Towards Tool Support For Agile Modeling: Sketching Equals Modeling

Thomas Buchmann
Chair of Applied Computer Science I, University of Bayreuth
Bayreuth, Germany
thomas.buchmann@uni-bayreuth.de

ABSTRACT

Model-driven development is a well-known practice in modern software engineering. A wide variety of different tools exist, which support model-driven development. Usually, these tools do not provide dedicated support for agile modeling, as they can be used with any development process. In this paper, we present an extension to our UML-based modeling tool Valkyrie which allows free-hand diagram sketching. Thus, it addresses agile modeling as whiteboards and papers can now be replaced with tablet computers or other touch-enabled (hand-held) devices.

Categories and Subject Descriptors

D.2.8 [**Software**]: Software Engineering—*model-driven development, agile modeling, eclipse, android, sketching*

General Terms

Model-driven software engineering

1. INTRODUCTION

Model-driven software engineering is a discipline which evolved during the last decade. A wide variety of different tools exist, which support the modeler during the development process. Since model-driven software development is not tied to a special software development methodology, these tools usually can be used with any development process.

Agile model driven development (AMDD) [1] applies commonly known principles and practices from traditional source code based agile software development to model-driven development. In his book [2], Scott W. Ambler states, that agile modeling (AM) asks to use the simplest tools possible (e.g. papers and whiteboards). Complex modeling tools should only be used, when they provide the best value possible.

However, sketches drawn on papers or whiteboards have to be distributed to all involved team members, e.g. by scanning or photographing the result. This raises several problems. While in source code based approaches, where diagrams are only used for documentation purposes, a photograph of a whiteboard sketch might be enough, model-driven approaches demand for models as first class entities. Thus, every diagram that has been sketched on a whiteboard or on a piece of paper has to be redone in the respective modeling tool, which results in an additional overhead. Furthermore, sketches on whiteboards or papers are often missing some essential details like role names or cardinalities of associations for example. Usually these errors are fixed at a later time, when the sketch is redone with the respective modeling tool.

On the other hand, we observe an increasing popularity of touch-enabled devices. Smartphones nowadays have even more processing power than desktop computers had a few years ago. Modern devices are also equipped with HDMI video output and can therefore be easily attached to big TV screens or beamers. This motivated us to add sketching capabilities to our UML-based modeling environment called *Valkyrie* [5]. This approach provides several advantages: (1) a device running our tool might replace papers or whiteboards in agile modeling processes as sketches directly result in corresponding model instances. No manual redrawing of whiteboard sketches in a modeling tool is required. (2) Since the tool follows common UML standards, inconsistencies like missing role names or cardinalities are immediately reported to the user and thus can be fixed right away.

The paper is structured as follows: Section 2 gives a detailed overview about our Valkyrie environment and also addresses the free-hand drawing support as well as our work on porting the approach to Android devices. Related work is discussed in section 3 while section 4 concludes the paper.

2. TOOL SUPPORT

2.1 Overview

In this section, we give a brief conceptual overview about our UML modeling tool *Valkyrie* and how it is designed. Furthermore, we describe the model-driven tools and frameworks that were used during the development process.

Figure 1 shows an overview about the diagrams currently supported by Valkyrie and their usage in the different phases of the software engineering process. Use case diagrams and activity diagrams are used during requirements engineering. In that case, activity diagrams serve as a formalism to further detail single use cases. Analysis and design is supported through package diagrams, class diagrams, object diagrams, activity diagrams and statecharts respectively. Furthermore, we are currently working on a support for UML Action Language Foundation (ALF) [14]. ALF will be used in Valkyrie to specify the behavior of operations defined in the class

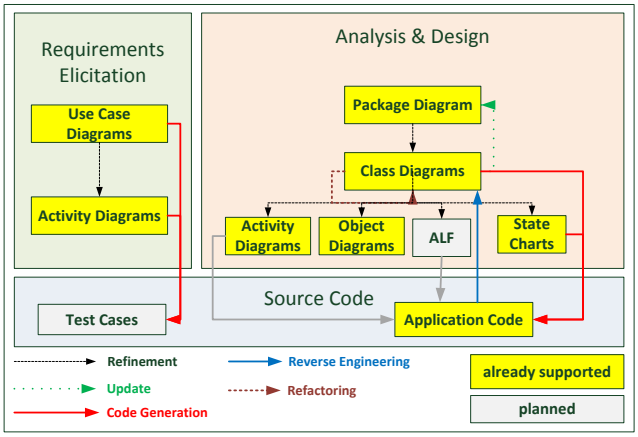

Figure 1: Valkyrie's diagrams and relations.

diagram. Furthermore, classes may be refined by statecharts defining a protocol state machine. In that case, the statechart defines valid states of a class. The transitions defined in the statechart can be called from operation implementations.

Valkyrie itself has been developed in a highly model-driven way. Figure 2 depicts the architecture of Valkyrie. Using the Eclipse UML2 metamodel [10] (which is based on EMF [17]) offers the following advantages:

Integration A tight integration into the Eclipse platform

Data exchange The semantic model can be exchanged easily between different UML diagram editors (e.g. Topcased, UML Lab, Valkyrie, etc.)

Focus on concrete syntax Tool developers can focus on concrete syntax development since abstract syntax and model validation is provided by the Eclipse UML2 project.

Model-driven The Eclipse community offers a broad spectrum of model-driven tools which are based on the Ecore metamodel. These tools were used heavily when developing Valkyrie.

During the development of Valkyrie, model-driven frameworks were used for the following tasks:

Concrete syntax development The diagram editors were implemented with the help of GMF [11]. Manual extension of the generated code was required in some places.

Model-to-model transformations We use ATL [12] model-to-model transformations for several purposes: (1) refactoring of class diagrams, (2) deriving the platform-specific model which is the basis for the code generation and (3) in our reverse engineering mechanism.

Model-to-text transformations To generate code out of UML class diagrams and statecharts, we use Acceleo. The Eclipse M2T (model-to-text) project offers three different frameworks for this purpose. JET, XPand and Acceleo. We chose Acceleo, because it is based on an official standard: MOFM2T [13].

Text-to-model transformations To support software modernization projects for legacy systems, we added reverse engineering capabilities to Valkyrie. We use MoDisco [4] and its capabilities to discover a Java model from existing Java source code. Using an ATL transformation, the discovered Java model is transformed into an UML model for further editing with Valkyrie.

Valkyrie provides extensive support for modeling-in-the large as described in our previous work [6]. To the best of our knowledge, there is no other tool, which supports modeling-in-the-large to that extent. Due to space restrictions, the reader is referred to our article [6] for a detailed explanation of how we realized modeling-in-the-large with Valkyrie. Furthermore, Valkyrie's code generation puts special emphasis on handling associations. Associations, as specified in the UML superstructure [15], require at least two member ends. In case of navigable ends, these ends may be either owned by the opposite classifiers or the association. Navigability and ownership affect the generated code. Thus, our code generator does not only create properties with the respective type and cardinality, it also provides a set of accessor methods to allow easy access to associations. In case of bi-directional navigability, the source code contains mechanisms to ensure the referential integrity of model instances at runtime. If the association ends are owned by the involved classifiers, the code mentioned above is generated into the respective Java classes. Otherwise, a separate Java class for the association is created.

2.2 Sketching support

Sketch[1] is an Eclipse project, which addresses the integration of freehand drawing capabilities into GEF (Graphical Editing Framework)[2] based editors. In their paper [16] Sangiorgi and Barbosa depict their approach realized in that framework. They implemented a sketch recognition algorithm based on Levenshtein's distance algorithm for string comparison [9]. Point lists storing directions are used as a basis for sketch recognition in this approach.

A training interface allows end users to add sketches to shapes based on a 1:1 mapping. In its current state, Sketch is able to interpret different shapes and map it to pre-assigned model elements.

As stated in section 2.1, Valkyrie's diagram editors have been generated with the help of GMF (Graphical Modeling Framework). Internally, GMF is based upon GEF (Graphical Editing Framework). As a consequence, the *Sketch* library can also be used with GMF-based editors. Nevertheless, in order to make *Sketch* work together with our Valkyrie environment, several adoptions and extensions to the *Sketch* library were required.

An issue of the *Sketch* library is the handling of connections. In contrast to the handling of shapes which is supported by the library, it is only possible to detect a connection between two edit parts (in case the first and the last point of the resulting point list lies upon an edit part). A notification is sent to the diagram editor, that a connection between two edit parts has been created. Since commonly not only one type of connection exists between nodes in a diagram (just think of associations and inheritance relationships between classifiers in a class diagram), we had to extend the library to support different types of connections and especially, to recognize them and send notifications to associated diagram editors accordingly.

Furthermore, we added a mechanism which allows the 1:n mapping of shapes to diagram elements which is fully configurable by the

[1] http://www.eclipse.org/sketch/
[2] http://www.eclipse.org/gef

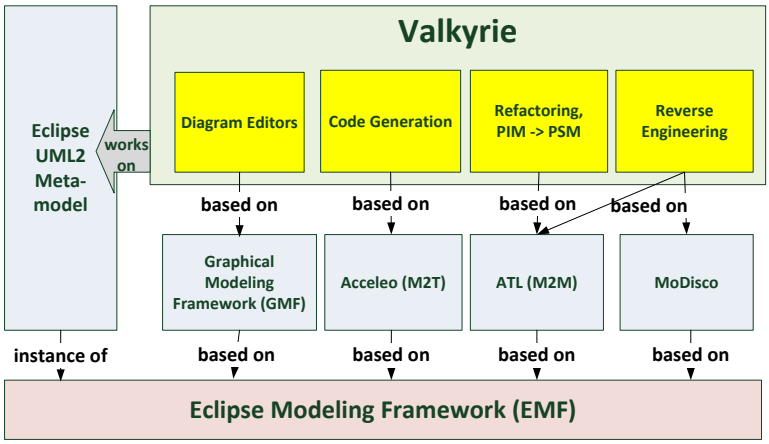

Figure 2: Architecture and used frameworks.

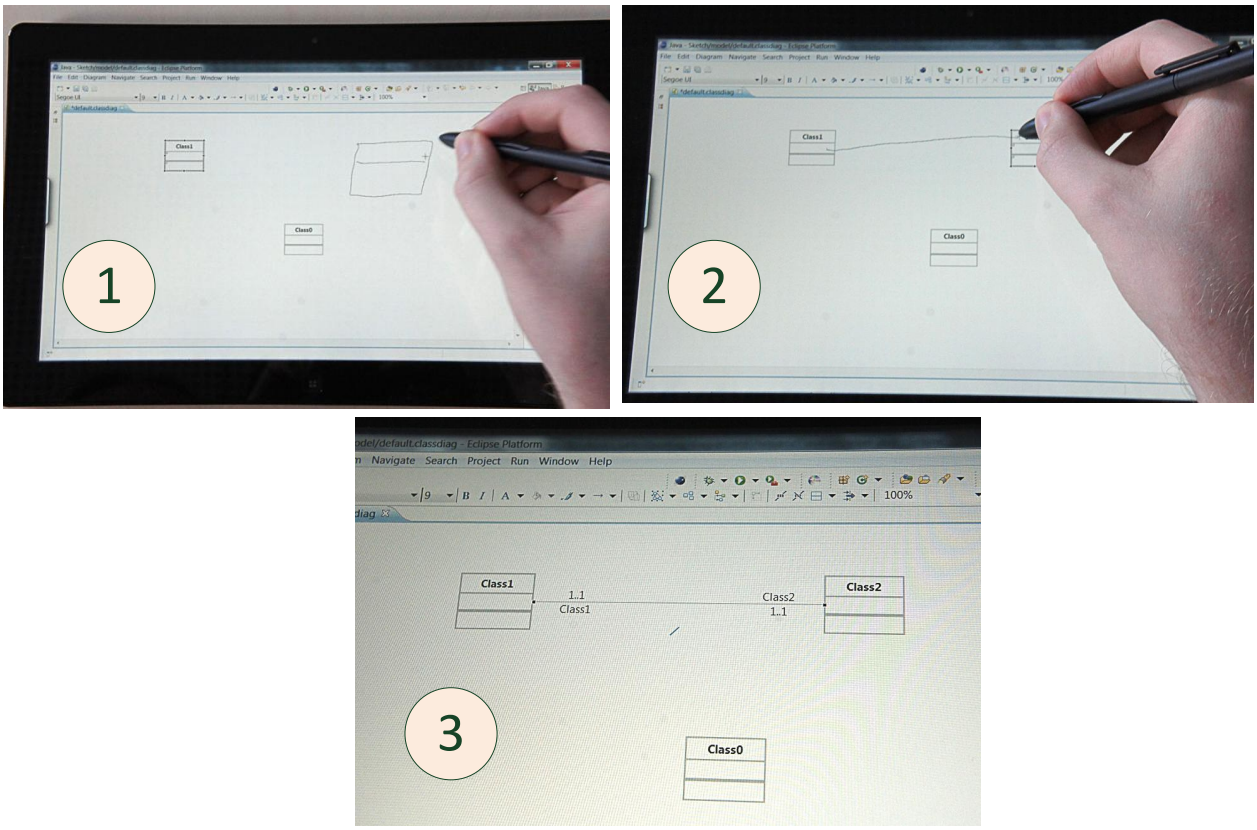

Figure 3: Valkyrie with sketching capabilities.

user. For example, a square can be mapped to classes or interfaces in the class diagram editor as well as to packages in the package diagram editor or states in the state machine editor. Sketch's training interface can be used to add new shapes and our mapping support allows the user to map these shapes to different diagram elements in the respective editors. E.g. in terms of connections between shapes, the user could assign solid connections to associations and dashed connections to generalizations and interface realizations respectively. In its current state, the sketch extension has been integrated into the class diagram editor of Valkyrie. We are currently working on the integration into the other diagram editors.

Figure 3 shows screenshots of our Valkyrie class diagram editor in sketching mode. It runs on a Samsung XE700T tablet pc running Windows 7. The tablet pc allows pen input. Entering a new class with the corresponding class sketch is shown in (1). After the sketch has been recognized, the corresponding diagram element is automatically inserted at the corresponding location in the diagram. (2) depicts how an association between two classes is established by drawing a solid line. Again, after the sketch has been recognized and source and target of the connection has been determined, the appropriate editor command is issued to add an association between the respective classes to the diagram. The final result is shown in (3). Furthermore, our tool is also able to recognize sketches to delete model elements and to add child elements to their containers, e.g. adding properties and operations to classes (not shown in Figure 3).

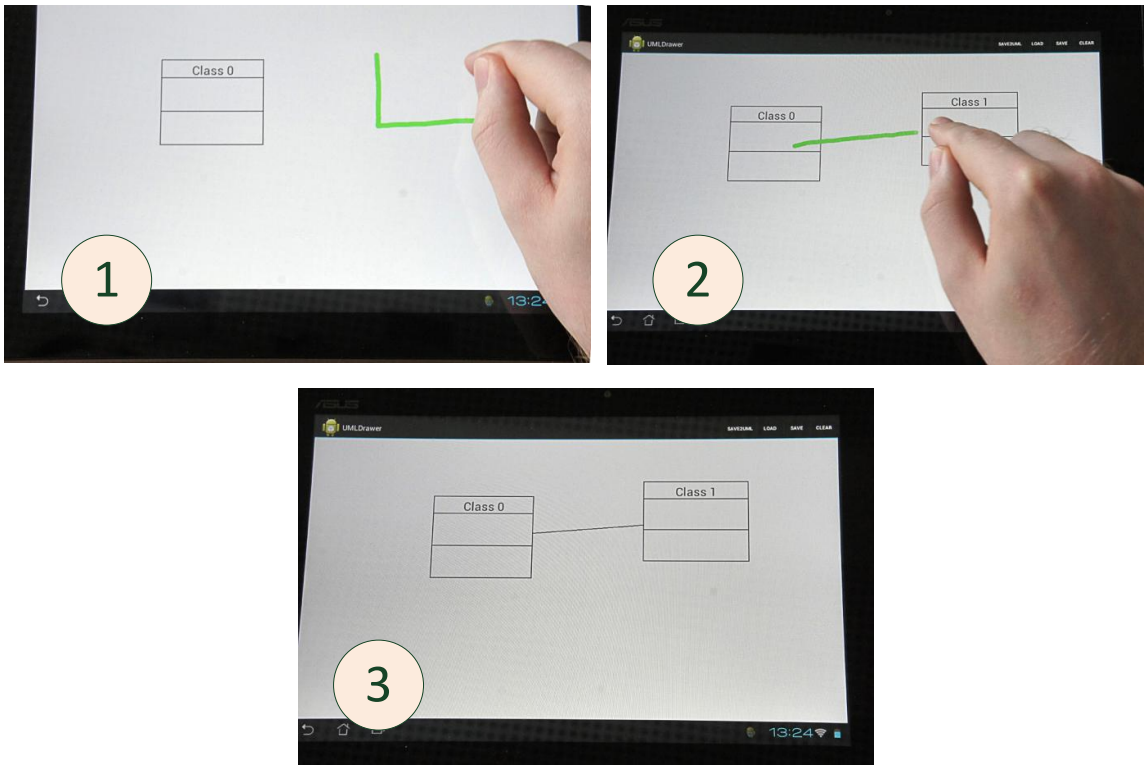

Figure 4: Android app.

2.3 Support for Android devices

We tried to port our successful approach described in section 2.2 to the Android world, as those devices become more and more popular. Usually they are also equipped with powerful multi-core CPUs which promises enough power to run complex applications. Unfortunately, there is no Eclipse distribution for Android systems at the moment. Therefore, we decided to write an application from scratch which provides sketching support for UML diagrams and which is able to persist the sketched models in an XMI file that can be used with our Valkyrie environment for further editing. At the moment, our app is restricted to class diagrams only.

For a seamless integration into the Valkyrie environment, we decided to use EMF and the Eclipse UML2 metamodel also in our Android app. As stated above, there is no Eclipse distribution for Android and also there is no port of EMF to this platform. After some efforts, we managed to get both the Eclipse Modeling Framework (EMF) and the Eclipse UML2 metamodel running within our app. Thus, when sketching a class diagram, corresponding model instances are created accordingly and they are persisted in XMI automatically.

In terms of gesture recognition, the Android platform offers broad support. It provides an app called "Gesture Builder" which allows to record new gestures. The file produced by this app can be imported and reused by own apps. Furthermore, Android provides a GestureOverlayView which can be used in own activities and allows for automatic gesture recognition. After a gesture was performed on a GestureOverlayView, events are fired indicating the name of the gesture that has been recognized. Enriching an app with gesture support is relatively straightforward. The Android SDK documentation recommends to use the GesturePerformedListener which is invoked as soon as the gesture is completed. Unfortunately, only one direction (horizontally or vertically) can be used for gestures when using this listener, as it always assumes the other direction for scrolling purposes. Furthermore, selecting diagram elements was not possible with this listener, as it gets invoked only after a gesture is completed and recognized correctly. Selecting an element is a touch event only which is not reported. Thus, we decided to use OnGestureListener which provides dedicated callback methods for events that may occur during a gesture.

In its current state, our app allows the developer to add and remove classifiers to a class diagram and to interconnect them with generalization and interface realizations as well as associations respectively. Furthermore, it enables the modeler to add and remove attributes and operations to classifiers. All elements can be selected and moved within the diagram. To specify the respective names of model elements, we exploited Android's voice recognition capabilities enabling the user to assign names to classifiers, attributes, operations, associations and association ends.

The screenshots in Figure 4 demonstrate the use of our Android app supporting freehand drawing of class diagrams. It is run on a Asus EEE Pad Transformer. Entering a new class with the corresponding gesture is shown in (1). Once the gesture has been recognized, the corresponding diagram element is created and inserted at the proper location in the diagram. (2) depicts how an association between two classes is established by a solid line. The final result is shown in (3). Our app also provides special gestures to delete model elements or to add properties and operations to classes and interfaces respectively (not shown in Figure 4).

3. RELATED WORK

Since any UML-based modeling tool also supports agile modeling as the UML is not bound to a specific development process, we focus our comparison in this section to tools that explicitly address agile modeling in their tool description. For a comparison of

Valkyrie and other UML modeling tools, the reader is referred to [5].

3.1 Agilian

Agilian[3] is an industry-standard tool which promises to support agile software development with UML, BPMN, ERD, DFD and mind map. It provides mouse gestures to draw new or connect existing diagram elements. However, gesture support is limited to fixed gestures only and no support to add new gestures or to map the existing ones to specific diagram elements is provided. Furthermore, Agilian comprises team collaboration capabilities with connectors to common version control systems like Subversion, ClearCase or CVS, which is not yet supported by Valkyrie. Like Valkyrie it provides capabilities for model refactoring. Besides the support for creating UML models, it also supports the creation of SysML, BMM, BPMN or ERD models. To provide support for legacy projects, import mechanisms for various file formats (including, Rational Rose, MS Excel or MS Visio) exist. However, as Agilian lacks support for code generation or reverse engineering of existing source code, it is a drawing tool only. Hence it does not provide support for model-driven development but only for model-based development.

3.2 Altova UModel

Altova promises to provide support for agile modeling with the tool *UModel*[4]. It provides support for all diagrams supported by UML and is compliant with version 2.3 of the UML specification. However, the diagram editors themselves do not provide support for agile modeling as proposed in our paper. Instead, the user is forced to use mouse and keyboard to enter information to the respective diagrams. Entering additional information like name or type of attributes for example requires special dialogs. Besides sketching support, our tool Valkyrie tries to avoid dialogs as much as possible as it allows the user to add such additional information in a textual way directly within the diagram. Like Valkyrie, UModel provides code generation from class diagrams and statecharts. Furthermore, UModel also supports code generation from sequence diagrams. It supports Java, C# and VB.Net languages. UModel also supports reverse engineering of Java, C# or VB.Net code into UML diagrams. While Valkyrie in its current state is only able to reverse engineer the static structure of a software system, UModel is also able to reverse engineer behavior into sequence diagrams.

3.3 UML Lab

UML Lab[5] aims to provide support for agile modeling, custom code generation and $Roundtrip-Engineering^{NG}$, which allows the user to change model and code simultaneously and changes are automatically propagated from code to model and vice versa. It puts strong emphasis on class diagrams, code generation and the user interface. The user interface comprises classic tools like the palette as well as gesture recognition, context sensitive hints and model autocompletion. Like Agilian, UML Lab only provides basic support for mouse gestures and lacks support to add new shapes or to map them to specific model elements. Currently UML Lab also supports class diagrams only, while Valkyrie provides dedicated support for modeling-in-the-large using package diagrams [6]. Furthermore Valkyrie provides editors for use case diagrams, activity diagrams, state machine diagrams (including code generation support) and object diagrams. While Valkyrie uses the MoDisco framework to realize reverse engineering, UML Lab follows an approach which is based on parsing the code generation templates [3]. While this approach provides benefits in the roundtrip-engineering process, it can not be used if the code which is subject to the reverse engineering does not conform to the standards specified in the code generation templates.

3.4 Sparxsystems Enterprise Architect

Enterprise Architect[6] is another industry standard CASE tool. It is based upon UML 2.4.1 and provides code generation support for 10 different programming languages. It provides a so called whiteboard mode, which unfortunately does not allow freehand drawing. Instead, tools for graphical primitives are supported which allow to create new elements on the canvas. Furthermore, all UML elements can be placed on the canvas, too. In our oppinion, this approach is not very helpful in agile modeling, as the modeler spends a lot of time in selecting the appropriate tools from the diagram palette in order to add new diagram elements. Contrastingly, our approach allows the interpretation of user sketches and the automatic creation of corresponding diagram elements which enables the user to sketch diagrams just like on a piece of paper or on a real whiteboard.

3.5 Android apps

To the best of our knowledge there is no Android app which provides support for drawing UML class diagrams using gestures and voice input similar to our app described in this paper. There are a few drawing tools like *UML Factory*[7] and *AndyUML*[8]. UML Factory uses a tool palette to add model elements to diagrams. Dialogs are required for further refinement. Constrastingly, our app only uses gestures and voice input. As a consequence, our app directly replaces paper and whiteboards, as the users can just "draw" their diagrams on the canvas, which results in a better support for agile development processes. AndyUML is just a frontend to *yUML*[9]. yUML uses a special textual syntax to create diagrams. Therefore, it can not be used as an adequate replacement for papers or whiteboards, since diagrams are specified by the user via a textual syntax rather than by sketching.

3.6 E-whiteboard based approaches

In the past there have been some approaches described in [20] or [8] that are based on electronic whiteboards. These approaches require special software that recognizes the hand drawn sketches and which afterwards exports these sketches into the respective modeling tool. Contrastingly, in our approach the user can sketch directly within the modeling environment. Additional recognition software or transformation steps are avoided in our approach. Furthermore, since we provide support for mobile devices, the user is not restricted to a fixed and large whiteboard.

4. CONCLUSION AND FUTURE WORK

In this paper we presented an extension to our UML-based modeling tool *Valkyrie* which allows for freehand sketching support for diagrams. Using this tool, whiteboards or papers might become obsolete in agile development processes. Instead of manually converting diagrams sketched on a whiteboard during a meeting to a modeling environment (which can be a cumbersome and error prone

[3] http://www.visual-paradigm.com/product/?favor=ag
[4] http://www.altova.com/agile_umodel.html
[5] http://www.uml-lab.com/en/uml-lab/
[6] http://www.sparxsystems.com/products/ea/index.html
[7] http://www.umlfactory.com/
[8] https://play.google.com/store/apps/details?id=com.yeradis.android.yuml
[9] http://yuml.me/

process), our tool supports sketching directly within the modeling tool. The resulting model can be refined further without any additional steps.

In the future, we plan to further extend our tool by investigating other algorithms to detect freehand drawing. Furthermore, we want to integrate model differencing [18] and model versioning [19], which both are further research topics at our lab into Valkyrie. Future work on the Valkyrie core also comprises test case generation from use case diagrams and activity diagrams. Furthermore, we plan to integrate support for the Action Language Foundation (including code generation from ALF specifications) [14]. Additionally, we are working on a mechanism to support round-trip engineering to allow the seamless editing of source code and diagrams. In [7] we describe our approach to add modeling with graph transformations to Ecore models. We plan to adopt this approach to our Valkyrie environment, empowering the user to specify the behavior of methods defined in the class diagram using a powerful declarative and graphical notation.

Future work on the Android App comprises support for all diagrams currently supported by Valkyrie. Furthermore, we plan to also implement a configurator, which allows the user to map gestures on specific model elements. Additionally we will add support for wireless data exchange via bluetooth and/or WLAN, to easily exchange models between Android devices and or desktop computers. There are lots of apps that promise to recognize handwriting. A useful extension to our app would be integrating one of them in order to allow the modeler to specify names not only by voice recognition, but also by handwriting.

To test our approach, we are planing to use our Valkyrie environment with the sketching clients in upcoming student projects which use agile development processes.

Acknowledgements

The author wants to thank Anke Giebler-Schubert for adopting the GEF Sketch library during a master project and Patrick Pezoldt for working on the Android App during his master project. Furthermore, I want to thank Bernhard Westfechtel for the valuable inputs on the draft of this paper.

5. REFERENCES

[1] S. W. Ambler. Agile modeling: A brief overview. In A. Evans, R. B. France, A. M. D. Moreira, and B. Rumpe, editors, *pUML*, volume 7 of *LNI*, pages 7–11. GI, 2001.

[2] S. W. Ambler. *Agile modeling: Effective Practices for Extreme Programming and the Unified Process*. John Wiley & Sons, Inc., New York, 2002.

[3] M. Bork, L. Geiger, C. Schneider, and A. Zündorf. Towards roundtrip engineering - a template-based reverse engineering approach. In I. Schieferdecker and A. Hartman, editors, *ECMDA-FA*, volume 5095 of *Lecture Notes in Computer Science*, pages 33–47. Springer, 2008.

[4] H. Bruneliere, J. Cabot, F. Jouault, and F. Madiot. MoDisco: a generic and extensible framework for model driven reverse engineering. In *Proceedings of the IEEE/ACM international conference on Automated software engineering*, ASE '10, pages 173–174, New York, NY, USA, 2010. ACM.

[5] T. Buchmann. Valkyrie: A UML-Based Model-Driven Environment for Model-Driven Software Engineering. In *Proceedings of the 7th International Conference on Software Paradigm Trends (ICSOFT 2012)*. INSTICC, July 2012.

[6] T. Buchmann, A. Dotor, and B. Westfechtel. Model-driven software engineering: concepts and tools for modeling-in-the-large with package diagrams. *Computer Science - Research and Development*, pages 1–21, 2012. Online first.

[7] T. Buchmann, B. Westfechtel, and S. Winetzhammer. MODGRAPH - A Transformation Engine for EMF Model Transformations. In *Proceedings of the 6th International Conference on Software and Data Technologies*, pages 212 – 219, 2011.

[8] Q. Chen, J. Grundy, and J. Hosking. Sumlow: early design-stage sketching of uml diagrams on an e-whiteboard. *Softw. Pract. Exper.*, 38(9):961–994, July 2008.

[9] A. Coyette, S. Schimke, J. Vanderdonckt, and C. Vielhauer. Trainable sketch recognizer for graphical user interface design. In C. Baranauskas, P. Palanque, J. Abascal, and S. Barbosa, editors, *Human-Computer Interaction Ű INTERACT 2007*, volume 4662 of *Lecture Notes in Computer Science*, pages 124–135. Springer Berlin / Heidelberg, 2007.

[10] Eclipse Foundation. Model development tools (mdt). http://www.eclipse.org/modeling/mdt/?project=uml2, Feb. 2012. last visited: 2012/02/27.

[11] R. C. Gronback. *Eclipse Modeling Project: A Domain-Specific Language (DSL) Toolkit*. The Eclipse Series. Boston, MA, 1st edition, 2009.

[12] F. Jouault, F. Allilaire, J. Bézivin, and I. Kurtev. Atl: A model transformation tool. *Science of Computer Programming*, 72:31 – 39, 2008. Special Issue on Second issue of experimental software and toolkits (EST).

[13] OMG. *MOF Model to Text Transformation Language, Version 1.0*. OMG, Needham, MA, formal/2008-01 edition, Jan. 2008.

[14] OMG. *Action Language for Foundational UML (Alf)*. Object Management Group, Needham, MA, ptc/2010-10-05 edition, Oct. 2010.

[15] OMG. *UML Superstructure*. Object Management Group, Needham, MA, formal/2011-08-06 edition, Aug. 2011.

[16] U. B. Sangiorgi and S. D. Barbosa. SKETCH: Modeling Using Freehand Drawing in Eclipse Graphical Editors. In *FlexiTools2010: ICSE 2010 Workshop on Flexible Modeling Tools*, Cape Town, South Africa, May 2010.

[17] D. Steinberg, F. Budinsky, M. Paternostro, and E. Merks. *EMF Eclipse Modeling Framework*. The Eclipse Series. Boston, MA, 2nd edition, 2009.

[18] S. Uhrig. Matching class diagrams: With estimated costs towards the exact solution? In *Proceedings of the 2008 International Workshop on Comparison and Versioning of Software Models (CVSM 2008)*, pages 7–12, Leipzig, Germany, 2008. ACM, New York, NY.

[19] B. Westfechtel. A formal approach to three-way merging of emf models. In D. D. Ruscio and D. S. Kolovos, editors, *Proceedings of the 1st International Workshop on Model Comparison in Practice (IWMCP 2010)*, pages 31–41, Malaga, Spain, July 2010. ACM. Copyright ACM.

[20] J. Wu and T. C. N. Graham. The software design board: a tool supporting workstyle transitions in collaborative software design. In *Proceedings of the 2004 international conference on Engineering Human Computer Interaction and Interactive Systems*, EHCI-DSVIS'04, pages 363–382, Berlin, Heidelberg, 2005. Springer-Verlag.