Sketches and Natural Language in Agile Modeling

Dominik Fuchß

KASTEL – Institute of Information Security and Dependability, Karlsruhe Institute of Technology, Karlsruhe, Germany

Abstract
Hand-drawn sketches help to quickly grasp structures and facts about software architectures during software development. Therefore, software architects and software developers also use them in their everyday work. A major problem with drawings is the fact that, unlike more formal models, their information must be interpreted rather than read out directly. Thus, they are difficult to process automatically. This paper presents an approach to create links between elements in sketches and elements in more formal models such as architectural models. In order to achieve the element recognition, we consider natural language as further source of information. We show various options how these links can support architects and developers in their design, for example by reporting inconsistencies directly.

Keywords
Informal Models, Formal Models, Sketches, Agile Modeling

1. Introduction

During agile software development, Informal Artifacts (IAs) and Formal Artifacts (FAs) are created by the project team. In our case FAs are artifacts that are instances of technical meta models and can therefore processed automatically to get information from them. An example for FAs would be code or models / diagrams that have been created in a formal language editor. Since the *UML Diagram Interchange* [1] specifies an exchange format for UML diagrams, they can be seen as formal artifacts. IAs like hand-drawn sketches, transcripts of discussions, or voice recordings are difficult to process automatically because they have to be interpreted first. However, the interpretation of IAs is not only desirable but necessary to access their inherent knowledge. In contrast to that, FAs can be mapped to their underlying meta model to access their information directly.

We illustrate the benefits of automatic analysis of IAs with an example in Figure 1. The figure shows an architect explaining a part of the current system during a developer meeting using a smart whiteboard. Unfortunately, the architect made a mistake while sketching that would lead to inconsistencies with a stored formal artifact. This inconsistency could be detected by automatic analysis of the sketch and comparing elements of the sketch with model elements.

The overall goal of this thesis is to establish links between sketches and FAs like architecture models by using sketch recognition techniques, natural language processing, and speech recognition. This enables tool support for architects and developers to directly point out possible

ECSA'21: Doctoral Symposium, September 13–17, 2021, Växjö, Sweden
dominik.fuchss@kit.edu (D. Fuchß)
0000-0001-6410-6769 (D. Fuchß)

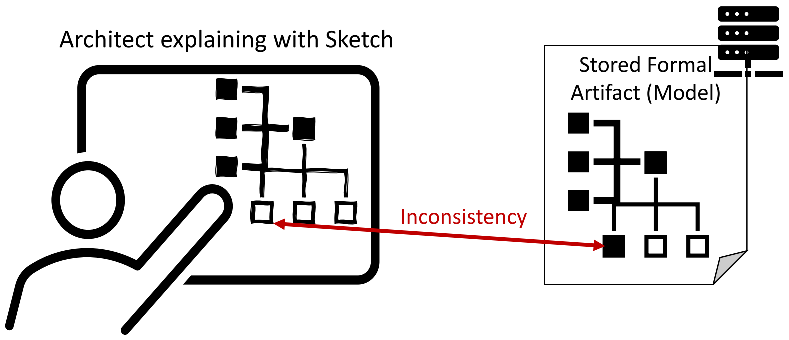
Architect explaining with Sketch
Stored Formal Artifact (Model)
Inconsistency

Figure 1: Architect explains Architecture by using Sketches

inconsistencies. In addition to this scenario, other applications of such links are feasible, e.g., a support for drawing diagrams on the basis of existing models.

This work shall address the following problems: System knowledge from sketches is difficult to use in an automated way (**P1**). Furthermore, there is a lack of support for finding inconsistencies while working with sketches (**P2**). Therefore, our Research Questions (RQs) at the current stage are the following:

RQ1 How can formal artifacts be used to improve the recognition of (informal) sketches?

RQ2 Does tool support for working with sketches improve the software development process?

RQ3 How can inconsistencies between formal artifacts and sketches be detected?

RQ1 deals with the detection of elements from sketches. Especially, whether the knowledge of existing model elements helps to resolve uncertainties and ambiguities during recognition is the question here. **RQ2** addresses the mentioned future tool support that shall help developers and architects during the Software Development Process (SDP). Here, we want to analyze its influence on the SDP. **RQ3** is the most complex so far. Assuming the existence of formal artifacts (e.g., digital models of the (planned) system), the RQ deals with the general question, if inconsistencies between sketches (IAs) and such FAs are detectable. The problem here is the differentiation between inconsistency, absence of certain elements because of simplifications, and presence of elements because of creation or planning activities.

The paper is organized as follows: In a first step, the related work will be discussed in Section 2. Afterwards, we focus the actual approach to work with sketches in SDPs in Section 3. Then, the planned evaluation steps and expected results will be discussed in Section 4. Section 5 concludes this paper and reflects the thesis.

2. Related Work

This section discusses the related work of the thesis. First, we focus on *Modern Sketch Recognition Systems* that are related to the SDP. Afterwards, we analyze research regarding *Assistance Systems for Developers and Architects.*

Modern Sketch Recognition Systems Wüest et al. [2] developed *FlexiSketch*, a framework that allows end-users to draw models on a tablet or computer. In addition, the system supports the user by creating an on-the-fly meta model from the elements the user has drawn. The system is able to use concepts like cardinality for the meta models. The meta model elements are identified by comparing drawn elements and grouping similar ones. In contrast to our approach, their focus lies on creation of models and meta models.

Schäfer [3] created a system that is able to detect sketched processes in a business process domain. To achieve the recognition, they built *Arrow R-CNN* a "deep learning detector for flowchart structure recognition". The overall goal of them is the "end-to-end flowchart recognition". In contrast to our approach, they focus on one single type of diagram: Business Process Diagrams / Flowcharts. Furthermore, they think about using a formal model to correct recognition errors. Therefore, in contrast to our approach, they do not consider the information of informal models to correct the formal model. Besides that, they are focusing on business process modeling instead of architectural models. In their recent research [4], they applied their approach on a finite automata dataset. Since the recognition results are promising, it has to be considered whether their approach can be adapted for architectural models in our research.

This related work supports the idea that automatic processing and recognition of sketches should be facilitated. At the same time, the connection with existing formal models still has to be accomplished.

Assistance Systems for Developers and Architects Samuelsson and Book [5] built a sketched-based user interface for Integrated Development Environments (IDEs). They state that "sketching could be a more intuitive way of expressing user intentions than navigating nested menus". Their system shall provide an IDE that is controlled by sketches. E.g., a user could mark a method by a circle and draw an arrow pointing to the method's new location. In contrast to our work, they are focusing on sketches as a user interface.

OctoUML [6] is a system to bridge the "gap between informal designing [...] and formal design and documentation practices in subsequent development". Therefore, OctoUML is able to generate formal models out of sketched diagrams. The system is able to recognize sketched UML classes and to name them by using voice commands. In contrast to our work, they focus on combination of formal and informal models rather than finding inconsistencies between sketches and existing models.

In summary, assistance systems focus on user interactions or creation of diagrams. The linking of formal artifacts and sketches as informal artifacts still needs to be addressed.

3. Approach for Linking Sketches and Software Architecture

This section explains the approach that is currently planned to tackle the research questions. In this paper, sketches, speech, and/or written text are the considered Informal Artifacts (IAs). Digital architectural models (e.g., *Palladio Component Model (PCM)* [7]) and code are the formal ones (FAs).

The overview in Figure 2 illustrates that the approach shall analyze information from informal and formal artifacts to perform specific tasks (e.g., recovery of trace links). Sketch recognition

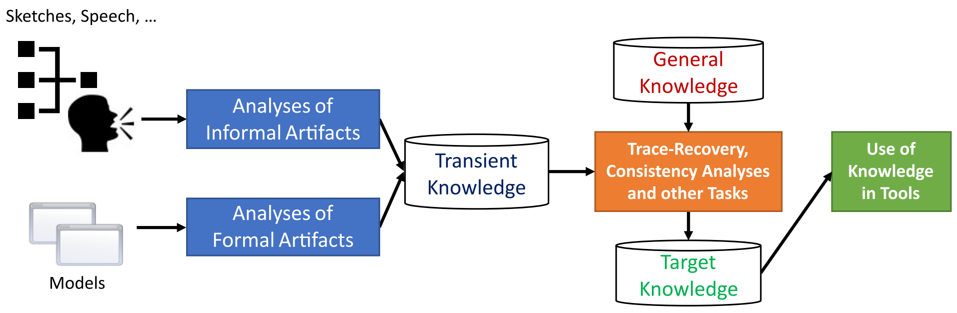
Sketches, Speech, ...
Models
Analyses of Informal Artifacts
Analyses of Formal Artifacts
Transient Knowledge
General Knowledge
Trace-Recovery, Consistency Analyses and other Tasks
Target Knowledge
Use of Knowledge in Tools

Figure 2: Overview of the Approach

is used to gather the structure of diagrams describing a system or parts of it. Therefore, sketch recognition as part of the analyses of IAs is used to obtain the inherent knowledge of sketches. For the purpose of recognition, *Arrow R-CNN* [4] could be used, as they have already shown good results for the detection of flow charts. Speech and natural language processing techniques can be used here to, e.g., name the recognized elements (cf. Section 2). Imagine architects who describe which component they are currently sketching. Alternatively, handwriting recognition approaches can be used to identify the element's names. In the current state of the work, no final decision has been made as whether speech should be considered as a source of information. The result of this analysis stage subsumes all structural information obtained from diagrams as well as any information gathered about possible names or element types (e.g., class, interface, ...)

After the analyses of the formal and informal artifacts, the system can use two knowledge sources: *General Knowledge (GK)* and *Transient Knowledge (TK)*. GK refers to knowledge about the domain or knowledge about architecture in general (e.g., architecture patterns). TK describes the information obtained from analyses of FAs and IAs, such as PCM models [7] or a representation of the source code as formal artifacts and sketches as informal artifacts.

Using GK and TK the envisaged tool shall perform specific tasks (e.g., the creation of trace links). This execution provides the *Target Knowledge* that can be used to present certain information (e.g., inconsistencies) to users of the approach. Since we first want to focus on the structure of static models to link sketches and models, we plan to evaluate algorithms for "inexact pattern matching" [8] in graphs. Therefore, we conduct exploratory experiments to test whether these algorithms can be applied in our use cases.

4. Evaluation

This section covers the planned evaluation of our work using a Goal-Question-Metric-plan [9].

The first goal of this work is the creation of links between sketches and (architectural) models (**G1**). In this context, it is important that these links are made available to the architects and developers in a usable manner. Therefore, the key question regarding **G1** is how good elements of sketches can be linked to elements in models (**Q1.A**: *Linking*). In addition, we want to evaluate if architects and developers consider tool support for working with sketches as useful (**Q1.B**: *Usability*). The second goal of the thesis is the automatic detection of inconsistencies between sketches and models (**G2**). The important questions are how can inconsistencies between sketches and models be characterized (**Q2.A**: *Categories*) and how many of them are recognized automatically (**Q2.B**: *Classification*)? In terms of **Q1.A** and **Q2.B** measures for classification tasks such as precision, recall, and F_β score shall be applied to case studies. Regarding **Q2.A**, inconsistency classes that occur could be defined based on the case studies. In order to measure the final usefulness of a tool for working with sketches and models (**Q1.B**), a demonstration with expert interviews shall be used.

Due to the early state of the work, this is a first planning of a possible evaluation. It is expected that through this type of evaluation both technical and personal aspects related to sketching can be investigated.

5. Conclusion

We presented our planned approach to tackle the missing connections between sketches as informal artifacts and more formal artifacts such as code. In order to clarify our research, we stated several research questions and goals we want to address. Even though the basic concepts of the work had to be defined first, important questions and steps regarding the evaluation could be identified. Although we are still at the beginning of the work, we have demonstrated the importance of addressing the problems regarding the lack of processing sketches.

References

[1] Object Management Group (OMG), UML Diagram Interchange, 2006. URL: omg.org/spec/UMLDI/1.0/PDF.

[2] D. Wüest, N. Seyff, M. Glinz, FlexiSketch: a lightweight sketching and metamodeling approach for end-users, Software & Systems Modeling 18 (2019) 1513–1541.

[3] B. Schäfer, Business Process Sketch Recognition, Proceedings of the Dissertation Award, Doctoral Consortium, and Demonstration Track at BPM 2019 - BPM Doctoral Consortium (2019) 6. URL: ceur-ws.org/Vol-2420/paperDC9.pdf.

[4] B. Schäfer, M. Keuper, H. Stuckenschmidt, Arrow R-CNN for handwritten diagram recognition, International Journal on Document Analysis and Recognition (IJDAR) (2021).

[5] S. G. Samuelsson, M. Book, Towards Sketch-based User Interaction with Integrated Software Development Environments, in: Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops, ACM, Seoul Republic of Korea, 2020, pp. 181–184.

[6] B. Vesin, R. Jolak, M. R. Chaudron, OctoUML: An Environment for Exploratory and Collaborative Software Design, in: 2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C), 2017, pp. 7–10.

[7] R. Reussner, S. Becker, E. Burger, J. Happe, M. Hauck, A. Koziolek, H. Koziolek, K. Krogmann, M. Kuperberg, The Palladio Component Model, Technical Report, Karlsruhe, 2011.

[8] R. Pienta, A. Tamersoy, H. Tong, D. H. Chau, MAGE: Matching Approximate Patterns in Richly-Attributed Graphs, IEEE International Conference on Big Data (2014) 6.

[9] V. R. Basili, D. M. Weiss, A Methodology for Collecting Valid Software Engineering Data, IEEE Transactions on Software Engineering SE-10 (1984) 728–738.