OctoUML: An Environment for Exploratory and Collaborative Software Design

Boban Vesin
Department of Computer & Information Science
Norwegian University of Science and Technology
Trondheim, Norway
boban.vesin@idi.ntnu.no

Rodi Jolak, Michel R. V. Chaudron
Joint Department of Computer Science and Engineering
Chalmers University of Technology and Gothenburg University
Gothenburg, Sweden
{jolak,chaudron}@chalmers.se

***Abstract*—Software architects seek efficient support for planning and designing models at multiple levels of abstraction and from different perspectives. For this it is desirable that software design tools support both informal and formal representation of design, and also support their combination and the transition between them. Furthermore, software design tools should be able to provide features for collaborative work on the design. OctoUML supports the creation of software models at various levels of formality, collaborative software design, and multi-modal interaction methods. By combining these features, OctoUML is a prototype of a new generation software design environment that aims to better supports software architects in their actual software design and modelling processes.**
Demo video URL: https://youtu.be/fsN3rfEAYHw
OctoUML Project URL: https://github.com/Imarcus/OctoUML

***Keywords*-software design; modelling notations; multi-modal interaction; collaborative design; user experience; UML**

I. Introduction

Designing software consists of exploring design problems, discussing solutions and creating software models as design artifacts. Such artifacts provide a bridge between problem and software implementation by describing user's needs as well as the product to be developed. As software systems are gaining increased complexity, the importance of efficient software design tools is also increasing. Software models change frequently and are quite often updated by many designers simultaneously [2]. These models should present a description of complex systems at multiple levels of abstraction and from a different perspectives. Therefore, it is crucial to provide software design tools that give possibilities for efficient and collaborative development as well as options for multi-modal interaction.

Modelling tools can be classified into two groups: informal and formal [3]. We mean by informal any tool that supports informal design in the sense that it does not constrain the notation used. Indeed, informal tools are preferred for their flexibility as well as the role that they play in unleashing designers' expressiveness. Examples of such tools are whiteboards, paper and pencil. While we mean by formal any tool that support one or few formalized notations. Typical examples are UML CASE-tools (e.g. Rational Rose, Enterprise Architect, Papyrus, StarUML, etc.). Formal tools are usually used for code-generation and/or documenting purposes.

During early design phases, software designers often use informal tools (e.g. whiteboards) to sketch their thoughts and compare design ideas. Once the designers settle on one possible solution, they proceed to create a formal version of the sketchy design. In particular, they move from the whiteboard, start-up the computers, run a formal tool (a CASE-tool), and re-enter the solution that has been created previously during the early design phase. So there is a gap between informal designing in early software design phases and formal design and documentation practices in subsequent development. To bridge this gap, we present *OctoUML*, a software design environment that supports exploratory and collaborative design meetings. OctoUML provides means to allow the creation of both sketchy hand-drawn elements and formal notations simultaneously. Moreover, it allows the transformation of sketchy designs into formal notations.

Oviatt and Cohen [9] illustrated the importance of multi-modal systems in reshaping daily computing tasks and predicted their future role in shifting the balance of human-computer interaction much closer to the human. We enabled OctoUML to support multiple modes of interaction including mouse, keyboard, touch/multi-touch using fingers and styluses, sketching, and voice modality.

More often than not, the process of software design involves several designers working on the same project simultaneously. This could also occur in user-centered design situations where users are involved in the design process. We implemented OctoUML to support design collaborative sessions, both *in-situ* (via the adoption of multi-touch technique) and at a distance *"remotely"* (by using a client-server paradigm). OctoUML can be run using a number of input devices ranging from desktop computers over large touch screens to large interactive whiteboards.

The paper is organised as follows: the related work is presented in section two. Further information on OctoUML, its architecture and features, and the performed evaluation are reported in section three. The future objectives and concluding remarks are presented in the last section (section four).

II. Related Work

Several studies proposed different approaches to enhance the software design process. Mangano et al. [8] identified some

behaviors that occur during informal design. In particular, designers sketch different kind of diagrams (e.g. box and arrow diagrams, UI mock-ups, generic plots, flowcharts, etc.) and use impromptu notations. The authors implemented an interactive whiteboard system (called *Calico*) to support these behaviors and identified some ways where interactive whiteboards can enable designers to work more effectively.

Wüest et al. [12] stated that software engineers often use paper and pencil to sketch ideas when gathering requirements from stakeholders, but such sketches on paper often need to be modelled again for further processing. A tool, *FlexiSketch*, was prototyped by them to combine free-form sketching with the ability to annotate the sketches interactively for an incremental transformation into semi-formal models. The users of *FlexiSketch* were able to draw UML-like diagrams and introduced their own notation. They were also able to assign types to drawn symbols. Users liked the informality provided by the tool, and had the will to adopt it in practice.

Magin and Kopf [7] created a multi-touch based system allowing users to collaboratively design UML class diagrams on touch-screens. They have also implemented a new algorithm to recognize the gestures drawn by the users and to improve the layout of the diagrams. However, their tool does not allow for informal freehand sketching of arbitrary notations.

Lahtinen and Peltonen [6] presented an approach to build speech interfaces to UML tools. The authors set up a spoken language to manipulate the UML models, and built a speech control system (*VoCoTo*) integrated with a CASE-tool (*Rational Rose*). They stated that speech recognition is applicable to be used to enhance the interaction with UML tools.

Table I summarizes the main supported functionalities by OctoUML and illustrates the differences to the related work.

Related Work	Informal & formal notations	Interaction Modalities	(MT,RC*)
Calico	informal hand-drawn notations	mouse, keyboard and touch	(no, no)
Flexisketch	informal hand-drawn notations	mouse, keyboard and touch	(no, no)
Magin&Kopf	formal notations creation via gestures	touch-based	(yes, no)
VoCoTo	formal notations	mouse, keyboard and voice	(no, no)
OctoUML	creation and mix of informal and formal notations simultaneously	mouse, keyboard, single touch, multi-touch, and voice	(yes, yes)

TABLE I
COMPARISON BETWEEN OCTOUML AND THE RELATED WORK.
*MT:MULTI-TOUCH, RC:REMOTE COLLABORATION.

III. OCTOUML

In a previous work [3], we presented our vision for a new generation software design environment. To realize our vision, we developed a prototype called OctoUML [5]. OctoUML is a software design environment that supports exploratory and collaborative software design. It is used to create and organize diagrams as well as supports their modification and evolution. Firstly, we illustrate the architecture of OctoUML. Secondly, we describe the main functionalities that are supported by OctoUML (sections B and C). Later on, we provide a scenario showing how such functionalities could support the design process. Lastly, we provide some details on OctoUML evaluation.

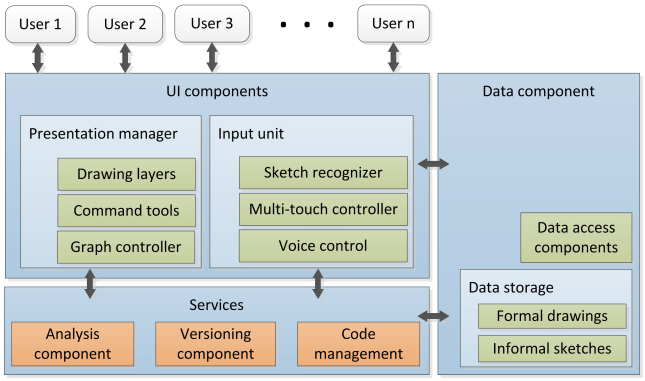

Fig. 1. Architectural Components of OctoUML

A. OctoUML Architecture

The key architectural components of OctoUML are presented in Figure 1. The environment contains three major components: *UI component*, *Data component* and *Services*. The current version of the system offers only the *UI* and *Data* components. Additional services will be added during future development. The *UI component* consists of: *Presentation manager* and *Input unit*. The *Presentation manager* provides means for performing stylus or touch-based input commands on devices being used. *Drawing layers* include support for both informal and formal modelling layers. The *Command tools* are responsible for transferring the inputs from users to different controllers. The *Graph controller* allows switching between different input techniques with combining of multiple layers. The *Input unit* is responsible for processing different inputs. In particular, a *Sketch recognizer* is provided to recognize and transform informal models into formal concepts, and hence allows to maintain and transfer the designs for further processing tasks. A *Multi-touch* controller captures and coordinates the inputs from different touch-points. All the program data are saved and stored in the *Data component*. Our tool uses a set of data structures to manage and maintain the sketched elements and formalized designs.

B. Informal and Formal Notation

Whiteboards (or any informal tools e.g. paper and pen) are used during early software design phases because of their flexibility and immediacy, but also becuase they do not constrain the notation being used. Informal notations (e.g. *sketches*) can be used to express abstract ideas representationally, to allow checking the entirety and the internal consistency of an idea as well as to facilitate development of new ideas [11]. Furthermore, informal notations can have a very close mapping to the problem domain. However, the informal notations often need to be formalized in order to allow their manipulation and process e.g. sharing, code generation or documentation.

Modelling tools should not constrain designers to create only some specific notations. Furthermore, they should maintain the characteristics of formal tools in their support of design transfer and persistence [3].

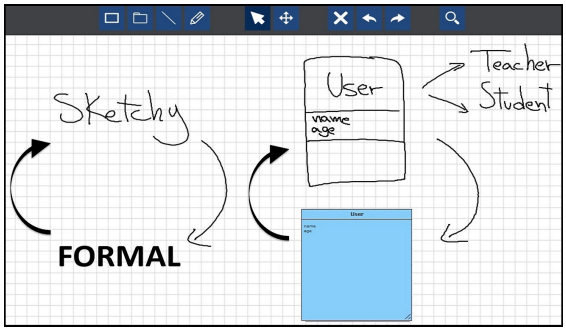

Fig. 2. Combination of different notations on the same canvas

OctoUML allows the creation of both hand-drawn informal sketches and computer-drawn formal elements (currently UML class and sequence models) on the same canvas simultaneously (Figure 2). OctoUML bridges the gap between early software design process, when *informal* tools are typically used, and later documentation and formalization process, when *formal* tools are used. Beside supporting the creation of software models at different levels of formality, OctoUML is equipped with a *Sketch recognition unit* which enables sketch formalization. In particular, OctoUML allows the transformation of models from informal to formal and vice versa at any time during the modelling session. Furthermore, we adopted a layering technique by which the informal notations belong to one layer that we call the *informal layer*, and the formal notations belong to another layer that we call the *formal layer*. The user can then select to see the layers in combination or isolation.

C. Interaction Modes and Collaboration

The usability of current CASE tools is a common source of criticism [4]. The interaction with such tools is often based on using the mouse and keyboard. Other modes of interaction (e.g. touch, gesture and voice) could be more natural and intuitive. In order to improve the user experience of OctoUML and increase its accessibility, the interaction modalities of OctoUML are enriched by providing a *voice-commands* recognition component capable of transforming designers' voice-commands into control actions.

The process of software design often involves more than one designer working on the same project simultaneously. OctoUML promotes collaborative design by adopting a *multi-touch* technique and supporting *remote collaboration*. Next, we provide more details on the supported functionalities:

- *Multi-touch* is an interaction technique that permits the manipulation of graphical entities by more users at the same time. Our tool allows multiple users to design diagrams simultaneously by performing simple touch gestures.
- In order to improve the user experience, we integrated a *voice-commands* control component within the *Input unit*. The component is capable of handling the most commonly used functions during the design process. Thus, users can use voice commands in order to create and edit elements of software diagrams.
- To open up new opportunities for interactive collaborative design, our tool supports *remote collaborative* sessions between geographically distributed teams. One team of designers can run a server instance of OctoUML, whereas another team can join the session as client connecting to the server. Video calls and chatting tools will be integrated in order to support the joint design sessions.

D. Design process in UctoUML: A Scenario

Figure 3 illustrate the design process in OctoUML. Activities that are currently supported by OctoUML are distinct in green. Let us think about the following scenario: a group of software designers meet to explore and discuss design ideas of a specific software product. The designers start with the creation of some informal sketchy designs using OctoUML being deployed on a large interactive whiteboard. After that, the designers proceed with a selective transformation of some informal sketches into a formal model. Later on, the created model is analyzed to check possible flaws and performance bottlenecks. Finally, the model is saved and uploaded to a version control repository. The designers meet again (on-site or from different locations) when new requirements come out or having earlier requirements exposed to changes. They fetch the design that was previously shared on the version repository, update the design, and commit a new version that is now compliant to the new requirements.

E. Evaluation

Two *user studies* were performed to evaluate OctoUML. In both studies, the participants had to do a modelling task using OctoUML, answer a System Usability Scale (SUS) questionnaire [1], and participate into semi-structured interviews. The first study involved fourteen software engineering students (ten PhD and four M.Sc. students) and two post-doc researchers. The main purpose of the first study was to evaluate the usability of OctoUML as well as to investigate whether supporting the mix of informal and formal notation could support the design process. OctoUML got an average SUS-score of 78.75 which is a high usability score according to [10]. The participants stated that informal notations could be valuable artifacts beyond being just explorative means. They also stated that such notations support designers' activities in understanding the problems and communicating ideas. Figure 4 shows the feedback from the participants regarding the use of informal and formal notations within OctoUML.

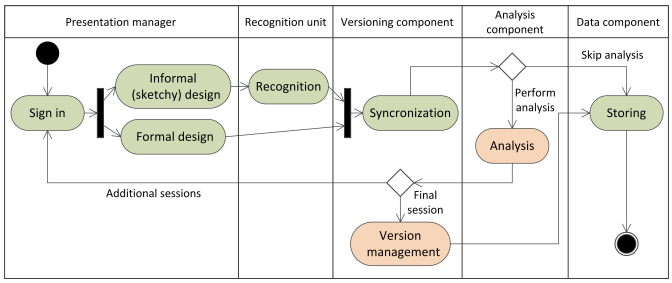

Fig. 3. Design process in OctoUML

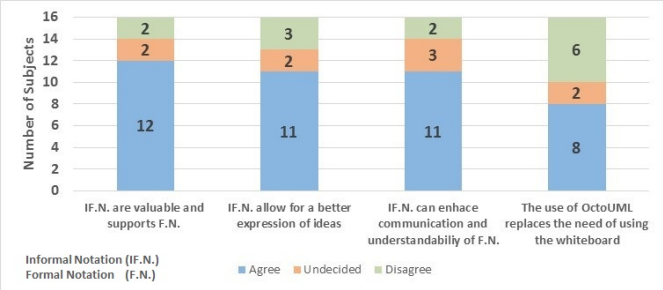

Fig. 4. User study I: informal vs. formal notations

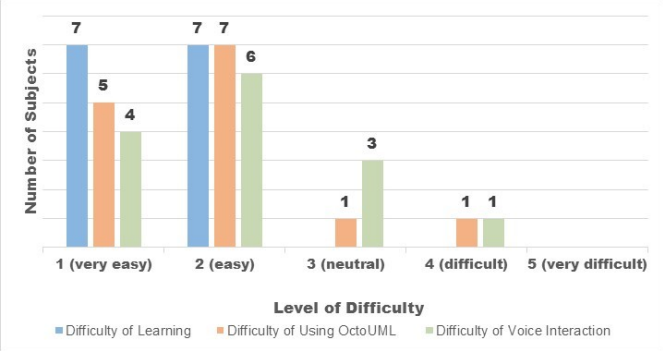

Fig. 5. User study II: usability and learnability of OctoUML

The second study involved fourteen participants (three PhD and eleven M.Sc. software engineering students). The main purpose was to evaluate the learnability and usability of OctoUML as well as the role of the voice interaction modality in enhancing the user experience and supporting the software design process. OctoUML got a SUS-score of 74.6 which can be considered a quite good usability score [10]. The majority of the participants stated that it was easy to learn and use the different functionalities of OctoUML (including the voice interaction modality), see Figure 5. Furthermore, the voice interaction modality was perceived helpful in overcoming non-ergonomic tasks e.g. typing via a keyboard.

IV. Conclusion and Future Development

In this paper we presented OctoUML, a prototype of a new generation software design environment for collaborative software design. It provides support for mixing informal hand-drawn elements with formal notations. Moreover, it supports different input methods and interaction modalities.

OctoUML combines the advantages of both informal tools e.g. interactive whiteboards and formal tools e.g. CASE tools, and therefore is able to bridge the gap between early software design process (when designers often sketch their ideas) and formalisation/documentation process. OctoUML was evaluated by conducting two user studies and involving thirty participants in total. The main goal was to get feedback on the viability and usability of OctoUML. The results show that the participants enjoyed their experience with OctoUML and had a positive perception regarding its usability.

The current architecture of OctoUML allows future expansions of the system with additional functionalities. The goal is to implement and incorporate additional features in the subsequent versions of the system:

- *Analysis component*. It will perform software model analysis. This tool will be used to automatically evaluate the created software models to detect general design flaws, security flaws and performance bottlenecks.
- *Versioning component*. The purpose is to provide a repository for keeping track of the version history of stored models, and the ability to observe changes that are made to specific artifacts in the environment. The system should also be able to resolve conflicts when two users change the same model data. Such component would increase the potential for parallel and distributed work, improve the ability to track and merge changes over time, and automate management of revision history. It would also allow multiple designers to work concurrently, supporting tight collaboration and a fast feedback loop.
- *Code management*. Models and code must be combined throughout the development process. Users will be able to generate code from formalized UML class diagrams as well as view models and codes side by side and jump between editing one and keeping the other synchronized.

V. Acknowledgement

We would like to thank Marcus Isaksson, Johan Hermansson, Emil Sundklev and Christophe Van Baalen for their help in the development and evaluation of OctoUML.

References

[1] J. Brooke et al. Sus-a quick and dirty usability scale. *Usability evaluation in industry*, 189(194):4–7, 1996.

[2] M. R. V. Chaudron, W. Heijstek, and A. Nugroho. How effective is uml modeling? *Software & Systems Modeling*, 11(4):571–580, 2012.

[3] M. R. V. Chaudron and R. Jolak. A vision on a new generation of software design environments. In *First Int. Workshop on Human Factors in Modeling (HuFaMo 2015). CEUR-WS*, pages 11–16, 2015.

[4] L. Fowler, J. Armarego, and M. Allen. Case tools: Constructivism and its application to learning and usability of software engineering tools. *Computer Science Education*, 11(3):261–272, 2001.

[5] R. Jolak, B. Vesin, M. Isaksson, and M. R. V. Chaudron. Towards a new generation of software design environments: Supporting the use of informal and formal notations with octouml. In *Second Int. Workshop on Human Factors in Modeling. CEUR-WS*, page : in print, 2016.

[6] S. Lahtinen and J. Peltonen. Adding speech recognition support to uml tools. *Journal of Visual Languages & Computing*, 16(1):85–118, 2005.

[7] M. Magin and S. Kopf. A collaborative multi-touch uml design tool. *Technical reports*, 13, 2013.

[8] N. Mangano, T. D. LaToza, M. Petre, and A. van der Hoek. Supporting informal design with interactive whiteboards. In *Proc. of the SIGCHI Conf. on Human Factors in Computing Systems*, pages 331–340. ACM, 2014.

[9] S. Oviatt and P. Cohen. Perceptual user interfaces: multimodal interfaces that process what comes naturally. *Communications of the ACM*, 43(3):45–53, 2000.

[10] J. Sauro. *A practical guide to the system usability scale: Background, benchmarks & best practices*. Measuring Usability LLC, 2011.

[11] B. Tversky. What do sketches say about thinking. In *2002 AAAI Spring Symposium, Sketch Understanding Workshop, Stanford University, AAAI Tech. Report SS-02-08*, pages 148–151, 2002.

[12] D. Wüest, N. Seyff, and M. Glinz. Flexisketch: A mobile sketching tool for software modeling. In *Mobile Computing, Applications, and Services*, pages 225–244. Springer, 2012.