In: Nolte, A., Prilla, M., Rittgen, P. and Oppl, S.: Proceedings of the International Workshop
on Models and their Role in Collaboration at the ECSCW 2013 (MoRoCo 2013)

Collaborative Creativity: From Hand Drawn Sketches to Formal Domain Specific Models and Back Again

Christian Bartelt, Martin Vogel, Tim Warnecke
Software Systems Engineering - Department for Computer Science
University of Clausthal, Germany
{christian.bartelt, m.vogel, tim.warnecke}@tu-clausthal.de

Abstract. Most of the time developers make extensive use of software tools in a software development process to support them in their day-to-day work. One of the first and most important phases is the design phase. Here tools are missing which support the creative and collaborative workflow (parallel/distributed). At the moment software designers uses classic whiteboards in team meetings to express their ideas. Subsequently a coworker uses a mobile phone or a camera to take photos of the work and remodel the picture with a modeling tool. That process is very inconvenient, error-prone and hinders a creative modeling cycle. For overcoming this ineffective process this paper shows a new approach to use digital whiteboards to transform free hand sketches in formal models and back again while modeling in a distributed team. The approach is completely independent from a pre-defined modeling language. It provides an interactive training mode to learn new graphical syntax elements and map these elements to formal meta-model entities. Based on the approach a collaborative sketch and modeling infrastructure was implemented.

Video: https://www.youtube.com/watch?v=0i3M9djPrRM
[Mirror: http://sse-world.de/index.php?cID=3611]

Motivation and State-of-the-Art

Nowadays, software development is a creative and distributed team process. The early and creative design phase is very important for successful software projects.

Normally software designers do not use modeling tools like MagicDraw UML (MagicDraw 2013) in this early phase because of their inconvenient operation. Modeling tools are made for precise model design, but not for creative sketching. That is the reason why software designers using whiteboards in team meetings to visualize and communicate their ideas within the group (Cherubini et al. 2007). Nevertheless, after a successful team meeting, one of the designers has to transform the drawn sketches in formal models using the previously rejected modeling tools. Those transformations are error-prone because for e.g. the designer tries to optimize the diagram or forgets some elements. Thus it is possible that the new formal diagram cannot be recognized by the other team members in a later meeting because of its changed appearance.

Another widely known problem in this domain describes that everybody has to be present to accomplish such a creative meeting. If a team is spread all over the country or worse over the world it needs a lot of effort, money, and time to get them all to one place. One possible solution is to use Screen Sharing Software like TeamViewer (TeamViewer 2013) to share some kind of drawing software and additionally utilize a telephone conference system. An issue here is that two or more software applications are used together and none of them is particular conceptualized for software designers.

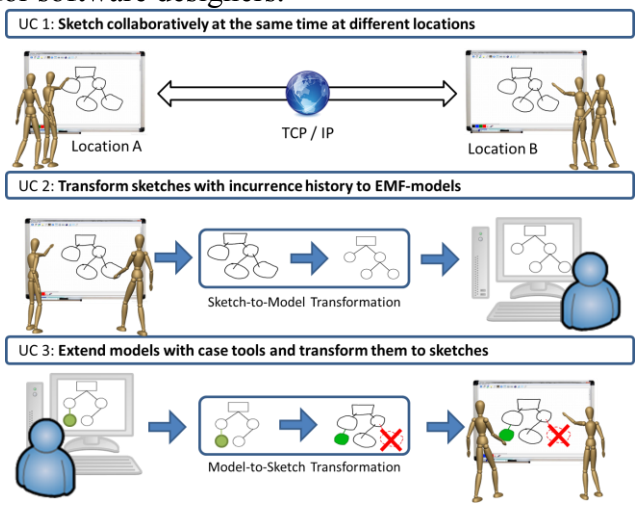

Figure 1: Use cases

In Figure 1 three typical modeling use cases in the creative engineering phase are depicted. The first use case (UC 1) describes two distributed teams are working parallel on the same sketch model. UC 2 shows how a sketched model, which was drawn by a team of designers and transformed to a formal domain specific model. This transformed model is based on Ecore, which is now usable with a corresponding EMF- or GMF-Editor (whereas the whole history of origins of the sketch is preserved). This allows a single user to extend and change the model based on the results and feedback of the previous team meeting. UC 3 illustrates how such a modified model is transformed back to a sketched model.

Because Scribbler knows how the sketched model looked before all changes made in the GMF-Editor can be visualized, e.g. an element was deleted or added.

The last both use cases make it possible to establish a creative life cycle which starts with a onetime configuration. This configuration consists of a knowledge base, which contains of previously learned sketches for the chosen DSL, and a mapping between those sketches and elements of the DSL meta model (Ecore). It should be noted, that this configuration can also be done after drawing sketches. After the team drew some sketches the sketch model is transformed, with help of the configuration, to a formal model (UC 2) which is suitable for a GMF-editor. In this editor a single user modifies the model based on the feedback of the meeting before. After he finished his modifications, the model is transformed back to a sketch model and given back to the team (UC 3). All modifications done in the GMF-Editor are visualized and the remaining, but unchanged, sketches look like in the first meeting. Thus the team is able to better recognize what happened in the last meeting. To improve the process of recognizing a sketch, which is made some days or weeks before, the designer can also use the implemented history viewer which recorded every stroke made on the canvas.

With similarities to these use cases in the last few years several research initiatives are started with the topic intuitive modeling respectively model sketching. In (Sangiorgi & Barbosa 2010) a recognition mechanism for sketched model elements was presented. This mechanism uses a similarity calculation between drawing traces based on the Levenshtein distance (Levenshtein 1966) and is also a foundation for the research here explained. Some innovative research results about sketch recognition in the area of requirements modeling was described in (Wüest, Seyff, and Glinz 2013). Some further recognition techniques based on vector comparison between sketches and GEF/GMF model elements were published in (Scharf 2013).

Scribbler – The Collaborative Sketching/Modeling Infrastructure for Domain Specific Models

The developed sketching/modeling platform Scribbler picks up the upon explained use cases. Therefore, it must overcome several challenges like sketch recognition, formal model synthesis from recognized sketches, sketch synthesis from formal model, and an efficient mechanism to allow distributed parallel sketching. Furthermore, the approach should be easy/user friendly adaptable for sketching any kind of domain specific syntax.

Sketch Recognition and Knowledgebase

A sketch is per definition a freehand drawing, consisting of some individual elements, which is not yet finished and tries to transport some kind of idea (Davies 1990). Following from this a sketch and its elements has meanings for the people who work with it, but for computers sketch elements are only a set of x- and y-coordinates and maybe colours. These coordinates must be interpreted in a way such the computer knows what the drawing person had in mind. Aggravated by the fact that every human being sketches figures a little bit different, four main problems must be taken into account when recognizing a figure. First the drawing order of a figure alters between different users, second the size of a figure changes with attempt, third a figure can be inclined to the right or left side and last users often don't draw solid lines.

To solve these problems, Scribbler uses an extended and modified version of an algorithm described by (Coyette et al. 2007). In a first step of the procedure a sequence of numbers based on the intersections with a predefined grid is produced. In the example in Figure 2 the sequence of the circle is *1 1 2 3 4 5 8 9 12 13 14 15* because two intersections are detected in field 1, one additional intersection in field 2, one additional intersection in field 3 and so on.

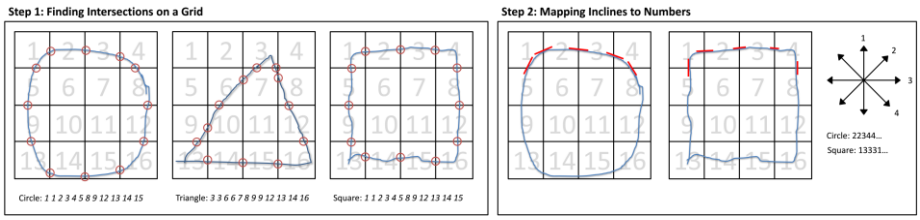

Figure 2: Sketch Recognition

This is pretty straight forward, but not every drawn object has a unique numerical sequence. A circle could have the same sequence as a square because they have the same intersections with the used grid. In a second step the incline for every intersection of the drawn line is measured and mapped to a number corresponding to the scheme shown on the right side of Figure 2. This generates a second numerical sequence for both sketches, which is now completely different.

After constructing such a pair of sequences a knowledgebase of previously drawn figures is needed to compare them with new drawn sketches and finding a match using Levenshtein distance (Levenshtein 1966). Building such a knowledgebase needs some kind of learning environment, which is done in Scribbler with an own dialog.

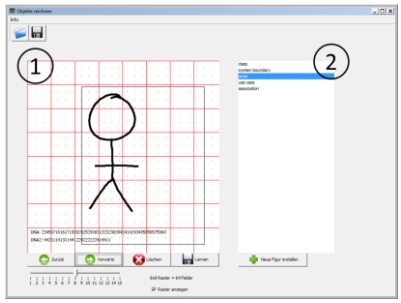

Figure 3: Learning environment

This dialog is shown in Figure 3 and consists of a training canvas (1) and a list of elements (2) which were already learned. The user is also able to add new versions of a sketch by redrawing it over and over (leads to better recognition results) again and to create new sketches by adding them to the list. Every drawn sketch will be automatically added to the knowledgebase. The knowledgebase is stored in a file, which can be exchanged with other users. For collaborative work recognizing of sketches is important, because the team gets a visual feedback of what happened. If the feedback is not that result which they actually discussed, the team can react immediately and change the type of the sketch to the desired one.

From Sketches to Formal Models and Back Again

Transforming a hand drawn sketch to a formal Ecore (EMF 2013) based model describes a process of mapping sketches to elements of the given meta model. At first glance this sounds easy but a lot of information, like for example position, size and the history of origins, is lost if hand drawn model elements are just mapped to their corresponding EMF counterparts. Transforming the EMF model back to a hand drawn sketch is not possible any more without theses information.

Due to this problem a new file type was needed. In this file three types of information are stored at the same time for every element. The first is the meaning of an element defined by the corresponding meta model. The second is the graphical representation given by the Graphical Modeling Framework (GMF 2013) such as, for example a UML class is a rectangle with an additional horizontal stroke. The last type of information stored the coordinates of the hand drawn strokes and the history of origins logged by Scribbler. This new file type sets Scribbler in the position to transform hand drawn sketches to ECore based models and back again without loss of information.

Collaboration: Drawing together and saving the history of origins

Another problem with hand drawn sketches is that the incurrence of its elements is not traceable anymore. This is especially problematic if a sketch was drawn some days or weeks ago and someone tries to remember what happened in the

meeting (Cherubini et al. 2007). Scribbler solves the problem by saving all drawing actions which happened on the canvas in a file for later use. The core of it fires for every user action an event, e.g. drawing or moving. Events are stored in a queue for the plugins. This queue is stored in a file with the whole sketch model. Thus, the history of origins still remains. For the transformation from sketches in formal models, the history is also stored in the new model file. After this step it is possible to review the whole drawing process of the sketch in a user defined speed and to stop and restart it whenever the user wants to.

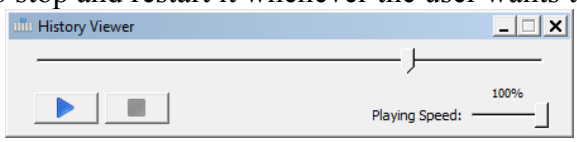

Figure 4: History viewer

Figure 4 shows the user interface of the history viewer. It looks like an audio/video tool with a play button and a slider for the timeline. Thus, the history of origins of the model can play back. This feature is important for collaborative modeling, because if a new member joins the team, he can comprehend how the model is originated in the team. He can jump to every position in the origination process. Further the history viewer might be used in future for contextual modeling. This means that the team navigates to the position, which they want to modify and change the information in the sketch. After the modification they navigate to the end of the timeline and continue the work at the model. The modification saved in the history of origins.

Another component of Scribbler is the collaboration platform. Since Scribbler is an intuitive modelling tool, which is inspired by a standard whiteboard, it is necessary to construct a collaborative lock-free environment in which everyone immediately sees if a user starts a new sketch, how he draws it and the name of the user. Implementing such a collaborative environment is a challenge because every client doesn't use necessarily the same hardware and software. This fact leads to two new problems. Different devices have different screen resolutions and bandwidths.

Solving problem number one, Scribbler scales up every drawn sketch to a fictional resolution and scales them down to the actual resolution of the corresponding device. This procedure ensures that every screen size is supported no matter how big or small it is. Problem number two is solved by transferring only mouse movements/events and the coordinates, so no screen sharing is necessary. Furthermore, the server saves every draw session. This feature allows sending all transmitted data to a new connecting client and pushing him to the current stage of work. Additionally the server has the ability to save the cached data of a session in a local file. Thus it is possible to load and continue such an older session or view it in the history viewer. Thereby every member of a team can prepare for the next meeting or evaluate the session afterwards.

Tool Implementation

Scribbler is just a simple paint program which supports only basic operations like draw, move, scale, delete, save and load. Every drawn sketch consists of a series of raw dates, like, for example, coordinates and mouse movements. Scribbler gains its sketch recognition and collaboration skills through plugins.

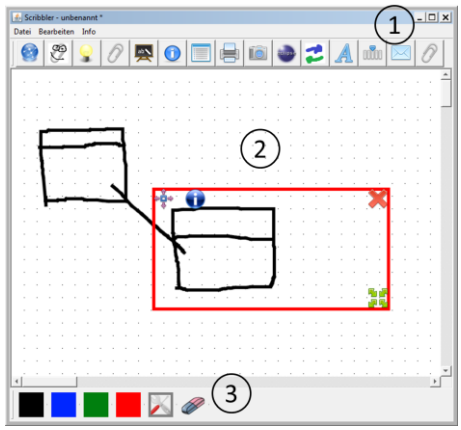

Figure 5: Tool Scribbler

A screenshot of the tool is shown in Figure 5. At the top of it (1) all current loaded plugins are represented with an icon. In the center (2) is 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.

Evaluation

During the project duration three industry partners from different domains used the Scribbler for their daily work – with customers and for architectural and structure meetings - for about four weeks. Every team had an experimental setup composed of a digital whiteboard and some tablet pcs and a catalogue of questions to evaluate the Scribbler. The results of the evaluation as described below. The three teams used the Scribbler only for collaborative work – especially the history viewer and the server sessions - to prepare the next meeting. The teams assessed this functionality as valuable and it is very helpful for their daily work. Furthermore, they used the learning environment to insert their own elements for their own domain languages. Scribbler was able to learn all of these elements and the recognition rate was very good. Concerning the usability and the training period every participant rated the Scribbler as good.

Conclusion

Ensuing from the requirements regarding an intuitive modeling infrastructure that does not hinders the creative engineering process the sketching/modeling platform Scribbler was developed. Scribblers allows a distributed, parallel (collaborative) sketching of engineering models on digital whiteboards, the transformation of sketches in (semi-)formal domain specific models and back again, an easy and interactive learning of new domain specific syntax elements, and a recording/playback of the modeling/sketching history. The fundamental concepts of all these features are explained in this paper. Furthermore the implemented software infrastructure is presented. For future work the recording of further context information during the sketching modeling process (e.g. voices of modelers within the history of model evolution etc.) is planned.

This research work was supported by “German Federal Ministry of Education and Research” (BMBF) within the Project “KoMo – From Sketch to Model: Cooperative Modeling with Domain Specific Languages” (2011-2013).

References

Cherubini, Mauro 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*, 557–566. CHI ’07. New York, NY, USA: ACM. doi:10.1145/1240624.1240714. http://doi.acm.org/10.1145/1240624.1240714.

Coyette, Adrien et al. 2007. «Trainable sketch recognizer for graphical user interface design» in . *Proceedings of the 11th IFIP TC 13 international conference on Human-computer interaction*, 124–135. INTERACT’07. Berlin, Heidelberg: Springer-Verlag. http://dl.acm.org/citation.cfm?id=1776994.1777013.

Davies, Diana. 1990. *Harrap’s Illustrated Dictionary of Art and Artists*. Chambers.

EMF. 2013. «EMF». www.eclipse.org/modeling/emf/.

GMF. 2013. «GMF». www.eclipse.org/modeling/gmp/.

Levenshtein, Vladimir I. 1966. «Binary Codes Capable of Correcting Deletions, Insertions and Reversals» in . *Soviet Physics Doklady*, 10:707 – 710.

MagicDraw. 2013. «NoMagic - MagicDraw». www.nomagic.com/products/magicdraw.html.

Sangiorgi, Ugo Braga & Barbosa, Simone D. J. 2010. «SKETCH: Modeling Using Freehand Drawing in Eclipse Graphical Editors» in . *Proc. FlexiTools Workshop*. doi:http://www.ics.uci.edu/ tproenca/icse2010/flexitools/papers/10.pdf.

Scharf, Andreas. 2013. «Scribble - A Framework for Integrating Intelligent Input Methods into Graphical Diagram Editors» in . *Software Engineering 2013 Workshopband (inkl. Doktorandensymposium)*, 591–596. http://www.se2013.rwth-aachen.de/downloads/proceedings/SE2013WS.pdf.

TeamViewer. 2013. «TeamViewer». www.teamviewer.com.

Wüest, Dustin; Seyff, Norbert & Glinz, Martin. 2013. «FlexiSketch: A Mobile Sketching Tool for Software Modeling» in Uhler, David et al. (arg.). *Mobile Computing, Applications, and Services*, 225–244. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 110. Springer Berlin Heidelberg. http://link.springer.com/chapter/10.1007/978-3-642-36632-1_13.