Using Voice Commands for UML Modelling Support on Interactive Whiteboards: Insights and Experiences

Rodi Jolak, Boban Vesin, and Michel R.V. Chaudron

Joint Department of Computer Science and Engineering
Chalmers University of Technology adn Gothenburg University
Gothenburg, Sweden
{jolak,vesin,chaudron}@chalmers.se

Abstract. The ultimate goal of software design tools is to provide efficient support for designing software systems. In our previous work we presented OctoUML, a prototype of a new generation software design environment. OctoUML supports collaborative software design and enables different input methods for creation of software models at various levels of formality. Recently, we added a voice recognition unit into the system and provided users with possibilities of giving voice commands. This paper presents insights and gathers experiences from enabling a voice interaction modality within software design environments. By conducting two user studies, we found that (i) the general perception regarding the usability is positive, (ii) the voice interaction modality is mostly preferred for text input, and (iii) the employment of voice interaction modality within the software design environments enhances the efficiency of the software design process.

Keywords: MDE, software design environments, interaction modalities, voice recognition, accessibility, integration with HCI

1 Introduction

The Unified Modelling Language (UML) is a well-known and commonly used language for software design [22, 24]. Various Computer-Aided Software Engineering (CASE) tools can help software developers to create a UML-model of software. Although these tools can be run today on different devices (e.g. computers, tablets, etc.), the input methods that they support are still limited, and therefore diminishing their usability [17]. Moreover, the interaction with such tools is not always well-designed for user experience, easy learning and effective use. Consequently, many software designers tend to avoid using CASE tools which are considered complex and time-consuming [1, 7].

Multimodal systems use integrated multiple interaction modalities (e.g. sketch, touch, voice, etc.). Oviatt and Cohen [8] illustrated the importance of multimodal systems in reshaping daily computing tasks and predicted their future role in shifting the balance of human-computer interaction much closer to the human. Harris [12] highlighted the importance of voice, as an incorporated modality within multi-modal interfaces, in both opening up a new cognitive dimension and overcoming the barriers of "era-ending" graphical interfaces. The usability of software design environments could be enhanced by introducing voice recognition, as this could terminate the need of using the keyboards and

counteract the matter of menus hierarchy. Yet, the effectiveness could be also improved even further [16].

Interactive whiteboards support touch-input and are useful tools for creating knowledge and sharing information. They increase the learning outcome for students in the classroom [9,11] and enhance the effectiveness and interaction at company meetings [10]. Such capabilities of interactive whiteboards make them potential input devices for software design tools.

In a previous work, we presented our vision on a new generation of software design environments [6]. One of the characteristics which we proposed for such environments is that they should be equipped with microphones to record the spoken discussions, and have a recognition component to interpret users' voice-commands.

Recently, we created a software design environment, called *OctoUML* [15], that provides efficient support for the design of hardware and software architectures. OctoUML can be run using different input devices ranging from desktop computers, over touch screens, to large electronic whiteboards. Beside supporting the creation of software models at different levels of formality, OctoUML is equipped with tools for multi-touch and sketch recognition which enable concurrent collaborative modelling and sketch formalization, respectively. Our current goal is to improve the usability and efficiency of OctoUML by supporting and evaluating new modes of interaction. To this end, we added a voice recognition component into OctoUML. This component is capable of accommodating the most commonly used functions of the system (watch the demo videos[1] [2]).

This paper presents insights and gathers experiences from supporting voice interaction modality within software design environments. The contribution of this work is mainly based on answering and discussing the following three research questions:

- **(RQ1)** For which features of a software design environment do users find it practical to interact through voice commands?
- **(RQ2)** What are the perceptions of users regarding the usability of the voice interaction modality supported by OctoUML?
- **(RQ3)** Does the employment of voice interaction modality within the software design environments enhance the efficiency of the software design process?

To answer these questions, we conducted two user studies. In the first user study, *USS1*, a version of OctoUML without the voice recognition feature was used (*OctoUML-Lite*), while for the second user study, *USS2*, we enabled the voice recognition feature (*OctoUML-V*). Details regarding the two user studies are presented in Section 4.

2 Related Work

A single interaction modality does not allow for an effective execution of tasks and use of environments [18]. Multimodal systems provide possibilities to use a combination

[1] http://goo.gl/JP4Mfg
[2] https://goo.gl/uU6Zm3

of modalities or change to a better-suited modality, depending on the specifics of the task [23].

Like [23], we believe that supporting multiple interaction modalities within software design environments would make the interaction: (i) more intuitive, especially during software design processes when designers discuss and communicate their ideas to each other via voice and gestures, and (ii) more effective by allowing the users to switch to a better-suited modality for the execution of one particular task. Moreover like [8], we believe that multi-modal systems have the potential to shift the balance of human-computer interaction much closer to the human.

Repetitive Strain Injury (RSI) is a condition where pain and other symptoms occur in muscles, nerves and tendons after doing repetitive tasks (e.g. using a keyboard frequently) [28]. Several studies have been carried out to help programmers with RSI by using voice recognition techniques [2, 3, 13]. However, Mills et al. [20] pointed out that such techniques have a high rate of faults and errors which negatively influences their usability.

Lahtinen and Peltonen [17] 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 prototype (called *VoCoTo*) of a speech control system integrated with a CASE-tool (*Rational Rose*[3]). They stated that speech recognition is applicable to be used to enhance the interaction with UML tools. Soares et al. [26] presented a framework, *VoiceToModel*. It allows the creation of requirements models, e.g. conceptual UML class diagram, from speech recognition mechanisms. They evaluated their prototype through an experiment with fourteen computer science students. The users experienced some difficulties in using *VoiceToModel*. However, they were overall satisfied and liked the interaction model provided by the framework.

The objectives of the two aforementioned related works [17, 26] were to assess the viability of their systems and get a *proof of concept* instead of making a statistical proof for their approaches. In this study, we aim to discover which tasks in the software design process are better qualified to be supported by a voice interaction modality. Moreover, we want to assess the efficiency of our approach by comparing two versions of OctoUML; one version is enabled to recognize voice-commands and the other one is not.

Nishimoto et al. [21] designed a multi-modal generic drawing tool that supports speech, mouse, and keyboard inputs. Their tool has a speech recognition system based on Hidden Markov Models [25]. The evaluation showed that their multi-modal drawing tool with speech recognition reduces the required operation time per task, mouse movement, and commands number.

We assess the efficiency of the employment of the voice interaction modality in software design environments by following a similar evaluation approach to [21]. In particular, we measure the amount of time and number of steps that are required to accomplish a specific task using two different input modalities (keyboard/touch and voice) that are supported by OctoUML. Furthermore, we run formative evaluations in order to get feedback on the usability and usefulness of our approach as well as get suggestions for improvement.

[3] http://www-03.ibm.com/software/products/en/rosemod

3 The Software Design Environment: OctoUML

OctoUML was mainly designed to bridge the gap between early software design process, when informal tools (e.g. whiteboards) are typically used for brainstorming and design reasoning, and subsequent formalization process, when formal tools (e.g. CASE-tools) are used for documentation purposes. Such a gap or discontinuity was reported by Budgen [5] in his study on why software design environments do not support realistic design practices. Indeed, ideas and logical basis for the design solution can be easily lost when moving from the early design reasoning process to the formalization process, especially when there is a non-short timespan between the two processes.

OctoUML is a collaborative software design environment that allows the creation and modification of diagrams at different levels of formality. In particular, the environment allows mixing of both informal (e.g. sketches) and formal (e.g. UML class diagrm) modelling notations. Furthermore, OctoUML is equipped with tools for multi-touch and sketch recognition, and supports the transition from informal notations to formal ones. The users of OctoUML are provided with options for moving, resizing, grouping and separating software elements, regardless of their informal or formal character [14].

Figure 1 shows the architecture of OctoUML. The architecture is organized in a way to effectively fit with complex business work-flows as well as to support future integration of different modules and other enterprise applications.

The design 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 developments. The *UI component* consists of two separated but interconnected parts: *Presentation manager* and *Input unit*. The *Presentation manager* provides means for performing stylus or touch-

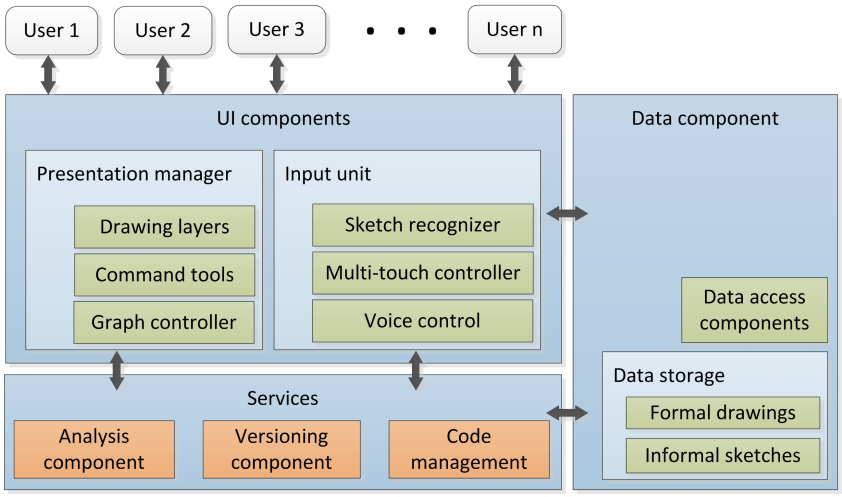

Fig. 1. Architecture of OctoUML (Currently implemented components are presented in green.)

based input commands on devices being used. *Drawing layers* include support for both informal and formal modelling styles. 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, 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. Sketched elements as well as formalized designs are saved and stored in the *Data cloud*. The *Voice control* component is the main focus of this paper and is described in the following subsection.

3.1 Integration of The Voice Control Component

In order to improve the usability of OctoUML and increase its accessibility, 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 manage various elements of software diagrams. The *Voice control* component was implemented using *Sphinx4*. Sphinx4 is an open source voice recognition library developed by Carnegie Mellon University [29].

There are two main types of commands that trigger the *Voice control* component, and therefore allow the interaction with OctoUML via voice:

- Type α: activation/execution of the different interaction tools available on the top of the main canvas (see Figure 2) such as create class, select, undo/redo, etc.
- Type β: assign names to the created packages and classes.

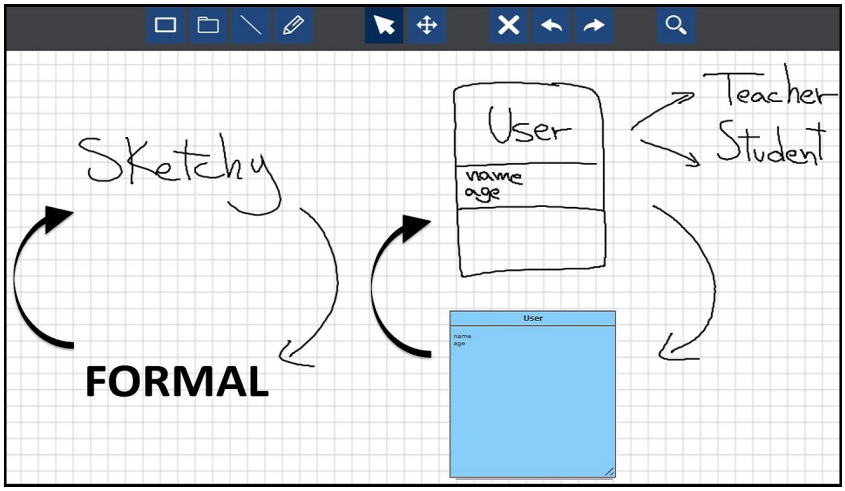

Fig. 2. The main canvas of OctoUML.

For the latter type of commands (β), the current version of the voice control component is based on a predefined dictionary that contains a list of expected words to be used. Table 1 provides more details on the voice commands of types α and β.

Type α	
Command	*Description*
Create Class	selects the class drawing tool
Create Package	selects the package drawing tool
Create Edge	selects the edge drawing tool
Selection Mode	selects the select tool
Moving Mode	selects the moving/panning tool
Undo/Redo	undo/redo actions
Type β	
Command	*Description*
Name	give names to classes and packages. *For instance*: double tap on one class, then say "Name" followed by the desired name.

Table 1. Different types of voice commands.

4 Study

We conducted two user studies to answer the research questions that are presented in Section 1. The two studies were conducted in two different sessions. In the first user study, *USS1*, a version of OctoUML without the voice recognition feature was used (*OctoUML-Lite*), while in the second user study, *USS2*, we enabled the voice recognition feature (*OctoUML-V*). The purpose of carrying out the user studies was to gather experiences, identify faults and discover areas of improvement of OctoUML. Furthermore, we wanted to get and subsequently compare quantitative data regarding the time and steps that are required for the accomplishment of a specific task using the two versions of OctoUML.

USS1 involved fourteen software engineering students (ten PhD and four M.Sc. students) and two post-doctorate software engineering researchers, whereas *USS2* involved fourteen participants (three PhD and eleven M.Sc. software engineering students). All the subjects are familiar with software design and have experience in using the UML. Moreover, they think that software modelling is a critical task for successful software development and evolution (See Figure 3: the range of ratings is [1 to 5] where 1 is the most negative score and 5 is the most positive score).

The subjects have a practical experience with a variety of modelling and design tools. These tools range from whiteboards, pen&paper to CASE-tools like Enterprise Architect[4], Visual Paradigm[5], Dia[6], ArgoUML[7] and Papyrus[8].

[4] http://www.sparxsystems.com/products/ea/
[5] https://www.visual-paradigm.com/
[6] http://dia-installer.de/shapes/UML/index.html.en
[7] http://argouml.tigris.org/
[8] https://eclipse.org/papyrus/

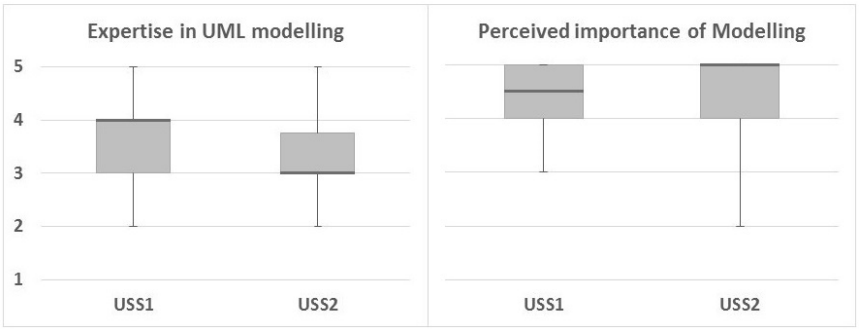

Fig. 3. Expertise in UML modelling and perceived importance of modelling.

We deployed the two versions of our tool *OctoUML-Lite* and *OctoUML-V* on a multi-touch interactive whiteboard (78 1/2" W x 53 3/8" H) that was connected to a Windows 7 PC with a Core Duo 3.00 GHz processor. Later on, each subject was introduced to the functionalities of OctoUML. The introduction lasted around 10 minutes on average. After that, the subjects were given a *software design assignment* which consisted of a short text describing a system to be designed using UML class diagram. The text of the assignment is provided 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. The courses may 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.

The subjects involved in USS2 were also given a sheet of paper containing a list of detailed voice commands. While carrying out the design assignment, the subjects were observed and video-recorded using a digital video camera in order to note and understand their activities. On average, each subject completed the design assignment in 20 minutes. After the completion of the design assignment, we asked the subjects to answer a System Usability Scale (SUS) questionnaire [4] in order to give a global overview of the subjective assessment of the usability of the two versions of OctoUML. SUS can be used on small sample sizes and be fairly confident of getting a good usability assessment [27]. 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, and the data were quantitatively and qualitatively analysed.

5 Results

The results are presented in a form of answers to the research questions that we posed and reported previously in Section 1.

R.Q.1. *For which features of a software design environment do users find it practical to interact through voice commands?*

We asked the subjects who were involved in the user study USS2 to rate the eligibility (suitability) of the supported voice commands of type α (create class, create package, create edge, selection mode, moving mode and undo/redo) and β (name class and name package), together with two additional commands that could be supported in the future: (i) *add attribute/method*, to allow the creation of attributes and methods via voice commands; and (ii) *delete class*, to allow the deletion of selected classes. The results are presented in Table 2, where the scale that is used for the rating ranges from 1 (not important) to 5 (very important).

Voice Commands	**Results**				
	User Study	Median	1st Quartile	3rd Quartile	Inter-Quartile Range
Type α (Creation)	USS2	4.00	3.00	5.00	2.00
Type α (Selection)	USS2	4.00	2.25	5.00	2.75
Type α (Moving)	USS2	4.00	3.00	4.00	1.00
Type α (Undo/Redo)	USS2	4.50	3.25	5.00	1.75
Type α (all)	USS2	4.00	3.00	5.00	2.00
Type β	USS2	4.50	4.00	5.00	1.00
Other Voice Commands	**Results**				
	User Study	Median	1st Quartile	3rd Quartile	Inter-Quartile Range
Add attribute/method	USS2	4.00	4.00	5.00	1.00
Delete Class	USS2	3.00	2.00	4.00	2.00

Table 2. Suitability (eligibility) of different types of voice commands.

We also asked our subjects if there are any other functionalities that are desired to be supported by voice commands. The following list reports these desired functionalities, where every functionality was mentioned by at least one subject:

- naming and changing the type of association,
- save, open, import and export files,
- create a new diagram,
- re-arrange the classes and packages,
- select and deselect classes, packages or edges,
- zoom in and out,
- exit the application,
- define your own voice commands.

R.Q.2. *What are the perceptions of the users regarding the usability of the voice interaction modality supported by OctoUML?*

The perceptions regarding the usability of the two versions of OctoUML (*OctoUML-Lite* and *OctoUML-V*) were collected via: *(i)* the SUS questionnaire and *(ii)* the semi-structured interviews that were run after the completion of the design assignment. As a matter of fact, the collected perceptions reflect satisfaction, comfort and acceptability of use. The results are presented in Table 3. The median is indeed the same for all the measurements concerning the usability of the two versions, except the measurement of the required learning effort to get going with the system (OctoUML-V required more learning effort).

Measurement	**Results**				
	OctoUML version	Median	1^{st} Quartile	3^{rd} Quartile	I-Q.R.[9]
Willing to use the system frequently	OctoUML-Lite	4.00	3.00	4.00	1.00
	OctoUML-V	4.00	3.25	4.00	0.75
Complexity of the system	OctoUML-Lite	2.00	1.00	2.00	1.00
	OctoUML-V	2.00	1.00	2.00	1.00
Ease of use	OctoUML-Lite	4.00	4.00	5.00	1.00
	OctoUML-V	4.00	4.00	4.75	0.75
Need of support to use the system	OctoUML-Lite	2.00	1.00	2.00	1.00
	OctoUML-V	2.00	1.25	2.00	0.75
Integrity of various functions	OctoUML-Lite	4.00	3.00	4.00	1.00
	OctoUML-V	4.00	4.00	4.00	0.00
Inconsistency in the system	OctoUML-Lite	2.00	1.00	2.25	1.25
	OctoUML-V	2.00	1.00	2.00	1.00
Intuitiveness	OctoUML-Lite	5.00	4.00	5.00	1.00
	OctoUML-V	5.00	4.00	5.00	1.00
Cumbersomeness to use	OctoUML-Lite	2.00	1.00	2.25	1.25
	OctoUML-V	2.00	1.00	2.00	1.00
Feeling confident when using the system	OctoUML-Lite	4.00	3.75	5.00	1.25
	OctoUML-V	4.00	3.25	4.00	0.75
Required learning-effort	OctoUML-Lite	1.50	1.00	2.00	1.00
	OctoUML-V	2.00	1.25	2.00	0.75
Ease of using the Voice Interaction Modality (VIM)	OctoUML-V	4.00	3.25	4.75	1.50
Perceived effectiveness of VIM	OctoUML-V	4.00	3.00	4.00	1.00

Table 3. Perceptions regarding the usability of OctoUML-Lite & OctoUML-V

During the semi-structured interviews, various emotional responses and experiences were shared with the interviewers. The subjects enjoyed the experience of using and interacting with the software design environment, especially via voice. Perceptions regarding the simplicity and ease of use of OctoUML were positive, and the subjects valued these aspects when comparing OctoUML to other software design environments that they used previously. The experience of naming the classes in UML class diagram via voice was much more appreciated compared to the experience of using the keyboard to do the same task. OctoUML-V was perceived useful for simplifying the process of class diagram creation and recommended to people with disabilities.

[9] Inter-Quartile Range

R.Q.3.*Does the employment of voice interaction modality within the software design environments enhance the efficiency of the software design process?*

During the semi-structured interviews that were held in the user study USS2, a few subjects (5 out of 14) perceived that the voice commands of type α did not significantly enhance the design process as it was not that big of a reach or hassle for the subject to click on the buttons in order to activate/execute the tools of the software design environment. However the subjects considered this type of commands useful for people with disabilities. While the voice commands of type β were much more appreciated by the subjects who predicted their potential in replacing the use of the keyboard which was perceived a time-consuming task. For that, we only considered the voice commands of type β in the assessment of the efficiency of the employment of the voice interaction modality in the software design environment (OctoUML). To assess the efficiency, we measured the amount of time and number of steps (interactions) that are required for naming classes and packages in all UML diagrams that were created during the two user studies (USS1 and USS2) using OctoUML-Lite and OctoUML-V, respectively. The results are presented in Table 4. They show that the process of naming the classes and packages via voice requires the same number of steps (3 steps: *select* then *name* then *confirm*), but less amount of time with respect to the same process using the keyboard. In fact, the difference in time is significant according to Mann-Whitney's test [19] (p-value is $0.047 < 0.05$). Furthermore, the standard deviation of the naming effort for OctoUML-V (SD = 0.32) is lower than OctoUML-Lite (SD = 1.76), and indicates that the calculated times for naming class diagram elements via *voice* are more closely clustered around the mean (the variance of the required time to name different elements via voice is small).

OctoUML Version	**Number of steps**	**Amount of time (Seconds)**		
		Mean	St.Dev.	Difference in mean rankings
OctoUML-Lite	3	2.12	1.76	*Mann-Whitney* test (p-value)
OctoUML-V	3	1.35	0.32	0.047

Table 4. Naming effort: *keyboard* (OctoUML-Lite) vs. *voice* (OctoUML-V).

6 Discussion

According to our subjects, the better-suited task in software design that should be supported by voice interaction modality is when the user needs the keyboard for text input i.e. naming classes and packages via commands of type β. The main reason is that using a keyboard is not ergonomic and a time-consuming task. In fact, it is easier, faster and more comfortable to use voice instead of typing [26]. When we asked the subjects to rate the eligibility (suitability) of the different voice commands, the voice commands of type α got less suitability score than the commands of type β (α's median is 4.0 against 4.5 for β). Even if such commands are of less suitability to the subjects, their support was strongly recommended because of two reasons: (i) the interaction with the software design environment via voice is more enjoyable than using the traditional way

e.g. keyboard or touch inputs, and (ii) the potential of the voice interaction modality that was perceived by the subjects in supporting people with disabilities. Indeed, some of our subjects wanted to be able to create UML class diagrams by using only voice commands. Of course for such a scenario, every possible feature and functionality of the software design environment is a candidate for voice recognition.

Overall, the perceptions regarding the usability of the two versions are similar. The median is indeed the same for all the measurements concerning the usability of the two versions, except the measurement of the required learning effort to get going with the system (see Table 3). In fact, the required learning effort for using OctoUML-V is more than that of OctoUML-Lite (OctoUML-V's median score is 2 against 1.5 of OctoUML-Lite). This is because the subjects had to learn a list of various voice commands that are necessary for using of the voice-interaction-enabled version; OctoUML-V. We have also noticed the learning effort issue during the software design session. Indeed, before starting with the design session, we supplied the subjects with a sheet of paper (short manual) containing a list of the supported voice commands. At the beginning of the session, the subjects often looked to the manual in order to remember the commands. However, after practicing and getting more used to the commands, the subject learned exactly how to master them without consulting the manual.

We found that the employment of voice interaction modality within the software design environments enhances the efficiency of the software design process by reducing the time required to *name* UML class diagram' classes and packages. However, numerous factors (e.g. the distance of the microphone, white noise, human pronunciation, etc.) may affect the effectiveness or accuracy of the voice recognizer, and as a consequence affect the enhancement in the efficiency of the software design process. This is in-line with Mills et al. [20] who pointed out that voice recognition techniques have a high rate of faults and errors which negatively influences their usability. In order to assess this issue, we counted how many voice commands were used during the user study USS2, and how many times the voice recognizer failed to correctly interpret such commands. In particular, we noted:

- *Unintended Commands*. Happen when a given voice command activates/executes a task different from the desired one.
- *Faulty Name Inputs*. Happen when a class or package gets a name different from the one assigned via a voice command of type β (recognition error).
- *Unrecognized Commands*. Stand for voice commands that could not provoke any consequence, in the sense that OctoUML-V could not interpret and even react to such commands.

The average number of used voice commands is 27 (the lowest is 9 and the highest is 42). Overall, the failure rate for using voice commands is 26% (see Figure 4). Such a rate was obtained via dividing the total number of voice recognition faults (100) by the total number of executed voice commands (381). Whereas the failure rate for using the voice commands of type β (*faulty name inputs'* failure rate) is 12%. This rate is small, however, it still affects the enhancement in the time required for naming the UML class diagram elements. In order to minimize the failure rate, more sophisticated recognizers are needed to reduce the effects of the factors that may compromise the recognition process.

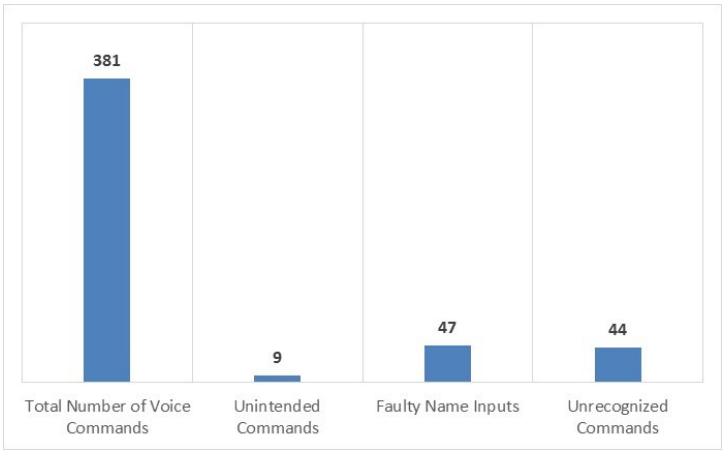

Fig. 4. Usage and faults of voice commands.

7 Threats to Validity

Construct Validity. We assigned a design task which was relatively simple compared to real world problems. This might have influenced the amount of discussions and usability interactions. However during the interviews, all the subjects perceived the potential of OctoUML in managing any kind of software design problems, even complex ones.

Internal Validity. None of the subjects were familiar with OctoUML. To mitigate this, we gave the participants a short introduction explaining the features and functionalities of OctoUML. Moreover, we provided the subjects involved in USS2 with a list of all voice commands that are supported and could be used during the software design session. During the interviews, the participants might have wanted 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 involved subjects in two the user studies, USS1 and USS2, may not represent the general population of software engineering community. This could threat the generality of the results. However we involved people with different background, modelling expertise and academical degrees.

8 Conclusion and Future Work

Modelling is a common approach for software development as it allows efficient definition of software artefacts in order to create a solution that meets the requirements. There is an evident need for efficient methods and tools for designing software products. Current software design tools constrain the realistic design process rather than supporting it [5]. Furthermore, they lack adaptation and deployment of advanced technologies and need flexibility on both platforms and used input methods.

The main goal of this study was to find out which features of software design environments are prioritized to be supported by voice interaction modality, as well as

understand whether the support of such features could: (i) enhance the usability and efficiency of software design environments, and (ii) be of benefit for software design processes. To achieve this goal, we designed and evaluated a multi-modal software design environment, OctoUML, that supports multiple interaction modalities such as touch, mouse, keyboard and voice. Furthermore, we conducted two user studies by involving a population sample of thirty subjects (2 post-docs, 13 PhD and 15 M.Sc. students) to evaluate and compare the usability as well as the efficiency of two versions of OctoUML; one is voice-recognition-enabled (OctoUML-V) and the other is not (OctoUML-Lite).

OctoUML-V was more appreciated by the subjects. Moreover, it also enhanced the efficiency of the software design process by reducing the required time for naming the elements of the software design diagram.

The collected perceptions regarding the usability will be utilized to conduct and track the development progress of OctoUML as more improvements could be done in the future. Furthermore, we will study the impact of employing *multi-touch* and *remote collaboration* techniques in OctoUML, and hence evaluate the usefulness of these techniques in supporting the software design process.

Acknowledgments

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

References

1. M. B. Albizuri-Romero. A retrospective view of case tools adoption. *ACM SIGSOFT Software Engineering Notes*, 25(2):46–50, 2000.
2. S. C. Arnold, L. Mark, and J. Goldthwaite. Programming by voice, vocalprogramming. In *Proceedings of the fourth international ACM conference on Assistive technologies*, pages 149–155. ACM, 2000.
3. A. Begel. Spoken language support for software development. In *Visual Languages and Human Centric Computing, 2004 IEEE Symposium on*, pages 271–272. IEEE, 2004.
4. J. Brooke et al. Sus-a quick and dirty usability scale. *Usability evaluation in industry*, 189(194):4–7, 1996.
5. D. Budgen. The cobbler's children: Why do software design environments not support design practices? In *Software Designers in Action: A Human-Centric Look at Design Work*, pages 199–216. Chapman and Hall/CRC, 2013.
6. 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.
7. N. L. Chervany and D. Lending. Case tools: understanding the reasons for non-use. *ACM SIGCPR Computer Personnel*, 19(2):13–26, 1998.
8. P. Cohen and S. Oviatt. Multimodal interfaces that process what comes naturally. *Commun ACM*, 43(3):45–33, 2000.
9. A. Drigas and G. Papanastasiou. Interactive white boards in preschool and primary education. *iJOE*, 10(4):46–51, 2014.
10. Å. Fast-Berglund, U. Harlin, and M. Åkerman. Digitalisation of meetings–from white-boards to smart-boards. *Procedia CIRP*, 41:1125–1130, 2016.

11. F. Gursul and G. B. Tozmaz. Which one is smarter? teacher or board. *Procedia-Social and Behavioral Sciences*, 2(2):5731–5737, 2010.
12. R. A. Harris. *Voice interaction design: crafting the new conversational speech systems*. Elsevier, 2004.
13. T. J. Hubbell, D. D. Langan, and T. F. Hain. A voice-activated syntax-directed editor for manually disabled programmers. In *Proceedings of the 8th International ACM SIGACCESS Conference on Computers and Accessibility*, pages 205–212. ACM, 2006.
14. R. Jolak, B. Vesin, and M. R. V. Chaudron. Octouml: An environment for exploratory and collaborative software design. In *39th International Conference on Software Engineering. ICSE'17*, page in print, 2017.
15. R. Jolak, B. Vesin, M. Isaksson, and M. R. Chaudron. Towards a new generation of software design environments: Supporting the use of informal and formal notations with octouml. In *Second International Workshop on Human Factors in Modeling (HuFaMo 2016). CEUR-WS*, pages 3–10, 2016.
16. A. E. Lackey, T. Pandey, M. Moshiri, N. Lalwani, C. Lall, and P. Bhargava. Productivity, part 2: cloud storage, remote meeting tools, screencasting, speech recognition software, password managers, and online data backup. *Journal of the American College of Radiology*, 11(6):580–588, 2014.
17. S. Lahtinen and J. Peltonen. Adding speech recognition support to uml tools. *Journal of Visual Languages & Computing*, 16(1):85–118, 2005.
18. J. Larson, S. Oviatt, and D. Ferro. Designing the user interface for pen and speech applications. In *CHI'99 Workshop, Conference on Human Factors in Computing Systems (CHI'99)*, 1999.
19. H. B. Mann and D. R. Whitney. On a test of whether one of two random variables is stochastically larger than the other. *The annals of mathematical statistics*, pages 50–60, 1947.
20. S. Mills, S. Saadat, and D. Whiting. Is voice recognition the solution to keyboard-based rsi? In *2006 World Automation Congress*, pages 1–6. IEEE, 2006.
21. T. Nishimoto, N. Shida, T. Koayashi, and K. Shirai. improving human interface drawing tool using speech, mouse and key-board. In *Robot and Human Communication, 1995. RO-MAN'95 TOKYO, Proceedings., 4th IEEE International Workshop on*, pages 107–112. IEEE, 1995.
22. A. Nugroho and M. R. Chaudron. A survey into the rigor of uml use and its perceived impact on quality and productivity. In *Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement*, pages 90–99. ACM, 2008.
23. S. Oviatt, P. Cohen, L. Wu, L. Duncan, B. Suhm, J. Bers, T. Holzman, T. Winograd, J. Landay, J. Larson, et al. Designing the user interface for multimodal speech and pen-based gesture applications: state-of-the-art systems and future research directions. *Human-computer interaction*, 15(4):263–322, 2000.
24. M. Petre. Uml in practice. In *Proceedings of the 2013 International Conference on Software Engineering*, pages 722–731. IEEE Press, 2013.
25. L. Rabiner and B. Juang. An introduction to hidden markov models. *IEEE ASSP magazine*, 3(1):4–16, 1986.
26. F. Soares, J. Araújo, and F. Wanderley. Voicetomodel: an approach to generate requirements models from speech recognition mechanisms. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, pages 1350–1357. ACM, 2015.
27. T. S. Tullis and J. N. Stetson. A comparison of questionnaires for assessing website usability. In *Usability Professional Association Conference*, pages 1–12, 2004.
28. M. Van Tulder, A. Malmivaara, and B. Koes. Repetitive strain injury. *The Lancet*, 369(9575):1815–1822, 2007.
29. W. Walker, P. Lamere, P. Kwok, B. Raj, R. Singh, E. Gouvea, P. Wolf, and J. Woelfel. Sphinx-4: A flexible open source framework for speech recognition. 2004.