Program Satisfaction Based on the Perception of Bugs as Features

Luke Badini

Advisor: Matt Anderson

March 15, 2017

Abstract

Bugs are an ever-present problem facing software developers. What if you could mitigate some of the user dissatisfaction associated with a bug by telling your users that the bug is actually a feature of the program? In this paper, I describe an experiment in which I tested whether telling a user that a bug is a feature affects their satisfaction with using the program. I had participants draw three UML diagrams using a modified version of ArgoUML and had them report their satisfaction with using the program. The result of this experiment was that you should not tell users about bugs in your program.

1 Introduction

The user interface is an important aspect of a program. Ideally, the users of a program should not have to deal with an unresponsive and buggy user interface. In fact, user interfaces are so important that nearly 50% of the code for computer programs is devoted to the user interface [7]. However, programmers are not perfect and there is a high possibility that their code will contain bugs. If these bugs are serious enough, users will notice them and potentially be unsatisfied when using the program. From a developer's standpoint, one does not want users to be unsatisfied with the program. It would be nice to find and correct bugs before users encounter them, but that is not always possible – locating and fixing the bug takes time. There is often a period of time between when a bug is detected and when it is fixed where users have to use a buggy program.

How much does telling a user that a bug is a feature affect their satisfaction with a program?

During the time when a user is exposed to a bug, it would be nice if a developer could mitigate the

dissatisfaction the user feels. I investigate whether telling the user that a bug is a feature increases user satisfaction of a buggy program.

For my experiment introduced several bugs into ArgoUML and told one group of users that bugs are features, and did not tell another group about the bugs. I also had a control group that used an unmodified version of ArgoUML. ArgoUML is an open source UML editor programmed in Java. The hope was that one group will think that the bug is a feature, and the other group will think it is a bug. Following this, I had each group draw three UML diagrams using the modified version of ArgoUML. Once they had drawn all three diagrams, they filled out a survey about their overall satisfaction with using ArgoUML. Further discussion of UML and ArgoUML can be found in Appendix A and Section 5.1 respectively.
There were two possible hypotheses I had for my experiment regarding user satisfaction:

1. The feature group reports a higher user satisfaction than the bug group.

2. The bug group reports a higher user satisfaction than the feature group.

Hypothesis 1 is the more ”obvious” outcome to me. It seemed more likely that the feature group would ignore the bugs in the UI and report higher user satisfaction scores than the bug group. However, there was the possibility that the feature group could overcompensate for the fact that I told them UI is slow and that the bug group would not find the slowness of the UI to be very problematic. This would have lead to the bug group reporting higher user satisfaction scores than the feature group.

Another hypothesis I had was that the feature group would complete their tasks more quickly and successfully than the bug group. I believed that since I did not tell the bug group about the slowness of the program, they would be more apprehensive and confused when the program did not behave in a way they thought it should. As such, I believed that they would take more time to create their diagrams, and possibly not be able to complete them within the time limit.

Ultimately, I was not able to prove either of my hypotheses with the data I collected. However, the results suggest that it might not be the best idea to tell users about a bug in your program.

In the following sections I first discuss background information about user interfaces, bugs, and user satisfaction. I then discuss ArgoUML and the modifications I made to it. Next, I discuss my experiment and the data I gathered. I then discuss the analysis of the data I gathered and present my conclusions.

2 User Interface

I believe that creating a responsive and intuitive interface has been, and continues to be, an important issue software developers face. A survey conducted by Myers et al.[7] shows that an average of 48% of

the code for a program is devoted to the user interface. Further, 45% of the time spent during the design process is spent designing the user interface. This suggests that software developers believe that the user interface is a very important aspect of a program. In addition to this, the two most difficult aspects of designing a user interface that developers reported were figuring out what the users wanted and designing an interface to accommodate all levels of users [7]. Because of this, I believe that developers spend so much time designing and programming the user interface in order for users to have a positive experience using their program.

There has been research done in what aspects of a user interface appeal to users. A study done by Gajos et al. [4] explored how three types of interfaces influenced user satisfaction. The interfaces used were a split interface, a moving interface, and a visual popout interface. The split interface implemented an additional toolbar that contained important functions. The moving interface moves functions from a pop up window into the main toolbar. The popout interface highlights important functions in a popout window. They concluded that the split interface produced better results compared to the baseline and visual popout interfaces, but compared to the moving interface there was no significant difference. I took the results of this study into consideration while designing my experiment in order to figure out what kinds of changes to the user interface impacted users negatively.

3 Bugs

A bug is a flaw in a program that causes the program to behave in an unintended way. Most of the time, these bugs are a result of a problem with the source code. I believe that a program should be as bug-free as possible in order to maximize how satisfied a user is when using the program.

There has been research done in bug classification. A study done by Herzig et al. [5] explored using machine learning algorithms to see how misclassifying bugs as features affects bug prediction. The authors manually went through bug reports for five open-source JAVA projects. Following this, they created a list of classification rules for bug reports and attempted to re-classify old bug reports using the new classification rules. The authors concluded that their machine learning algorithm predicts bugs as features too often and that bug checking should be done by a human. This study suggests that it is difficult to convince humans that a bug is a feature. This implies that with my experiment, it may be difficult to convince the feature group that the slowness of the UI is actually a feature rather than a bug. This could potentially lead to the results of the feature group being skewed since some participants might not be convinced enough that the slowness is a feature. However, this means that the bug group will be more likely to perceive the slowness of the UI as a bug rather than a feature. This is favorable because the results of the bug group will potentially

be more uniform in respect to how they rate the responsiveness of the user interface.

4 User Satisfaction

There are many different ways of measuring user satisfaction. One is by conducting a survey asking users to evaluate certain aspects of the program or system. An example of this can be seen in Crescenzi et al. [3] where participants from Amazon Mechanical Turk were presented with an information-seeking task using the authors' search interface. After completing the task, the participant was given a survey. The results of this survey were then compiled and used to calculate overall user satisfaction in using the search interface. A similar example can be seen in Paschali et al. [8]. The authors compiled a list of factors in video games that influence user satisfaction with a game. They then created a survey in which they asked gamers which of these factors mattered the most to them. The results of this survey were then compiled and used to measure which factors influenced user satisfaction the most according to gamers. Using the results of the surveys, one can measure user satisfaction based on feedback from actual users.

5 ArgoUML

ArgoUML is an open source UML editor programmed in Java (c.f., e.g., [1]). It is one of the most popular UML editors, boasting over 80,000 downloads. ArgoUML currently has support for creating class, state, use case, activity, collaboration, deployment, and sequence diagrams. Refer to Appendix A for a discussion of these diagrams. I used ArgoUML for my experiment because it is open source and I believed it would be easy modify. However, it turned out to be very challenging to implement my changes.

5.1 ArgoUML Modifications

For the first part of my research, I modified a few behaviors of ArgoUML:

1. The responsiveness of the hover-over tooltips.
2. The responsiveness of clicking to make a class object.
3. A random display shift when drawing objects to the screen.

In order to modify the time it takes for a hover-over tooltip to pop up, I changed the input of a function named *ToolTipManager.setInitialDelay()*. For example, if I wanted to make the delay be 200ms, I would give the *setInitialDelay()* 200 as input. In order to modify the time it takes to click to make a class object, I needed

to add a call to Thread.sleep() in the constructor for the class object. To change the accuracy of clicking to make a class object, I needed to add a number to the x and y variables inside the method that displays the class objects to the screen. For my experiment, I set the tooltip delay to 2000ms, I set the click delay to 1000ms, and I set a graphical distortion to draw figures at a random offset between -10 and 10 units. I originally thought these units were pixels but that was not the case (the documentation does not describe what they are either). In an experiment done by Kosinski and Cummings [6], students were given a list of letters and asked to press the spacebar when any of the letters on their list appeared on screen. The time between when the letter appeared on screen and when the student pushed the spacebar was recorded as their recognition time. The average recognition time among students was 384 milliseconds. Based on this, I decided to make my delay much larger than this value so that my users would almost certainly notice the delay.

5.2 Challenges of Modifying ArgoUML

There were two major problems I faced in modifying ArgoUML. The first was that a lot of the code was poorly documented (see Figure 1 for an example). When I was trying to find where in the code to put my modifications, I had to do tedious testing to see if the function I was looking at was actually the right function. Additionally, the source code was poorly structured. As an example, see the function *drawRect* in Figure 2. It would appear that this function draws rectangles; unfortunately it does not. Because of the poor documentation of the code, I was not able to figure out what *drawRect* does. There are many more examples of this throughout the source code which made it difficult for me to find the functions I needed to modify in order to accomplish what I wanted. Due to these two factors, I was not able to modify ArgoUML exactly the way I wanted to. Originally, I wanted to displace clicking the mouse. However, the result of me trying to add that was the graphical distortion of the class rectangle seen in Figure 4.

```
/**
 * @param fgVec the FigGroup
 * @param ft    the FigText
 * @param i     get the fig after fig i
 * @return the FigText
 */
protected FigText getNextVisibleFeature(FigGroup fgVec, FigText ft, int i) {
```

Figure 1: An example of the documentation of some functions in the ArgoUML source code.

```
public void drawRect(final Graphics g, final boolean filled,
        final Color fillColor, final int lineWidth, final Color lineColor,
        final int x, final int y, final int w, final int h,
        final boolean dashed, final float dashes[], final int dashPeriod) {
```

Figure 2: An example of a function in ArgoUML named drawRect. This function does not draw rectangles.

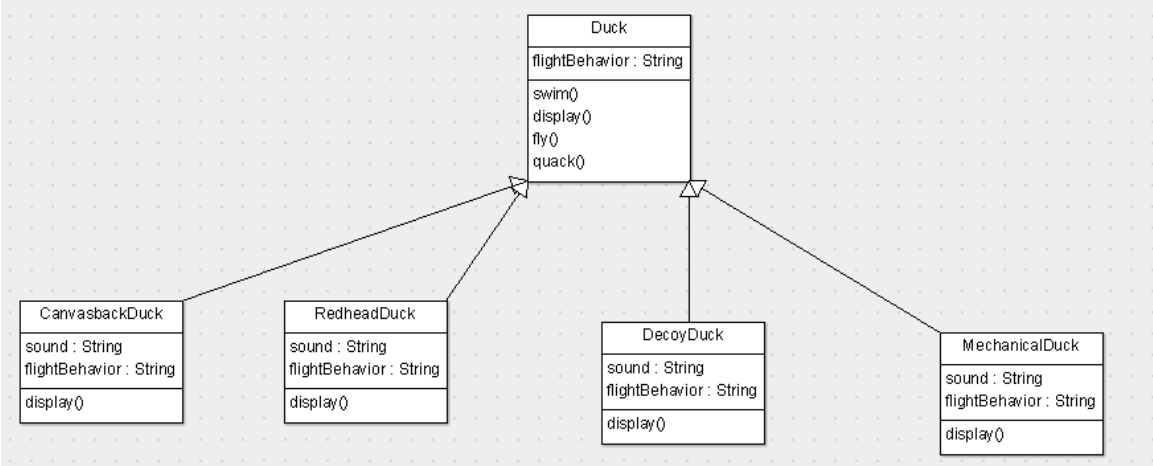


Figure 3: An example of a UML class diagram that the participants made using the unmodified version of ArgoUML.

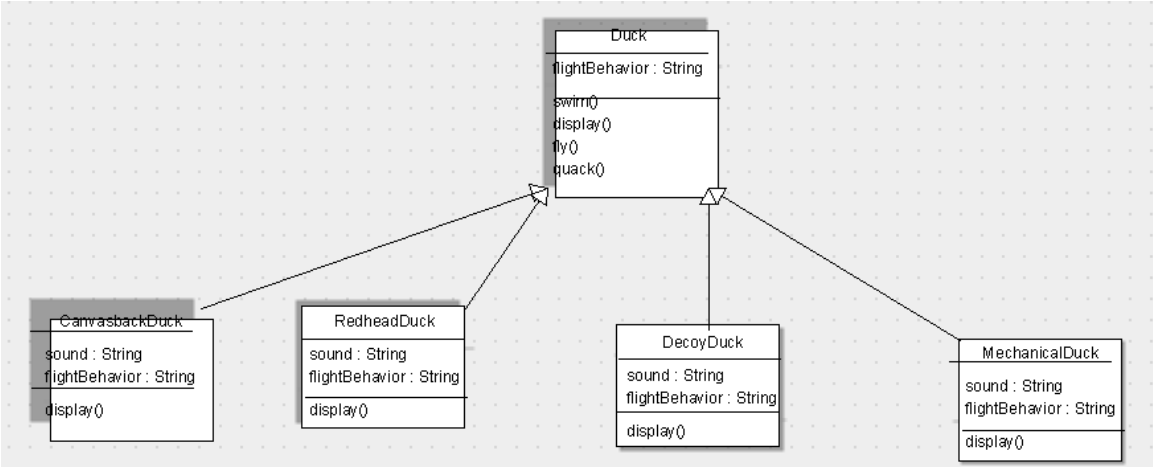


Figure 4: An example of a UML class diagram that the participants made using the modified version of ArgoUML.

6 Experiment

For my experiment, I had participants draw UML diagrams using ArgoUML. A control group was given an unmodified version of ArgoUML and two experimental groups were given the version of ArgoUML I made modifications to. Each participant was then given a tutorial of how to use ArgoUML and completed the tutorial using the version of ArgoUML they were assigned (see Appendix B for the full tutorial). Following this, each participant drew three UML diagrams and was given a survey once they were done. Sixteen student volunteers aged 18 to 22 from Union College participated in my study. Six participants had previously taken CSC-260 Large-Scale Software Design. I conducted my experiments in the HCI lab in CROCHET. There were three groups of participants:

1. **Control:** I presented the participants with the vanilla version of ArgoUML and told them that I was examining the effects of optimizing the user interface.

2. **Bug:** I presented the participants with the modified version of ArgoUML and gave them the same explanation as the control group.

3. **Feature:** I also presented the participants with the modified version of ArgoUML but attempted to convince them that the bug was a feature.

I kept a tally of how many trials of each group I had already run. When deciding which group to assign a new participant to, I randomly picked one from the group that had the lowest number of participants (e.g. if group 1 had 2 trials, group 2 had 2 trials, and group 3 had 3 trials, I would randomly pick either group 1 or group 2 for the new participant). This was to ensure that each group has close to an equal number of trials.

For my experiment, I presented the participants with a consent form and a brief description of an experiment which gave a description of UML and ArgoUML. Following these two documents, I gave the participant a paper tutorial on how to use ArgoUML and asked them to complete it. I then asked the user to make three class diagrams with the appropriate version of ArgoUML. While the participant was making the class diagram, I kept track of how long it took them to make each diagram. There was also a maximum time limit of 10 minutes on each of the diagrams, however no participant ever reached this maximum. Once they had made all the class diagrams, each completed a survey (see Appendix D for the full survey).

7 Data

The main source of data I collected was the responses from the participant surveys. In this survey, each participant was asked to rate various aspects of ArgoUML on a scale of 1 to 5 as well as answer a couple of yes or no questions (the full survey can be found in Appendix D). From the survey, I examined the responses for five questions:

1. When making a new class, how responsive was clicking the mouse?
2. When hovering over something, how responsive were the tooltip popups?
3. How accurate was clicking to make a new class?
4. How frustrating was it to use ArgoUML?
5. How satisfied were you with the overall usability of ArgoUML?

The average values of these responses for each group can be seen in Figures 5 and 6. I only examined five questions because the other question of my survey were not relevant to my experiment. This was to prevent the participant from guessing the intent of my experiment from the survey questions.

I also examined some of the responses to question 11. A few samples can be seen below.

- ”Stop making everything move every .2 seconds pls.”
- ”Make clicks more responsive.”
- ”Increased responsiveness and accuracy of the clicks.”

In addition to these, I recorded how long it took each participant to draw each of the three diagrams. The average values for these times can be seen in Figure 7.

	Click: Response	Tooltip: Hover	Click: Accuracy	Frustration	Satisfaction
Control	3.8	2.6	3.6	2.8	3.2
Feature	3.0	1.0	2.2	4.6	1.4
Bug	3.8	2.5	2.8	3.7	2.7

Figure 5: A table containing the average response values for each of the five questions.

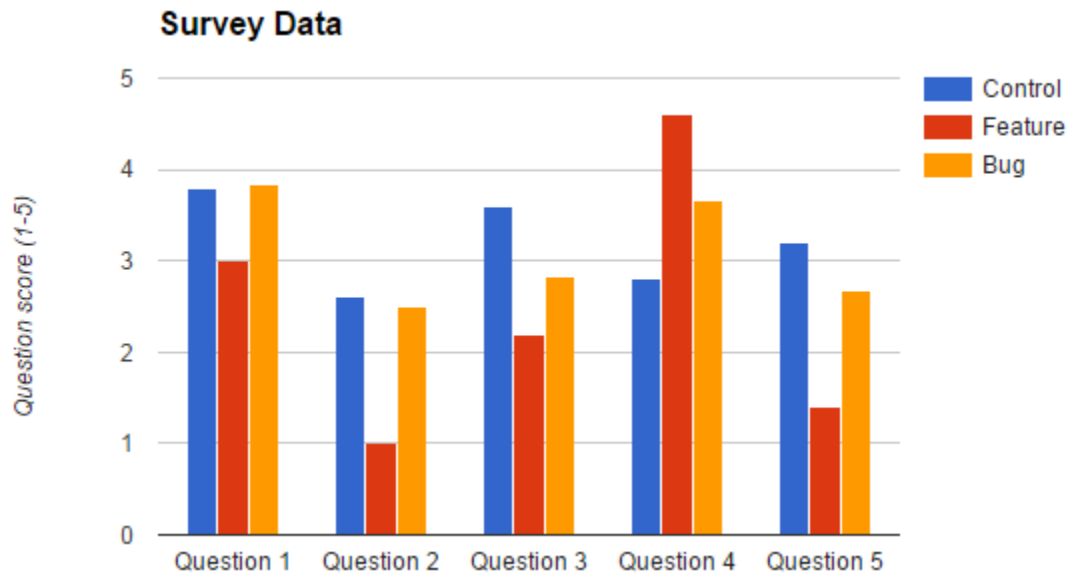


Figure 6: The average response values for each of the five questions I was interested in.

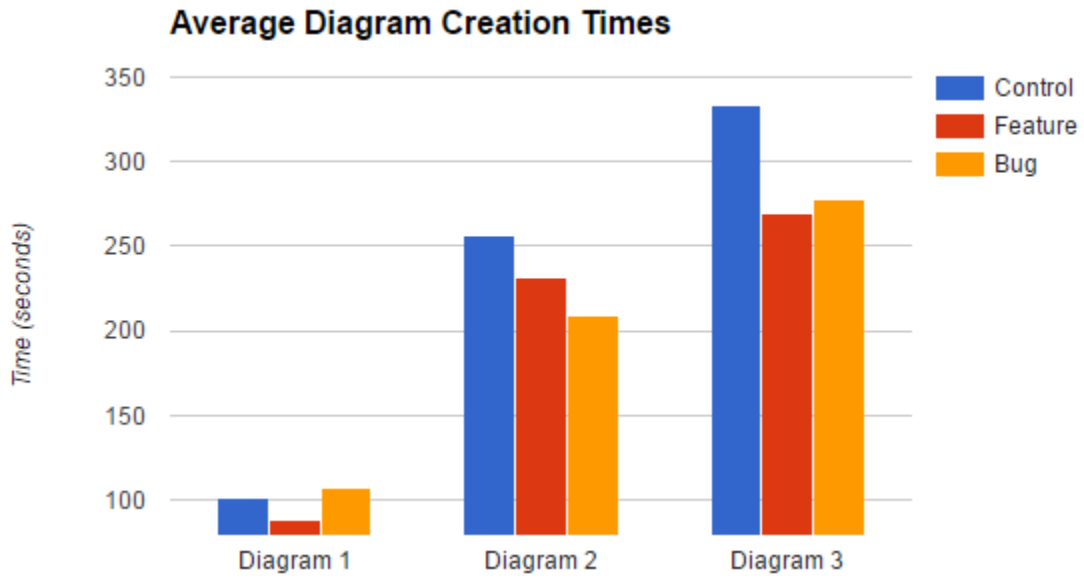


Figure 7: The average completion time values for each of the three UML diagrams.

8 Analysis

I calculated the average for the five questions mentioned in Section 7 for the control, feature, and bug groups (see Figures 5 and 6). In addition, I grouped the responses for these five questions into bucketed

tables based on the distribution of the scores. Each table contained three buckets of scores for the given question, the distribution of those responses for the control, feature, and bug groups, and the expected frequency for each response. Using these values, I calculated two-sample chi-square values for control versus feature, control versus bug, and feature versus bug. Following this, I used the chi-square values to compute the p-value for each of these cases using the table provided by Medcalc [2].

	Control vs. Feature	Control vs. Bug	Feature vs. Bug
Click responsiveness	0.975 - 0.20	0.975 - 0.20	0.975 - 0.20
Tooltip responsiveness	**0.02 - 0.01**	0.975 - 0.20	**0.01 - 0.005**
Click accuracy	**0.05 - 0.025**	0.20 - 0.10	0.975 - 0.20
Frustration	**0.025 - 0.02**	0.10 - 0.05	0.10 - 0.05
Satisfaction	0.10 - 0.05	0.975 - 0.20	0.20 - 0.10

Figure 8: A table containing the p-values for the 5 questions I examined.

Given the p-values from Figure 8, there are four values that are statistically significant at the 5% significance level:

1. Tooltip responsiveness for the control group versus the feature group
2. Tooltip responsiveness for the feature group versus the bug group
3. Click accuracy for the control group versus the feature group
4. Frustration for the control group versus the feature group

I also performed two-sample t-tests on the time data for control versus feature, control versus bug, and feature versus bug. There was no statistically significant data for any of these tests.

9 Results and Conclusions

Given the data I collected, there are a few conclusions I am able to make. First, the results of the t-tests on the time data suggests that the modifications I made to ArgoUML had no significant effect on how long it takes a user to draw a UML diagram using ArgoUML. Next, there were the four areas where the survey data produced statistically significant results: **(i)** tooltip responsiveness for the control versus feature group, **(ii)**

tooltip responsiveness for the feature group versus the bug group, **(iii)** click responsiveness for the control group versus the feature group, and **(iv)** frustration for the control group versus the feature group.

The first result suggests that the feature group was able to notice the decreased tooltip responsiveness I added to ArgoUML. However, the bug group was not able to notice the decreased tooltip responsiveness. This could be due to the users not using the tooltips often or not hovering over objects for a long enough time.

The second result suggests that telling the user that the tooltips were slow made them less satisfied with the responsiveness of the tooltips. This is interesting because I hypothesized the opposite would be the case. This result is possibly due to users expecting the tooltip responsiveness to not be delayed much and my modification made the tooltips too unresponsive. This gap between the user's expectations and reality might have caused for the feature group to report a lower average score for tooltip responsiveness than the bug group.

The third result suggests that the feature group was able to notice the decreased clicking accuracy I added to ArgoUML. However, the bug group was not able to notice the decreased clicking accuracy. This could be due to the participants across all groups not fully understanding the question in the survey due to my wording of the question. In addition to this, the modification I implemented was not exactly click accuracy – it was a graphical distortion that was not intended. Due to this, the question in my survey may have been misleading to the participants.

The fourth result suggests that the feature group was more frustrated with using the modified version of ArgoUML. However, the bug group was not noticeably more frustrated than the control group. Again, this could be due to the feature group's gap between their expectations and reality.

Unfortunately, there is not enough statistically significant data to be able to fully prove either of my hypotheses. However, the first result from the survey data is a step in the right direction. This result suggests that maybe you should not tell users about bugs in your program. Since users were less satisfied when I told them about the delayed tooltip responsiveness, it may be possible to prove that this decreased their satisfaction with using the program.

10 Future Work

My experiment is possibly lacking in two areas: not having enough data and poor experiment design. The first problem is that I only had 16 participants. This is a very small set of data. As such, performing this experiment again with significantly more participants may result in a different outcome.

As for the second problem, this is likely due to what was suggested by Herzig et al.[5], that it is difficult

to convince humans that bugs are features. As such, it is possible that the bug I implemented was too obtrusive. In other words, it was too difficult to properly convince the participants in the feature group that my modifications were actually a feature. Alternatively, I did not try hard enough to convince them it was a feature. In the future, implementing a better bug would make it easier to convince users and lead to better results.

References

[1] ArgoUML. http://argouml.tigris.org/, 2009.

[2] MedCalc. https://www.medcalc.org/manual/chi-square-table.php, 2017.

[3] Anita Crescenzi, Robert Capra, and Jaime Arguello. Time pressure, user satisfaction and task difficulty. In *Proceedings of the 76th ASIS&T Annual Meeting: Beyond the Cloud: Rethinking Information Boundaries*, ASIST '13, pages 122:1–122:4, Silver Springs, MD, USA, 2013. American Society for Information Science.

[4] Krzysztof Z. Gajos, Mary Czerwinski, Desney S. Tan, and Daniel S. Weld. Exploring the design space for adaptive graphical user interfaces. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, AVI '06, pages 201–208, New York, NY, USA, 2006. ACM.

[5] Kim Herzig, Sascha Just, and Andreas Zeller. It's not a bug, it's a feature: How misclassification impacts bug prediction. In *Proceedings of the 2013 International Conference on Software Engineering*, ICSE '13, pages 392–401, Piscataway, NJ, USA, 2013. IEEE Press.

[6] Bob Kosinski and John Cummings. The scientific method: An introduction using reaction time. pages 219–234, 1999.

[7] Brad A. Myers and Mary Beth Rosson. Survey on user interface programming. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '92, pages 195–202, New York, NY, USA, 1992. ACM.

[8] Maria Eleni Paschali, Apostolos Ampatzoglou, Alexander Chatzigeorgiou, and Ioannis Stamelos. Non-functional requirements that influence gaming experience: A survey on gamers satisfaction factors. In *Proceedings of the 18th International Academic MindTrek Conference: Media Business, Management, Content & Services*, AcademicMindTrek '14, pages 208–215, New York, NY, USA, 2014. ACM.

[9] Alan Shalloway and James R. Trott. *Design Patterns Explained*. Pearson Education, Inc., Massachusetts, USA, 2005.

A Unified Modeling Language

The Unified Modeling Language (UML) is a visual language used to create models of programs (c.f., e.g., the textbook [9]). A class is a template for creating objects that contains variables and methods (refer to Figure 9). Class diagrams describe the classes used in the system and show the relationships between classes. Each rectangle in a class diagram represents a class. The components of a class are (1) the name of the class, (2) the data members of the class, and (3) the methods of the class. The accessibility of a member of a class is denoted in one of three ways: (i) public members are denoted by a plus sign (+) (ii) protected members are denoted by a pound sign (#), and (iii) private members are denoted by a minus sign (-).

The four main relationships between classes are aggregation, composition, inheritance, and dependency. These relationships are represented by different types of arrows, as shown in Figure 10. Aggregation is when one object is contained within another object (e.g. an **Airport** has many **Aircrafts**). Aggregation is represented by an arrow with an open diamond at one end. Composition is when one object is entirely made up of other objects (e.g. a **Car** is made up of **Tires** and other parts). Composition is represented by an arrow with a shaded diamond at one end. Inheritance is when one class is a sub-class of another class (e.g. **Helicopters** are a type of **Aircraft**). Inheritance is represented by an arrow with an open triangle at one end. Dependency is when one class requires or depends on the elements of another class (e.g. a **Car** uses a **GasStation**). Dependency is represented by a dashed arrow.

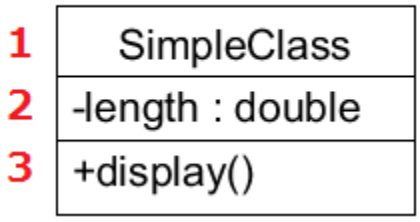


Figure 9: An example of a class in UML. Adapted from Figure 2-1 in Design Patterns Explained [9].

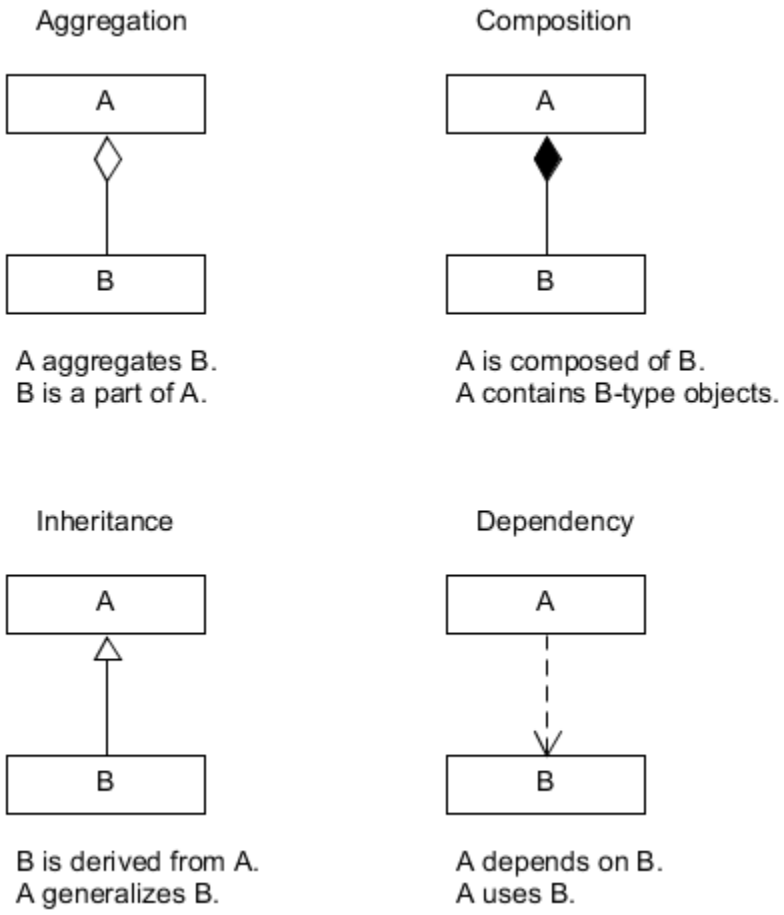


Figure 10: An example of how to show relationships between different classes. Adapted from "UML Notation for Relationships" in Chapter 2 of Design Patterns Explained [9].

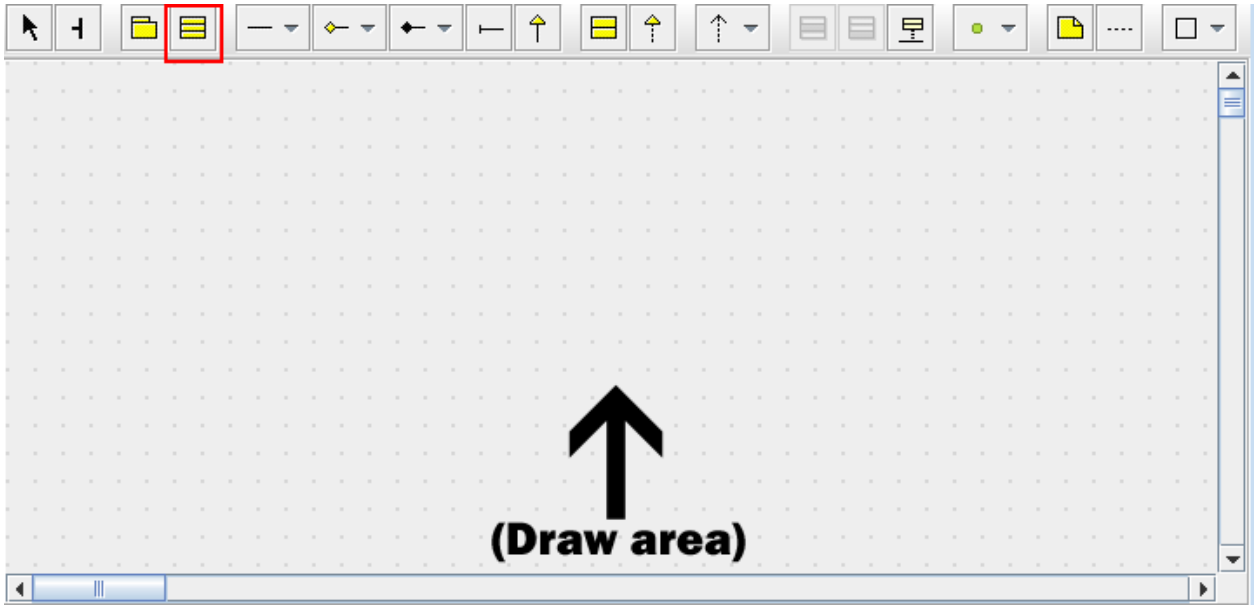
B ArgoUML Tutorial

ArgoUML Tutorial

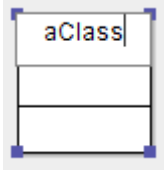
Please complete each of the steps described below in order to learn the basics of how to use ArgoUML:

- At any time, you can hover over a class or the tools in the toolbar in order to bring up a popup that tells you more about it.
- A class object in ArgoUML looks like this:

- Create a new class by selecting the "New Class" tool found in the toolbar. Once selected, click once on the draw area to make the class object:

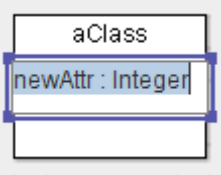


- Double-click on the top section of the class to add the name for the class. Name the class "aClass":



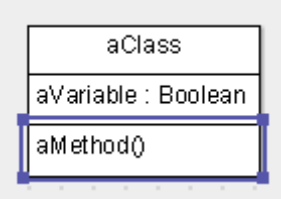
- Click on the name of the class to select it. Once selected, hover over the selected class and click the top + sign in order to add a new instance variable to the class.

- Start typing to give the instance variable a name and a type. Name the variable “newAttr” and give it type “Integer”



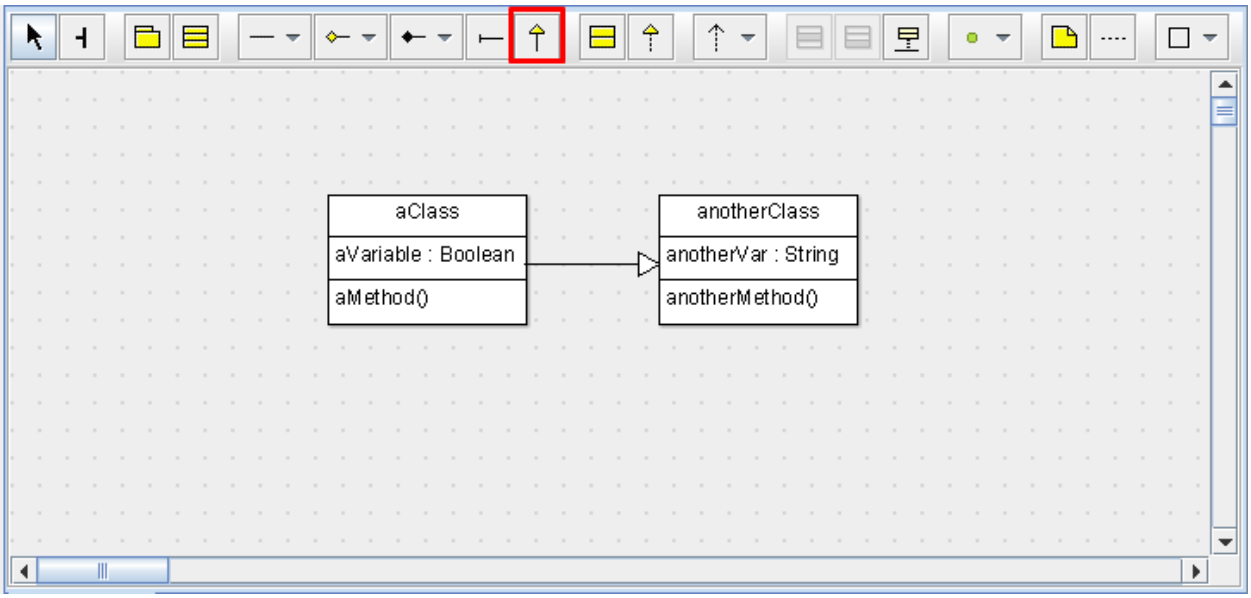
- Click on the name of the class to select it. Once selected, hover over the selected class and click the bottom + sign in order to add a new method to the class.

- Start typing to give the method a name and a type. Name the method “aMethod”. You do not need to type out the parentheses.



- Make a new class with name “anotherClass”. Give it a variable named “anotherVar” with type “String”, and a method named “anotherMethod” (see picture below).

- Select the “New Generalization” tool from the toolbar. Then click on aClass and drag to anotherClass and release the mouse (the arrow will point to where you release the mouse):



C Participant UML Diagrams

This section shows the UML diagrams I had participants draw for my experiment.

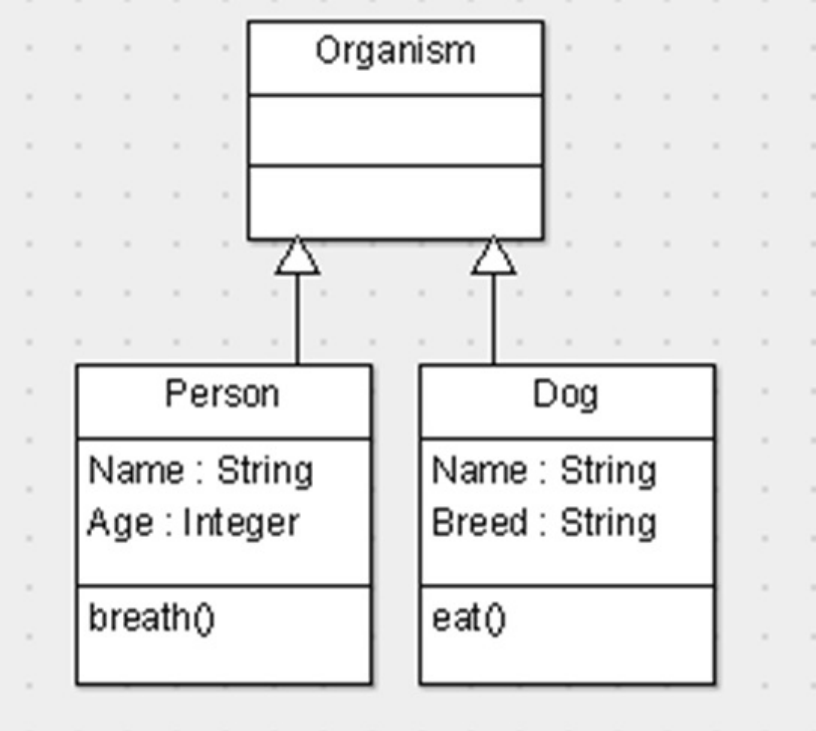


Figure 11: The first diagram participants drew.

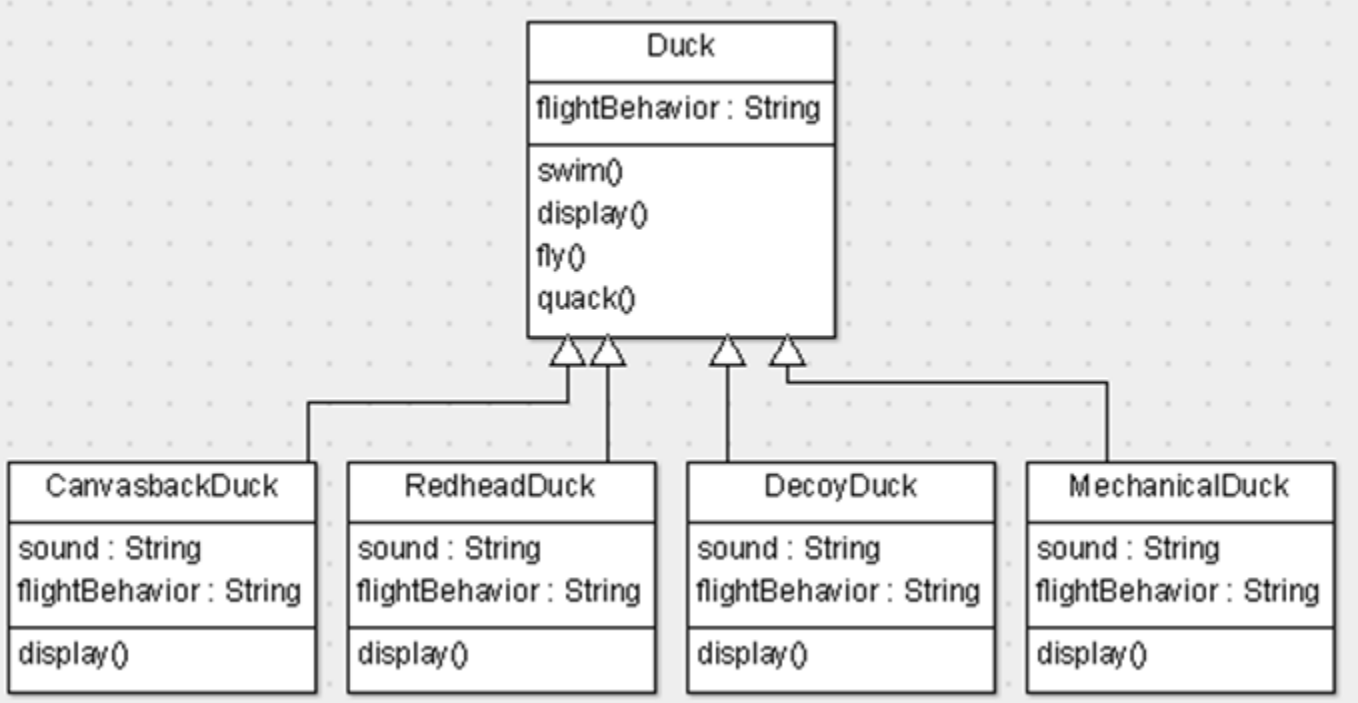


Figure 12: The second diagram participants drew.

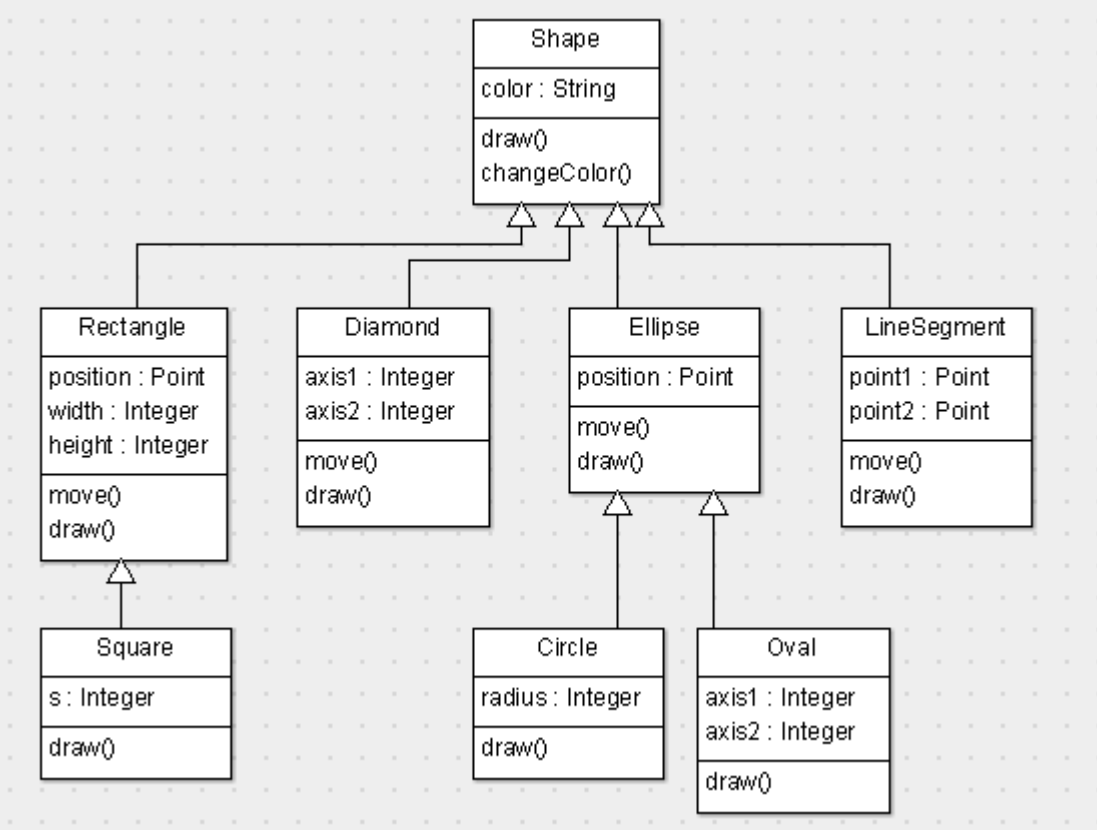


Figure 13: The third diagram participants drew.

D Participant Survey

ArgoUML Survey

Please answer the following questions.

1. **When making a new class, how responsive was clicking the mouse?**
   Mark only one oval.

	1	2	3	4	5	
Very unresponsive	◯	◯	◯	◯	◯	Very responsive

2. **When selecting a tool from the toolbar, how responsive was clicking the mouse?**
   Mark only one oval.

	1	2	3	4	5	
Very unresponsive	◯	◯	◯	◯	◯	Very responsive

3. **When typing inside a class object, how responsive were the keyboard presses?**
   Mark only one oval.

	1	2	3	4	5	
Very unrepsonsive	◯	◯	◯	◯	◯	Very responsive

4. **When hovering over something, how responsive were the tooltip popups?**
   Mark only one oval.

	1	2	3	4	5	
Very unresponsive	◯	◯	◯	◯	◯	Very responsive

5. **How accurate was clicking to make a new class?**
   Mark only one oval.

	1	2	3	4	5	
Very inaccurate	◯	◯	◯	◯	◯	Very accurate

6. **Did the bottom "properties" window ever go blank?**
   Mark only one oval.

   ◯ Yes
   ◯ No

7. **How frustrating was it to use ArgoUML?**
Mark only one oval.

	1	2	3	4	5	
Not very frustrating	◯	◯	◯	◯	◯	Very frustrating

8. **How satisfied were you with the overall usability of ArgoUML?**
Mark only one oval.

	1	2	3	4	5	
Very unsatisfied	◯	◯	◯	◯	◯	Very satisfied

ArgoUML Survey

Please answer the following questions.

9. **Would you recommend ArgoUML to someone else?**
Mark only one oval.

- ◯ Yes
- ◯ No

10. **Have you taken CSC-260 Large-Scale Software Design?**
Mark only one oval.

- ◯ Yes
- ◯ No

11. **How would you suggest ArgoUML be improved?**

Powered by
Google Forms