Scribbler – Drawing Models in a Creative and Collaborative Environment: from Hand-Drawn Sketches to Domain Specific Models and Vice Versa

Martin Vogel, Tim Warnecke, Christian Bartelt, Andreas Rausch
Clausthal University of Technology
Department of Informatics - Software Systems Engineering
Julius-Albert-Str. 4
38678 Clausthal-Zellerfeld, Germany
{m.vogel, tim.warnecke, christian.bartelt, andreas.rausch}@tu-clausthal.de

Abstract

In the early phases, software engineers use whiteboards and flip charts to create and discuss their ideas and later they transform manually the hand drawn pictures into machine readable models. During this transformation important sketch information, like the history of origin or some elements, will be lost. To solve this problem, we present a new approach using digital whiteboards to elaborate in a creative and collaborative environment hand drawn pictures and transform them into domain specific models and vice versa. This poster outlines the process of the automatic transformation from sketch models to models based on well-defined notations and vice versa in the early creative phases of software development.

Video: https://www.youtube.com/watch?v=0i3M9djPrRM

Keywords: Sketch Recognition, Model-Based Software Development, Collaborative Software Engineering

1 Introduction and motivation

These days software development is a distributed team process. The commercial success of products is increasingly important and the quality of the software has an important role in this context(Friedewald et al. 2001). However, less than 30% of all software projects are successful (Hartmann 2006). Especially critical are the early phases of the requirements analysis and the architectural design. Studies show, that the correction of errors from these early phases will cost up to 14 times as much as in case of errors in later phases (Westland 2002). Requirements analysis and architectural design are particularly important for the success of development projects. These early phases are also the creative phases. It is crucial for the success of a project, that teams work closely together – occasionally at different locations - and have strong cooperation. Thereby, handwritten sketches on whiteboards have been generally accepted in practice as the medium of communication. About 75% of developers like to use whiteboards during analysis and design (Cherubini et al. 2007) – even and especially in team workshops.

Normally software engineers do not use modeling tools like MagicDraw UML (MagicDraw 2013) in this early phase, because of their cumbersome handling. Modeling tools are made for precise model documentation, and not for creative sketching. Hence software engineers use whiteboards and flip charts in team meetings to create and discuss their ideas in form of various hand drawn pictures using well defined notations, like for instance, UML diagrams or Flow Chart diagrams. The existing (and partly expensive) modelling tools are used later on, to manually transform the hand drawn pictures into machine readable models for the next development tasks, like for instance, code generation. In follow-up meetings software engineers use the old hand drawn pictures and print-outs of the afterwards documented and further developed models to discuss and further improve their solutions. Thereby, they again produce additional hand-drawn pictures which again have to be later on manually integrated into the existing machine-readable models. Another topic is that some team members will not recognize the shown model, because in the transformation some important information is lost or a team member might try to optimize the resulting sketch model in the modeling tool.

2 Scribbler: A model sketching environment

To support software engineers in the early phases, we developed a collaborative and creative modeling tool called Scribbler. Scribbler uses digital whiteboards to transform free hand sketches in models based on well-defined notations and back again, saving the history of origins, recognizing objects and handwriting.

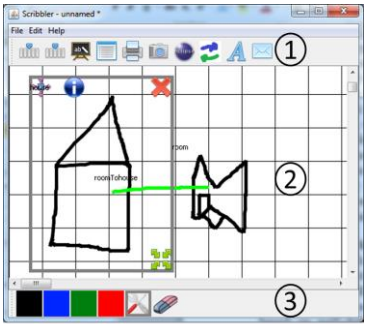

Figure 1: The Scribbler tool

Scribbler is portable for every device which uses java and supports ad-hoc meetings between geographically distributed locations. The transformation is completely

independent from a pre-defined modeling language. The tool provides a training mode to learn new graphical syntax elements and map these to formal meta models entities. Furthermore developers can quickly create new pluggable components for their own requirements (Bartelt et al. 2013). A screenshot of the tool is shown in Figure 1. At the top (1), all current loaded plugins (e.g. learning) are represented with an icon, in the center (2) the canvas and last but not least the toolbar is located at the bottom (3), which consists of four colors, an edit button and a rubber.

3 Transforming sketch models to models based on well-defined notations and back again

Transforming a hand drawn sketch to a formal based model, describes a process, of mapping sketches to elements of the given meta model. Figure 2 shows a small Ecore (EMF) meta model.

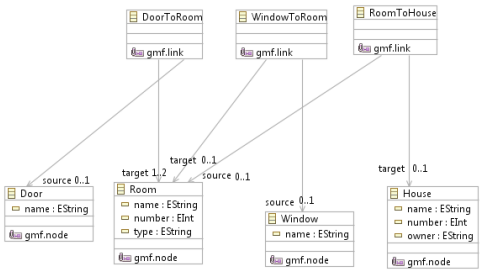

Figure 2: Ecore meta model

This meta model consists of entities like houses, rooms, windows and doors, and connections between this elements. In this example DoorToRoom, WindowToRoom and RoomToHouse are connections represented as lines. Room, Door, Window and House are represented by meaningful icons.

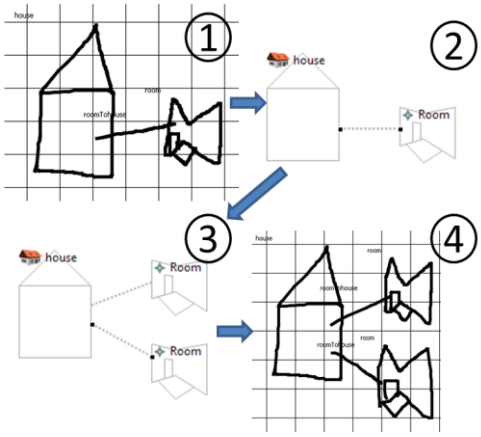

Figure 3: Transforming sketches into a model and back again

As mentioned above, the process of the transformation from sketch models to models based on well-defined notations is error-prone. Almost always the sketch information, like the history of origin, is lost and the formal model looks different to the sketch. To fix this problem, Scribbler can export sketched models to models based on well-defined notations which are generated with EMF – the Eclipse Modeling Framework - without losing sketch information. Furthermore, Scribbler can import these models from the EMF-Editor. Figure 3 shows an example of the import/export process of a sketched model. In **phase 1,** the team created a sketch model. After the meeting Scribbler can transform automatically the sketch model to a model based on well-defined notations. Thereto, Scribbler detects collision and inclusion, maps the recognized sketch objects to concrete elements of the Ecore meta model, stores the sketch model into a special tag and stores it into a file. **Phase 2** displays the formal model which is the result of the transformation. It is easy to see that the elements are at the same position and have roughly the same size like in the sketch model in. After the transformation of a sketch model to a formal model, the user is able to work on the model in the editor. In this example a user changes the position of the room and adds a new room. The modifications in the editor can be seen in **phase 3**. After that, the formal model can be imported in Scribbler. During the import Scribbler verifies the information of the formal model file. If there are valid sketch information Scribbler draws these elements on the canvas. If the position or size has changed, Scribbler adapts the coordinates. In this example a new room was added in the GMF-Editor. The application checks to which element in the knowledge base room is mapped and draws the room with this information. **Phase 4** shows the result of the import from the formal model file. Scribbler allows distributed, parallel (collaborative) sketching of models on digital whiteboards and tablet pcs, the transformation of those sketches in model based on well-defined notations and back again, recognize objects of every domain, recognize hand writing, an easy and interactive learning of new domain specific syntax elements and a recording/playback of the modeling/sketching history.

4 References

Bartelt, C., Vogel, M. & Warnecke, T., 2013. Collaborative Creativity: From Hand Drawn Sketches to Formal Domain Specific Models and Back Again. In A. Nolte et al., eds. *MoRoCo@ECSCW*. CEUR Workshop Proceedings. CEUR-WS.org, pp. 25–32. dblp.uni-trier.de/db/conf/ecscw/moroco2013.html#BarteltVW13.

Cherubini, M. et al., 2007. Let's go to the whiteboard: how and why software developers use drawings. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '07. New York, NY, USA: ACM, pp. 557–566.

Friedewald, M. et al., 2001. Softwareentwicklung in Deutschland: Eine Bestandsaufnahme. *Informatik Spektrum*, 24(2), pp.81–90.

Hartmann, D., 2006. Interview: Jim Johnson of the Standish Group. Available at: http://www.infoq.com/articles/Interview-Johnson-Standish-CHAOS [Acc. Nov 3, 2013].

MagicDraw, 2013. NoMagic - MagicDraw. Available at: www.nomagic.com/products/magicdraw.html [Acc. Nov 3, 2013].

Westland, J.C., 2002. The cost of errors in software development: evidence from industry. *J. Syst. Softw.*, 62(1), pp.1–9.