams or statecharts. The method is intended to support end-users (requirements engineers and stakeholders) in early phases of requirements elicitation and modeling. As opposed to related work, the method should allow for any type of graph-like diagram sketches, and it also should allow for seamless switching between sketching and metamodeling activities. Especially, people should be able to start sketching before any metamodeling is done.

The tool-supported method will also consider recent technological trends, particularly electronic whiteboards and large multi-touch displays. Some years ago, these techniques were still cutting-edge. Nowadays, they are available to end-users. We expect that, using the technology together with our envisioned tool in a requirements elicitation workshop, stakeholders are able to sketch their ideas, and requirements engineers are able to transform the sketches into semi-formal models.

A short example scenario shows how our envisioned tool could be used (we described more detailed scenarios in [12]):

A requirements engineer and two stakeholders use an electronic whiteboard to sketch requirements. The stakeholders are not familiar with modeling languages and do not use a specific notation. At the end of the meeting, there are rectangles, circles and arrows on the board. The engineer selects a symbol and assigns a type to it. The tool identifies similar symbols automatically. The engineer selects one of the arrows, defines it as a Connector, and defines its type as being a temporal relationship. The engineer has now a minimalistic modeling language and can define beautified symbols that represent the drawn symbols in a formalized version of the sketched model.

We define the following research objectives in order to reach our research goal.

RO 1: Define the concepts that a sketching tool needs to support in order to encourage requirements elicitation.

We need to clarify what the requirements for our envisioned tool are. Therefore we have to identify how requirements engineers and stakeholders use whiteboards to elicit and communicate requirements in early elicitation meetings.

RO 2: Develop and evaluate a technique for realizing tool-supported guidance for end-user metamodeling.

The objective is to provide a method that supports requirements engineers with automated guidance through the metamodeling steps.

RO 3: Based on results from RO 1 and RO 2, develop and evaluate a tool-supported method for the semi-automated transformation of requirements sketches into semi-formal models.

The final objective is to combine the results from RO 1 and RO 2, and to come up with a method that i) lets end-users decide what kinds of diagrams they want to sketch and ii) at the same time overcomes the media break between requirements sketches and semi-formal models.

IV. Research Questions

We want to answer the following research questions in our work. The first two questions are related to the modeling part of our method.

RQ 1.1: Does our method support the concepts that are used in diagram sketches of requirements?

With this question we ask if the method is tailored for the specific area of application. Our method is supposed to satisfy the needs of requirements engineers and stakeholders when they sketch diagrams during early requirements elicitation phases. Our method should be tailored to requirements elicitation by supporting concepts that are specific to this field.

RQ 1.2: Is it faster to create models using our tool and a big multitouch display, instead of sketching with pen and paper?

Comparing a software-supported method for diagram sketching with a pen and paper based sketching method, we expect to encounter speed benefits. This benefits are mostly achieved through editing features and a gesture based interface. In contrast, a lower drawing speed would have a big impact on the acceptance of our method.

The next three questions concern the feasibility of leaving a part of the metamodeling task to the end-users.

RQ 2.1: Does our method provide guidance for metamodeling in such a way that end-users can generate metamodels without the help of experts?

The answer to this question is very important for our research. A key part of our research will be to identify appropriate ways of giving metamodeling guidance to end-users. Especially, we want to explore whether it is possible to have an interface that does not require any programming or scripting for the definition of the metamodel.

RQ 2.2: Is our method for guided metamodeling useful for early requirements elicitation?

With our method we expect to avoid media breaks between requirements sketches and semi-formal models. This only

succeeds if the method provides good usability and utility so that end-users use the formalization features of our tool.

RQ 2.3: What are the limits of our method concerning the scalability and complexity of different diagram types?

Since we want to give end-users the flexibility to draw any kind of diagrams, we have to ask ourselves whether our method allows for the formalization of any sketched diagrams, or if end-users are limited to certain kinds of diagrams. Also, metamodeling can be a hard task even for metamodeling experts. The complexity of metamodeling increases with the complexity of the metamodel. Our method is probably limited to lightweight modeling languages (languages with little formal expressive power that consist of few different elements [4]). This is not necessarily a big limitation, as stakeholders and requirements engineers primarily use lightweight notations during early requirements elicitation stages [4]. The creativity gets hindered if more complex modeling languages are used.

V. Research Methodology and Validation

As a first step, we carried out a **systematic literature review** on the use of whiteboards and sketch-based tools for requirements elicitation. As described in the motivation and related work sections, we concluded that the gap between informal requirements sketches and semi-formal models is not yet bridged by current approaches. We then conducted a **domain analysis** and had discussions about our research ideas with RE experts from research and industry, which resulted in first evidence about the usefulness of our envisioned tool, and a list of characteristics such a tool should have (see Section VI-A). Additionally, we have the opportunity to exploit experience of a Swiss company that is specialized in RE. The company is heavily using sketches for requirements elicitation, and provides us with some insight into their elicitation process and the resulting diagram sketches. Together with the literature review, this addresses RO 1.

We came up with a **conceptual solution** (see Section VI-B) and are currently working on basic tool support. In parallel, we are performing **literature reviews** on HCI, metamodeling approaches and sketch recognition frameworks. This reviews help to address RO 2. Findings from the HCI field are important for the user interface of our envisioned prototype. Usability plays a central role for tool acceptance and the time needed to sketch requirements. Theory about the mapping between metamodels helps us in creating a method for the translation of requirements sketches into semi-formal models. Sketch recognition is not in the focus of our research, but is needed in our prototype. Therefore we search for an adequate existing recognition framework.

A subsequent analysis and selection of the identified methods in the literature will provide a basis for the design and development of a method for guided metamodeling. Together with the results from RO 1, we will come up with a more comprehensive tool prototype for diagram sketching and the transformation of the sketches into semi-formal models (RO 3). The prototype will be a software solution that is compatible with electronic whiteboards and multitouch screens. We will develop the prototype from scratch, but use a multitouch API and adapt an existing sketch recognition framework.

For the evaluation of the methods developed under RO 2 and RO 3, the first step will consist of two **controlled experiments** with students. We will also measure the time needed for sketching (RQ 2), and assess the comprehensibility of pen and paper sketches versus sketches augmented with metamodel artifacts. The experiments will be complemented by **industrial case studies**, where we will be able to test our prototype in the field. This will reduce the threat to external validity that is introduced when performing experiments with students.

We will perform our research in an iterative manner, rather than going through the described steps sequentially. This will help us to have an early prototype that we can improve over time. Especially, we soon have to evaluate if end-users can indeed sketch symbols and assign types to them, as this is a core part of our tool-supported method.

VI. Preliminary Results

The discussions mentioned in Section V led to a list of high-level requirements for our envisioned prototype. First, we describe these requirements. Then we give an overview of our ideas for our solution approach.

A. Requirements for a Tool-Supported Method

As a first result based on discussions with experts and the study of related work, we expect an optimal solution to satisfy the following requirements:

- High flexibility: allow the users to sketch any diagrams.
- Formalization capabilities: support the transformation of diagram sketches into classic semi-formal models by a semi-automated method.
- Natural sketching: allow the use of an input device that gives a natural feeling for sketching.
- Speed: reduce the time needed to sketch requirements.

We expect these characteristics to bring the following benefits for the end-users of our envisioned tool:

The first characteristic encourages creativity and makes requirements elicitation accessible to stakeholders who do not master specific modeling languages. It does justice to the fact that most current modeling tools are not flexible enough to handle different kinds of diagrams.

The second characteristic avoids media breaks. Sketched diagrams do not need to be manually re-created as semi-formal models; they *evolve* into semi-formal models. This overcomes the media breaks between free-form sketching and formal modeling of requirements. Avoiding media breaks can lead to less misunderstandings and models of better quality.

The third characteristic asks for an appropriate interface on the hardware side. Such an input device could be an electronic whiteboard with a bunch of pens, or a large multitouch display. It allows the users to draw in a way they are used to since their childhood. Furthermore, collaboration is facilitated when no physical input device needs to be passed on to a user before the user can start sketching.

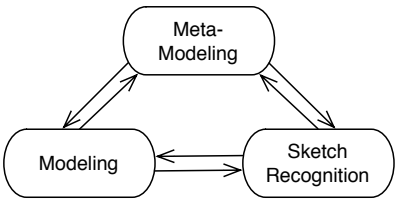

Fig. 1. The three activities leading to a semi-formal model in the end.

The fourth characteristic makes use of software-supported modeling. The use of various input gestures and auto-completion features can speed up the modeling process.

B. Conceptual Solution

The big challenge is to come up with a tool that actually understands what users sketch. We think that users should have as much flexibility as possible when sketching diagrams. We do not want to restrict users to any specific languages or notations. Therefore it is impossible for a tool to understand the sketches automatically. As a consequence, we use a semi-automated method for formalizing sketches. Users have to put meaning into their sketches manually. No programming or scripting should be required for this task.

The main idea of our tool-supported method is to let users decide *when* to put meaning into their sketches. Especially, it should be possible for users to first start sketching and assign syntax and semantics later (on demand). This is a key difference between our approach and related work, where any form of metamodeling has to be done beforehand.

We identified three main activities and a workflow that our method must support in order to fulfill our research goal, as depicted in Fig. 1. The workflow contains of arbitrary interleaving between the three activities **modeling**, **meta-modeling**, and **sketch recognition**. We have already discussed these activities in more detail in [12].

Two drawing modes will facilitate the **modeling** activity. One mode mimics a whiteboard and allows for free sketching, while the other mode enables users to modify individual objects (e.g. scale, move, copy).

The **metamodeling** activity starts when users assign a type to one of the sketched symbols. The symbol then gets added automatically to a *dynamic symbol library*. The symbol library consists of all defined symbols including their types. The library gets implicitly modified each time when users tag a sketched symbol with a type. A big challenge will be to invent an easy-to-use interface for more complex metalanguage definitions, such as associations and rules for associations. When switching back to modeling, the tool must not insist that all sketches conform to the metalanguage defined so far, but it must provide feedback about violations.

Sketch recognition is done by the tool, but the user is able to decide when recognition should take place, and if feedback about the recognition should be given instantly or not. The options assure that users do not get distracted from requirements elicitation. This idea conforms to a study [13] that investigated different recognition feedback strategies.

VII. Conclusion

In this paper we propose a method for a seamless integration of sketching, modeling, and metamodeling into a single environment in order to support the transition from ideas to models. End-users will have the flexibility of just start sketching diagrams and assign meaning to the sketches on the fly. Putting meaning into the sketches enables their transformation into semi-formal models. Therefore we foresee that our method will overcome the media break between requirements sketches and semi-formal models. We also expect that requirements sketches will look clearer and will be easier to understand: because of the meaning that is put behind the sketched objects, sketches are less ambiguous and can be beautified automatically. Furthermore, it will be easy to change and extend stored sketches.

After conducting a literature review on sketch-based tools for requirements elicitation, and a domain analysis that led to first ideas and findings, we are currently working on the first tool prototype and start to plan the evaluation.

References

[1] V. Goel, "Sketches of thought: a study of the role of sketching in design problem-solving and its implications for the computational theory of the mind," Ph.D. dissertation, University of California at Berkeley, CA, USA, 1991.

[2] A. Black, "Visible planning on paper and on screen: the impact of working medium on decision-making by novice graphic designers," *Behavior and Information Technology*, vol. 9, no. 4, pp. 283–296, 1990.

[3] M. Cherubini, G. Venolia, R. DeLine, and A. J. Ko, "Let's go to the whiteboard: how and why software developers use drawings," in *Proc. SIGCHI Conference on Human Factors in Computing Systems*, 2007, pp. 557–566.

[4] M. Glinz, "Very lightweight requirements modeling," in *Proc. 18th IEEE International Requirements Engineering Conference*, 2010, pp. 385–386.

[5] H. Ossher, A. van der Hoek, M.-A. Storey, J. Grundy, and R. Bellamy, "Workshop on Flexible Modeling Tools (FlexiTools2010)," in *Proc. 32nd ACM/IEEE Int. Conf. on Software Engineering*, 2010, pp. 441–442.

[6] 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.

[7] T. Hammond and R. Davis, "Tahuti: a geometrical sketch recognition system for UML class diagrams," in *AAAI Spring Symposium on Sketch Understanding*, 2002, pp. 59–68.

[8] 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.

[9] S. Kelly, K. Lyytinen, and M. Rossi, "Metaedit+: A fully configurable multi-user and multi-tool CASE and CAME environment," in *Advanced Information Systems Engineering*, Lecture Notes in Computer Science, P. Constantopoulos, J. Mylopoulos, and Y. Vassiliou, Eds. Springer Berlin / Heidelberg, 1996, vol. 1080, pp. 1–21.

[10] J. Grundy and J. Hosking, "Supporting generic sketching-based input of diagrams in a domain-specific visual language meta-tool," in *Proc. 29th Int. Conf. on Software Engineering*, 2007, pp. 282–291.

[11] U. B. Sangiorgi and S. D. Barbosa, "Sketch: modeling using freehand drawing in Eclipse graphical editors," in *ICSE 2010 Workshop on Flexible Modeling Tools*, Cape Town, South Africa, 2010. [Online]. Available: http://www.ics.uci.edu/ tproenca/icse2010/flexitools/papers/10.pdf

[12] D. Wüest and M. Glinz, "Flexible sketch-based requirements modeling," in *Requirements Engineering: Foundation for Software Quality*, Lecture Notes in Computer Science, D. Berry and X. Franch, Eds. Springer Berlin / Heidelberg, 2011, vol. 6606, pp. 100–105.

[13] P. Wais, A. Wolin, and C. Alvarado, "Designing a sketch recognition front-end: user perception of interface elements," in *Proc. 4th Eurographics Workshop on Sketch-Based Interfaces and Modeling*, 2007, pp. 99–106.