Towards a New Generation of Software Design Environments: Supporting the Use of Informal and Formal Notations with OctoUML

Empirical Study

Rodi Jolak, Boban Vesin, Marcus Isaksson, Michel R. V. Chaudron
Joint Department of Computer Science and Engineering
Chalmers University of Technology and Gothenburg University
Gothenburg, Sweden
{jolak,vesin,chaudron}@chalmers.se
imarcus@student.chalmers.se

ABSTRACT

Software architects seek efficient support for planning and designing models. Many software design tools lack flexibility in combining informal and formal design. In this paper, we present OctoUML, a proof of a concept of a new generation software design environment that supports an effective software design process. The system provides options for collaborative software design and different input methods for creation of software models at various levels of formality. The design and architecture of OctoUML are also presented. The evaluation shows that OctoUML provides a user-friendly environment and has the potential to effectively support the activities of the designers.

CCS Concepts

•**Software and its engineering** → *Model-driven software engineering; Design languages; Unified Modelling Language (UML); System modelling languages;*

Keywords

Sketching; Informal and Formal Desing; Recognition; Multitouch; Design Environment, UML

1. INTRODUCTION

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 [3]. These models should present a description of systems at multiple levels of abstraction and from different perspectives. Therefore, it is crucial to provide software design tooling that provides possibilities for efficient and collaborative development as well as options for creation and evolution of software models and architectures.

Sketches, or depictive expressions of ideas, pre-date written languages by thousands of years [18]. Many designers tend to sketch their initial ideas on the whiteboard [19]. These sketches present an intuitive way to prototype, communicate and record their thoughts. Sketches can facilitate discovery of new objects and foster new design ideas [17]. They effectively support the process of software design and serve designers to inspect and develop one design idea as well as reflect on some other alternatives [12].

On the one hand, whiteboards are commonly used for sketching initial software design, quite often by many people simultaneously [1]. They support informal design and do not constrain the notation being used. However, standard whiteboards lack of integration with subsequent software elaboration tools. Hence, re-modelling is difficult and a time-consuming task. On the other hand, CASE tools (e.g. StarUML, Rational Rose, Enterprise Architect, etc.) provide means to store and modify designs. However, they support one or more formal notations and hence restrictively require designers to use those specific notations for modelling. Actually, designers often sketch and use ad hoc notations that rarely adhere to standards like the Unified Modelling Language UML [14].

In previous work, we presented our vision on a new generation of software design environments [4]. One of the characteristics which we proposed for such environments is that they should be capable of supporting both informal and formal modelling. In other words, they would combine the advantages of both whiteboards and CASE tools, and therefore be able to bridge the gap between early design process (when designers often sketch their ideas) and formalization/documentation process. To realize our vision, we developed a software design environment called OctoUML[1]. This environment can be run using a number of input devices ranging from desktop computers over large touch-screens to large electronic whiteboards. It allows simultaneous creation of both informal freehand sketches (using fingers or styluses) and formal computer-drawn notations, e.g., UML class diagrams. We have enriched our tool with some features, functionalities and services in order to support designers' activ-

© 2016 ACM. ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

[1]demo video: https://goo.gl/PmuUf8

ities and provide a practical and inspiring user experience. Further details on these aspects will be described in Section 3. To assess our concept, we asked some subjects to evaluate OctoUML and give feedback on its usability. The following questions are addressed:

- Does our tool provide a usable environment considering issues like ease of use, efficiency and user satisfaction?
- Does support for mixing informal and formal notation better support the software design process?

This paper is organized as follows: section two describes the related work. Illustration of our approach for supporting exploratory and collaborative software design is presented in section three. We provide the design architecture details of OctoUML in section four. Evaluation and results are presented in section five. We discuss the results in section six. Threats to validity are presented in section seven. Finally, we conclude the paper and illustrate our plan for future work in section eight.

2. RELATED WORK

CASE tools support software development activities such as diagramming, code generating and documentation [8]. However, software designers consider such formal tools overly restrictive and limitative of their expressiveness [5, 8, 9]. On the other hand, whiteboards are rather simple to use. In fact, they are frequently used by software designers due to their role in promoting creativity, idea generation and problem solving [6]. E-whiteboards provide the perspective for better software design support by permitting the management, control and maintenance of the contents [11, 5].

Mangano et al. [11] identified some behaviors that occur during informal design. In particular, designers sketch different kind of diagrams (e.g. box and arrow diagrams, UI mockups, generic plots, flowcharts, etc.) and use impromptu notations in their designs. 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. [21] 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 freeform 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 stated that they would be willing to adopt it in practice.

Chen et al. have developed SUMLOW [5], a sketching-based UML design tool for electronic whiteboard technology. It allows the preservation of hand-drawn diagrams and supports the manipulation of them using pen-based actions. UML sketches can be formalized and exported to a 3rd party CASE tool.

Damm et al. [8] conducted user studies in order to understand the practice of software modelling. They observed that designers alternate between whiteboards and CASE tools, extend the semantics of the notations to support the design activities and allow expressiveness, sketch new ideas informally, and actively collaborate when they work in teams. The authors considered that a usable modelling tool should be designed to come across the aforementioned observed behaviours. They developed a tool called *Knight*. *Knight* supports informal and formal modelling using gestures on an electronic whiteboard. In order to achieve intuitive interaction, *Knight* uses composite gestures and eager recognition of hand-drawn elements. Damm et al. showed that informal drawings were temporary and usually erased after producing the formal diagram.

Magin and Kopf [10] 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.

Baltes and Diehl [1] examined the usage of sketches in software engineering activities by conducting an exploratory study in software companies. The results showed that the majority of the sketches were informal, and the purposes of sketches were related to modelling, explaining or understanding. Baltes and Diehl also revealed that the sketches were archived digitally for re-visualization and future use. Like us, they think software design tools should enable informal design sketching.

3. APPROACH

Our motivation for creating OctoUML is to provide a more intuitive, inspiring and efficient tool to support exploratory and collaborative software design. Key innovations of our approach are: (i) enabling users to create and mix both informal hand-drawn sketches and formal computer-drawn notations at the same time on the same canvas, (ii) providing a selective recognition mechanism that is used to transform hand-drawn sketches into formalized contents, and (iii) enabling of multi-user support on a single input device. In the next subsections, we describe those novel aspects in more detail. Table 1 shows the differences between our approach and the related work.

Tool	Notation : Informal(IF)/Formal(F)	Recognition	Multi-touch
Flexisketch	**IF** (hand-drawn)	predict symbols based on incremental learning	N/A
SUMLOW	**IF** and **F**, but not simultaneously	holistic recognition	N/A
Knight	**IF** and **F**, but not simultaneously	eager recognition	N/A
Calico	**IF** (hand-drawn)	beautification of shapes	N/A
OctoUML	mix of **IF** and **F** (simultaneously)	selective recognition	enabled

Table 1: Comparison of OctoUML with some other tools mentioned in the *Related Work* section

3.1 Informality and Formality

Based on studies done by [8, 11, 12], we report some common practice behaviours that occur during software modelling meetings:

- Designers combine informal and formal models.
- Designers often alternate between CASE tools and whiteboards.
- Designers prefer all-purpose sketches which refer to many scenarios over sketches dedicated to a single scenario.

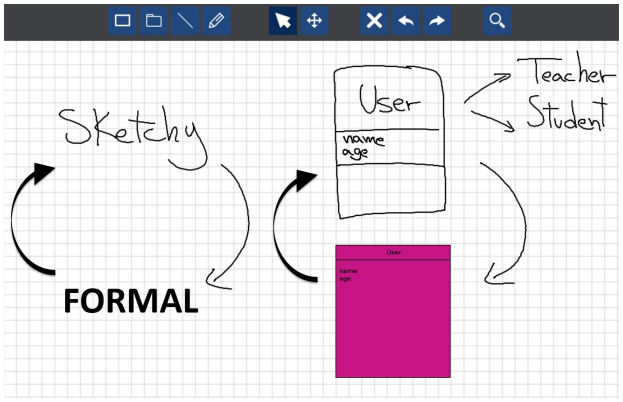

Figure 1: Combination of different notations on the same canvas

- Sketches rarely follow notational convention.
- Sketches are used at different levels of abstraction.
- Designers sketch different types of diagrams with different perspectives.
- Designers extend formal notations in order to explain their ideas to others.

Whiteboards support informal design by ensuring the users a total freedom in creating and using a variety of modelling notations. For example, informal hand-drawn sketches can be used to express abstract ideas representationally, allow checking the entirety and the internal consistency of an idea as well as facilitate the development of new ideas [17]. Informal sketches, as well as various informal tools are used by software developers during their work activities [5, 11, 20]. However, some tasks like model transfer and persistence are difficult and require a redundant work by re-drawing the design solution using a Computer-Aided Software Engineering (CASE) tool. CASE-tools provide a limited set of modelling notations, hence restrict designers' expressiveness by imposing the notation that can be used. Modelling tools should be holistic in order to support software designer's imagination and creativity. To that end, our tool allows a simultaneous creation of both informal and formal notations on the same canvas. The informal notations can be created using free-hand sketches, while the formal notations can be either hand-drawn following a specific syntax or created using computer-drawn ready-to-use elements available in the menu. At the moment, and for the creation of formal elements, our tool mainly supports UML class diagrams, but in the future we aim to support other types of UML diagrams. Figure 1 illustrates the main canvas of our tool. It shows how our tool allows the combination of informal and formal notations on the same canvas. Moreover, it shows how designers can transform the notations from one state to another i.e. from informal to formal and vice versa.

3.2 Recognition

Walny et al. [20] demonstrated that sketches have a lifecycle. In particular, a sketch starts as an informal representation of one idea and later on ends up having a formal representation. To facilitate that process as well as support tasks like: model transfer to third-party CASE tools, code generation and model documentation, our tool supports the transformation of UML hand-drawn elements to formalized computer-drawings and vice versa at any time during the modelling sessions. This has been made available using *PaleoSketch*, a primitive sketch recognition system [13]. There are two aspects that favor the flexibility and elasticity of the recognition process. Firstly, we allow users to select what they want to recognize in advance. Secondly, users can use undo/redo commands in order to move easily between the two forms; sketchy and formal.

3.3 Layering and Multi-touch

Having been inspired by the recent version of the Altova UModel tool [2], we decided to equip our tool with a layering mechanism. In particular, the software design is part of one layer which we call the formal layer. While another layer, the informal layer, contains the informal sketchy elements e.g. hand-written comments, illustrative drawings, highlighting arrows or circles, etc. The user can then select to see combined layers or layers in isolation. A key advantage of such layers is that they allow the isolation of informal and formal elements. As a consequence, designers will be able to move and edit the content of each layer independently without disturbing the rest of the design. For instance, users might want to archive, print or share the formal designs without including the sketchy elements. In that case, the formal layer can be a solution for them. On the other hand, having the two layers combined could help reveal some existing ambiguities in diagrams as well as give more insights to increase one's understanding of concepts, mainly, during diagram reviewing cycles. Baltes and Diehl stated that quite often two or more people are involved in sketching when the whiteboard is used as a medium [1]. We enabled our tool to support multi-touch. Multi-touch is an interaction technique that permits the manipulation of graphical entities with several fingers at the same time. This option allows concurrent collaborative modelling. In particular, it enables two or more designers to simultaneously work on the same canvas of the same device, especially when the device is an interactive whiteboard or a large touch screen.

3.4 Other features

CASE tools are better than whiteboards when we consider some aspects such as undo/redo and re-sizing utilities. For that, we enabled our tool to support the aforementioned features in order to allow designers to easily correct mistakes and have liberty to change the size of the elements. Sometimes designers complain about the limited size of tools' modelling space or canvas which may not be enough to capture all their design ideas. To overcome this inconvenience, the drawing canvas of our tool supports panning and zooming in/out actions. Panning allows users to drag the canvas in all directions in order to find more space for their designs. In addition, zooming helps to change the scale of the canvas, hence to enhance the visibility and readability of the designs.

4. DESIGN

In this section we present the current architecture of OctoUML. The architecture is organized in a way to effectively fit with complex business work-flows, data, and security needs as well as to allow for future integration of different modules and other enterprise applications.

[2]http://www.altova.com/umodel.html

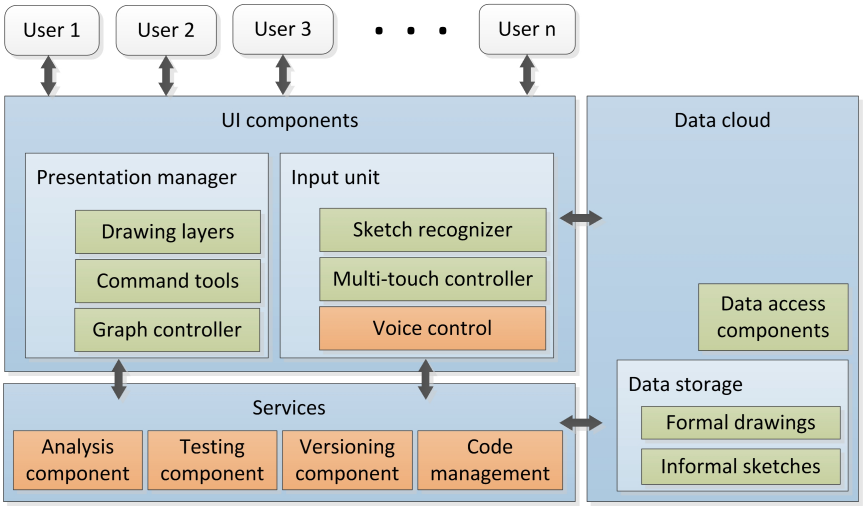

Figure 2: Architectural Components of OctoUML
(currently implemented components are presented in green)

The key architectural components of OctoUML are presented in Figure 2. The environment contains three major components: *UI component*, *Data cloud* and *Services*. The current version of the system offers only the UI and data cloud components. Additional services will be added during future development.

UI component consists of two separated but interconnected parts: *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. Depending on the chosen layer, users are presented with an appropriate toolbar. The *Command tools* are responsible for transferring the inputs from users to different controllers. The *Graph controller* allows switching between different input techniques as well as combining of different 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 cloud*. Our tool uses a set of data structures to manage and maintain the sketched elements, formalized designs, and session control for users. The modelling process and dynamic aspects of the system are presented in Figure 3.

5. EVALUATION

In order to answer the research questions presented in Section 1, we prepared user studies. Sixteen subjects were engaged in a design assignment. The assignment was to create a UML class diagram of a given scenario using our tool. To give a global overview of the subjective assessment of our tool's usability, we asked our subjects to answer the System Usability Scale (SUS) questionnaire [2]. Furthermore, we planned semi-structured interviews using both closed and open questions. The interviews were held after the completion of the design assignment. Our main concern was to get feedback from the participants regarding their experience in using OctoUML, so we focused on qualitative data more than quantitative data. We followed the grounded theory methodology [7], and used *NVivo*[3] in the qualitative data analysis process. More details together with the results are reported in the following subsections.

5.1 Participants and Modelling Expertise

Sixteen software engineering students and researchers were involved in doing the assignment and subsequently the interviews. In particular, there were four master students, ten PhD students and two post-doctoral researchers. Five participants have an experience in industry for some period of time. Twelve people worked on the design task in pairs, while the rest worked individually. Overall, the participants are experts in software modelling and have some experience in using UML. In fact, nine participants claimed to have high expertise in software modelling, five have a moderate expertise, while only two have low expertise. All participants believe that software design is a critical task for successful software development and evolution. In previous occasions, all but one participant did software modelling with other people in teams (collaborative modelling). The participants have a practical experience with a variety of modelling tools. These tools range from whiteboards, pen&paper to CASE tools like *Enterprise Architect*, *Visual Paradigm*, *Dia*, *ArgoUML* and *Papyrus*.

5.2 Design Task

We formulated a simple design problem, and asked the participants to design a domain model class diagram solution of it using OctoUML. Before starting with the task, we gave the participants a brief introduction regarding the features and functionalities of our tool. We directly observed the design processes and took notes. When the participants asked some specific questions about the design assignment,

[3]http://www.qsrinternational.com/

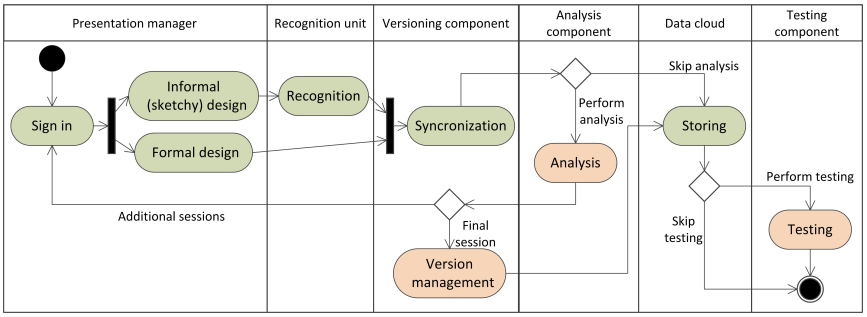

Figure 3: The Modelling Process
(currently supported activities are presented in green)

we told them that it is up to their interpretation. They could, however, ask questions concerning the design environment e.g. how to use certain tools. The participants did not get any help unless they asked for it. An interactive whiteboard was chosen as an input medium, and the design sessions were video-recorded. The text of the design assignment is presented in the following paragraph.

E-Learning System. The system is used by teachers, students and an administrator (who is also a teacher). One teacher is responsible for many courses. Consider that courses consist of many topics. Students can enroll into different courses. There is a news section within the system. Teachers add news for a specific course and the students can read them. Every course ends with an evaluation test. Teachers create a test and the students have to do it. The students get one of these grades: fail, pass, good, or very good.

5.2.1 Design Task Observations

While carrying out the design task, the participants were observed in order to understand their activities. This was done in two different ways: First, we directly observed the behaviour of the participants as they were performing the task and took notes. Second, the participants were recorded via a digital video-camera. This let us observe their behaviour indirectly through records of the task. The notes which we took were expanded by transcribing elements of the video recordings. At the beginning of the task the participants spent around one minute reading the assignment, then they proceeded with designing the solution. While carrying out the design task, some participants first created many UML classes then they associated them with different kind of relations. Other participants followed another strategy by creating two classes in the first place, then, they defined the relation between the two classes before continuing to create other UML classes. Even when two people were working on the same design at the same time, they rarely interacted with the e-whiteboard at the exact same time. Most of the participants tended to create the classes sequentially, and they discussed the properties of one class before proceeding with the creation of another class. Furthermore, they often divided the work between themselves, e.g. one was interacting with the tool to create classes, and the other one was reading the assignment as well as providing ideas for the solution. The participants were given a brief introduction about functionalities of the tool. Nevertheless, during the design task, some of the participants were confused and hesitant about using some features being provided by our tool. This was actually observable at the beginning of the task, but seemed to be overcome later on. In fact, the participants became more confident as they were gradually and progressively interacting with the tool. Moreover repeating some actions, such as selection and creation of classes, let them in some manner experience our environment' functionalities.

5.3 SUS Questionnaire

The *System Usability Scale* is an easy, standard way of evaluating the usability of a system [2]. It is a form containing ten statements, and users provide their feedback on a 5-point scale (1 is strongly disagree and 5 is strongly agree). It effectively differentiates between usable and unusable systems by giving a measure of the perceived usability of a system. It can be used on small sample sizes and be fairly confident of getting a good usability assessment [16]. The participants were given the forms directly after they finished with the design task. We considered the SUS score as a "preliminary feedback" on the usability of our tool. However, in order to consolidate the current findings, more people will be involved in testing our tool and answering the SUS questionnaire.

5.3.1 SUS Result

All sixteen participants filled out the SUS questionnaire. We calculated the SUS score reported by each participant. After that, we calculated the average of the usability values of all participants to obtain the overall OctoUML usability score[4]. The lowest score was 65 and the highest was 95, with an average score of 78.75. It falls at around the 80th percentile, and would result in a grade of a "B" which is a high usability score according to [15].

5.4 Interviews

After answering the SUS questionnaire, each participant was involved in a *semi-structured* interview. The conversations were recorded using a digital voice recorder. The interviewers took some notes which were expanded afterwards by transcribing the audio recordings. Both closed and open questions were used, and respondents' answers were quantitatively and qualitatively analyzed.

5.4.1 Interviews' Results

Several threads run through the interviews. Such threads are categorized as themes and reported subsequently.

[4]https://goo.gl/uwlwIp

5.4.1.1 Tool Usability

Ease of Use. All participants pointed out that our tool is intuitive, simple and easy to use.

> "*Easy to get started with, I do not have to understand the UML-standard*", "*It's simple to use*", "*Easy to change things*"

Learn-ability. The participants also stated that the tool is easy to understand and learn.

> "*easy to understand*", "*Easy to grasp what is what*", "*Intuitive for the most part*", "*So easy to learn*"

Efficiency. We let our environment inherit the fluidity and immediacy of a standard whiteboard. Furthermore, we wanted to maintain the recognition process to be as smooth and fast as possible. Some participants were impressed by the fluidity and immediacy of our tool in drawing and creating notations as well as in recognizing the sketchy elements.

> "*Very quick to draw classes and associations compared to CASE tools*", "*Liked it because it was very fast*", "*Not loading while recognizing, which is good*", "*Very smooth and quick recognition*"

Satisfaction. The basic functions of our tool met the expectations of most of the participants. Overall, the participants liked our tool. Two participant highlighted the eligibility of our tool in collaborative team modelling. One participants stated that he likes the "selective recognition" mechanism.

> "*Very straightforward once you get used to it*", "*It is really good for team design*", "*Nice that you select what to recognize*"

However, some challenges in tool usability were identified. one participant asked for more flexible switching between *informal* and *formal* input modes .

> "*I did not like how I switch between input modes*"

Some of the participants did not like the manner by which the elements are being selected i.e. by activating a dedicated button.

> "*Should select by just clicking on element without selecting the selection tool first*"

One participant stated that "typing-in" classes name and their properties using the virtual keyboard was inconvenient task due to the time that it takes, and asked us to find a better solution.

> "*Typing-in is a bit slow*"

We previously mentioned that our tool mainly supports UML class diagrams when creating formal "computer-drawn" elements. Some participants expressed the need of being able to create other types of UML diagrams as well as having more tool options.

> "*I want more features, options, more types of diagrams*"

We consider those challenges, hence they will be used together with the overall feedback of the users as a basis for future improvement of our tool.

5.4.1.2 Informal and Formal Notation

There was a strong belief among the participants (12 people agreed, 2 were undecided and 2 disagreed) that informal notations (i.e. sketches) support the formal design. Sketches can be used to interconnect different components

> "*define the components involved and the interactions among them*".

Sketches are valuable artefacts beyond being just explorative drawings

> "*obtain already a formal document, not just a sketch*".

There was also a strong belief among participants (11 people agreed, 2 were undecided and 3 disagreed) that having sketches beside the formal design can allow for a better expression of ideas

> "*map out the domain of the functionality and make sure everyhting is been covered*", "*get an idea of how things fit together, also notice bad ideas*".

Eleven people claimed that sketches can enhance the understandability and readability of the formal designs

> "*it is mostly for myself to get a clear understanding*", "*it is helpful to make you understand what you are gonna build*"

On the other hand, one person said that having sketches beside the formal design could complicate the diagram

> "*No, sketching will introduce complexity*".

Half of the participants think that being able to sketch beside creating formal designs in one tool can replace the need of sketching on a whiteboard or a paper.

> "*Yes, you could perform all the functions that you can do on standard whiteboard. Be able to quickly show ideas, and you do not have to take pictures of the whiteboard*".

While some participants claim that still people will not stop using standard whiteboards as well as pen and paper. The dependency on such tools, as the participants argued, arises from their immediacy and ease of use as well as being at easy disposal.

> "*No, because you will never remove the paper from the office*", "*right now still very dependent on paper*"

Sketching down thoughts and ideas on paper or whiteboard when designing software is a common behaviour when designing software. In fact, all participants do sketches to some degree. Some participants mentioned that it depends on the complexity of the problem

> "*I do not often sketch when it comes to simple stuff*".

According to the participants' responses, the main purposes of sketching are to:

i) understand problems and start to explore solutions

"*to define the components involved and the interactions among them*", "*start putting the solution together*"

ii) brainstorm as well as explore ideas

"*to brainstorm*", "*to facilitate how to express our ideas*", "*to visualise what is in my head*".

iii) communicate and discuss their ideas

"*give an explanation for other people about the system or a specific problem*", "*communication of ideas in teams*".

6. DISCUSSION

Before starting with the design task, the participants were given a short introduction about our environment and its functions. However, while we were observing the participants creating their designs, we noticed that they were not very inclined to use the sketching feature which was illustrated during the short introduction. In fact, only six (out of sixteen) participants used the sketching feature. We think that the time of the introduction part was most likely not enough to make the participants feel comfortable in using free-hand sketches. Furthermore, the design problem that we have chosen for the design task was simple and easy to solve. We tried to simplify it as much as possible to make it solvable in a short time. We believe that the simplicity of the design task could have defeated the need of sketching it up. The participants could easily get a good grasp of the design task simply by reading it. However when we did the interviews, the majority of the participants did agree that being able to mix informal and formal notations could support the design process and flow, thus could bridge the gap between the process of prototyping ideas on a paper and the process of entering a formalized version of such ideas in a CASE-tool. Next, we discuss the research questions based on our interpretation of the results.

- RQ.1 Does our tool provide a usable environment considering issues like ease of use, efficiency and user satisfaction?

OctoUML is designed to offer a usable interactive environment. First we focused on understanding some common practice activities that occur during software modelling sessions. Then, we tried to consider and adopt some novel interaction modalities which could interactively support the design process. The interviews' results show that the current version of our tool is easy to learn, effective to use, and provides an enjoyable user experience. Moreover, the results that we got from the SUS questionnaire on the usability of our tool consolidate the previous findings. Of course these results hold only for the device which is used as an input medium, the interactive whiteboard. Other media like tablets and standard PCs have different characteristics. Tablets have a smaller interaction interface when compared to interactive whiteboards, and this could raise some usability challenges. To assess these challenges further tests on different input media are required. On the other hand, our tests revealed some usability challenges related to our system. The main issue is the selection tool. The participants had to click on a specific button to activate the selection mode. According to some participants, that was not a user-friendly choice. Moreover, we asked the participants for their opinion about some new interaction techniques that could be adopted by our environment in the future. These techniques were appreciated by the participants and are discussed in the conclusion and future work section.

- RQ.2 Does support for mixing informal and formal notation better support the software design process?

In practice, software systems are becoming more and more complex, and the design of complex systems needs more effort and hence more sophisticated designing tools. In such cases, having the possibility to use informal notations beside the formal ones can better support designers' activities in understanding the problems, exploring solutions, brainstorming and communicating ideas. This is in line with the study of Mangano et al. [12]. They stated that informal notations, i.e. sketches, allow software designers to discuss design alternatives as well as mentally simulate the behaviour of complex systems.

Apart from the fact that our subjects did not sketch informal elements frequently, the majority of them think that being able to simultaneously create sketches and formal designs in one design environment could support the design process. Indeed, software designers often sketch their early-design ideas on a paper. However, when they want to preserve the design, they do a redundant work by re-drawing the solution using CASE-tools. Furthermore, they may forget to include some sketched ideas in the design formalization process.

There was a strong belief among the interviewees that having the possibility to create informal elements, i.e. sketches, assists the process of ideas expression and enhance the understandability of formal designs. The informal notations can be used both in brainstorming sessions and while creating the formal design. In the former case, they are used for design exploration and can be volatile. While in the latter case, the informal elements could be used to sustain and describe a specific design problem as well as support the formal design in conveying and reinforcing the information that they carry. Informal sketches, for example, may have a very close mapping to the problem domain. As a result, they could be valuable artefacts beyond being just explorative means.

7. THREATS TO VALIDITY

Construct Validity. The design task was simple, specific and easy to do. Moreover, it is relatively small compared to real world design problems. This might limit the creativity of the designers as well as influence the amount of discussions and usability interactions. However, during the interviews, we asked the participants to give their general opinion about the efficiency of our tool when it comes to handling different design problems that vary in size and complexity.

Internal Validity. None of our subjects was familiar with our tool and its functions. To mitigate this, we gave the participants a short introduction explaining the features of the tool. Moreover, during the interviews, the participants might want to please the interviewers by giving them a positive feedback. To mitigate this, we asked the participants to answer the SUS questionnaire which allowed them to give feedback anonymously.

External Validity. The participants being involved in both the design task and the interviews may not represent the general population of software designers. This could threaten the generality of the results. However we involved people with different backgrounds, modelling expertise and academical degrees.

8. CONCLUSION AND FUTURE WORK

Currently, most CASE-tools are modelling (or even diagramming) tools. Indeed, they lack support for the majority of design activities in which developers are engaged. In this paper, we presented a proof of concept of a new generation software design environment. Basically, our tool allows for simultaneous creation of both informal freehand sketches and formal computer-drawn notations. The users of OctoUML can create software designs by performing simple intuitive touch gestures. Moreover, they can manipulate the graphical entities with several fingers at the same time thanks to the *multi-touch* technology being adopted by our system. Furthermore, OctoUML supports the transformation of models from informal to formal at any time during the design sessions. We evaluated our tool by conducting user studies. The results show that OctoUML, as perceived by our subjects, provides a usable environment in terms of ease of use, efficiency and user satisfaction. Moreover, it seems that giving the possibility to create informal and formal notations in one software design environment could support both the design process and its flow.

Future Work. We will continue to realize our vision [4] of a new generation of software design environment. OctoUML will be equipped with microphones to record the spoken discussions, and a recognition system will be provided to interpret users' voice commands. To open up new opportunities for remote interactive collaborative design, our tool will be enabled to support remote collaborative sessions between geographically distributed teams as well as in class room environment between students and teachers. We also aim to integrate OctoUML with other software engineering tools to provide effective support for different development tasks (e.g. requirements gathering, testing, coding and versioning) and analysis tasks (e.g. performance).

9. REFERENCES

[1] S. Baltes and S. Diehl. Sketches and diagrams in practice. In *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 530–541. ACM, 2014.

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

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

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

[5] Q. Chen, J. Grundy, and J. Hosking. An e-whiteboard application to support early design-stage sketching of uml diagrams. In *Human Centric Computing Languages and Environments, 2003. Proceedings. 2003 IEEE Symposium on*, pages 219–226. IEEE, 2003.

[6] M. Cherubini, G. Venolia, R. DeLine, and A. J. Ko. 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*, pages 557–566. ACM, 2007.

[7] J. Corbin and A. Strauss. *Basics of qualitative research: Techniques and procedures for developing grounded theory*. Sage publications, 2014.

[8] C. H. Damm, K. M. Hansen, and M. Thomsen. Tool support for cooperative object-oriented design: gesture based modelling on an electronic whiteboard. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 518–525. ACM, 2000.

[9] J. Iivari. Why are case tools not used? *Communications of the ACM*, 39(10):94–103, 1996.

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

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

[12] N. Mangano, T. D. LaToza, M. Petre, and A. Van Der Hoek. How software designers interact with sketches at the whiteboard. *Software Engineering, IEEE Transactions on*, 41(2):135–156, 2015.

[13] B. Paulson and T. Hammond. Paleosketch: accurate primitive sketch recognition and beautification. In *Proceedings of the 13th international conference on Intelligent user interfaces*, pages 1–10. ACM, 2008.

[14] M. Petre. Uml in practice. In *Proceedings of the 2013 International Conference on Software Engineering*, pages 722–731. IEEE Press, 2013.

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

[16] T. S. Tullis and J. N. Stetson. A comparison of questionnaires for assessing website usability. In *Usability Professional Association Conference*, pages 1–12, 2004.

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

[18] B. Tversky. Visualizing thought. *Topics in Cognitive Science*, 3(3):499–535, 2011.

[19] J. Walny, S. Carpendale, N. H. Riche, G. Venolia, and P. Fawcett. Visual thinking in action: Visualizations as used on whiteboards. *Visualization and Computer Graphics, IEEE Transactions on*, 17(12):2508–2517, 2011.

[20] J. Walny, J. Haber, M. Dörk, J. Sillito, and S. Carpendale. Follow that sketch: Lifecycles of diagrams and sketches in software development. In *Visualizing Software for Understanding and Analysis (VISSOFT), 2011 6th IEEE International Workshop on*, pages 1–8. IEEE, 2011.

[21] 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.