**Part 1: Introduction**

**Interviewer:** Yeah. Okay, I'm re­cording now. And yeah, I will start now. Can you see my screen now?

**Participant:** Yep, yep. Okay.

**Interviewer:** Now before starting our interview, I would like to introduce you about my research topic and the objective of this interview. First, let's start with the AI4Code. Recent years witnessed a significant development of artificial intelligence or AI. Moreover, the development of software engineering and open-source projects they also create a very large data code database, like come from GitHub, Stack Overflow, or Gerrit. As a result, the researchers and practitioners paid a lot of effort on applying these models that train on a very large code database to automate the code related tasks, such as bug detection, but fixing or code completion. These models showed so very promising result and usually referred as AI4Code models. Unfortunately, recent works showed that these AI4Code models are not perfect. To understand the problem, let's see the example about an AI4Code model for vulnerability detection. The model will take the input as a code function and answer the following question: if function is vulnerable or clean? And now assume that I have the function here which is predicted as clean by AI4Code model. My question is: if we change the variable from queue to buffer, and how do you think about the prediction AI4Code model for this situation?

**Participant:** Just variable change, I think it would be clean. Yeah, but the way it is mentioned it could be if it is a derived class. If it is a parent and a child relationship. So, if if the object if the buffer object, reference or pointer, if you're passing a parent object or a child object, the output could be different. Yeah, but just to name change it to a totally we clean it out.

**Interviewer:** Yes, it should be clean, but the prior works showed that is existing AI4Code model do not predict this function as clean. Instead, the model predicts the code snippet as vulnerable. As we can see that this AI4Code model are not robust. Although the two code are same but the model provides different results. So, it raises a problem robustness of AI4Code model. Thus, we need a way to effectively test robustness of AI4Code model. And, a common approach is to use the code transformations to transform from original code into the new code. And then, we use the new code to test the robustness of the AI4Code model. This approach is motivated from the nature of programming in we have a different implementation for a same programming task. For example, these three programs serve for same purpose of calculating the sum of two input number, but we have three different implementations, because we have people have different coding styles. So, the we expect that AI4Code model super for the same these variants, because the variant is actually same, so the AI4Code models perform same and but the question is, how we have good coach transformations to create the variants. And as I mentioned before, prior works used semantic preserving transformation in which we will transform original program into new program with same semantic. For example, here we only renamed from a to x and b to y. And, now we have the same program, but we have different syntax, and we can use the new program to test the model and check if the prediction the keep the same as in the original program or not. However, the semantic-preserving information is not enough. For example, imagine that if we do not turn from a to x and b to y. Instead, we change them into random strings. In this case, we have a very odd implementation, and the variant may rarely happen in real-world coding in industry. So, it will create a false alarm about the robustness of the model. Therefore, in this study, we want to consider a second property is nature. And the purpose of this study is, we want to understand how to create natural code transformation. And, towards this, we I want to employ the human annotators to classify code information into two types: artificial transformation and natural transformation and do data analysis to analyze the data. But the question is what are natural coach transformations and we need a concrete and decidable definition to give the human annotator. Therefore, we have the interview to discuss with you and other senior developers. We will find which are the natural code transformations and give us some criteria and define some relevant properties with the definition. So, it is objective of this interview. And now to start our interview, I would like to give an example. Now I have the original program here and we have a new program in which the variable name is changed into this new variable name here. Do you think the code transformation is natural?

**Participant:** You mean nature in which domains? Is like humanly possible?

**Interview:** We believe that people may be different view about naturalness. So, I think it is up to you?

**Participant:** Yeah, this looks natural.

**Part 2: Properties**

**Interviewer:** Okay, it looks natural. Why do you think it is natural? Could you briefly explain why you consider the example is natural to you?

**Participant:** Yeah. So, let's say if I'm trying to implement this functionality into a different context, right. So, I'm trying to copy some code, which has already worked, and I'm trying to write something new. And but the objective is slightly different. So, I will change variable names as per the new project. So, if I was implementing something like a queue data structure, the queue is following. But here, I'm trying to implement something handler. So, I would change this manually into the names that suit my new project. So, I would have done this naturally. So yeah.

**Interviewer:** Yeah okay. So, I want to go deeper in this case. How would you assess the variable name is natural or not? Is it depend on projects or other factors?

**Participant:** So when we implement very big projects, the naming convention is split into very long, like maybe qmp is my project name. And if it is a synchronous or synchronous way of handling the signals, I would say, a sync. And because it could be a handler to know this signal could be something to handler. So it looks very natural to me the way it is written ump\_async\_cmp\_handler. So, it is looks like it is a very structured program and architect kind of programmer who would normally write variable names like this.

**Interviewer: Okay**

**Participant:** But even if somebody would write x&y, I would say it is unnatural. It didn't make any sense. But this looks a structure program.

**Interviewer:** Okay, yeah, thank you. So, my next question is, which properties of a program that you think are relevant to naturalness about code transformations?

**Participant:** So, I need previous slides. So, the three things that we're looking for is what was that? definition, the criteria and one more? And?

**Interviewer:** properties

**Participant:** Definition, criteria and properties?

**Interviewer:** Yeah.

**Participant:** So, anything that is natural, I would say. If I'm copying a code and changing a variable, the definition would be will this name fit? Will this name be able to identify itself in a big scheme of things? Okay. So, that is one of my definition, as soon as I see, I should be able to tell from which project it is from which class it is, and which what is its functionality, like when you say qmp\_async I can tell it is from a qmp project, it is all synchronous programming and CMD handler it is doing something to handler. So, that is a definition and criteria we are criteria is yes, it was is it split into different parts, if it is just X and Y, maybe it is automatic, because our transformation might not be able to understand a holistic view of what it is coming from. So, any junk value if it is there, I might not be able to tell if it's artificial or natural, but even humans can do x and y, but if it is very well categorized, architect would normally define it like that. And properties. Properties is if the first part is the project, second part is the class and third part is functionality, then that is how out say it is well written.

**Interviewer:** So, let's I try to summarize a little bit. You mean like when you name a variable, and you need to make sure that your name is meaningful and well-structured, right?

**Participant:** Yep

**Interviewer:** And you must have some rule, like predefined rules like convention. Am I correct to capture it on?

**Participant:** yes, you're correct. And in fact, like if it has to be really written well, variable, if it's number, it should end with n or start with n. If it is a string, it should start with the str. So even that is missing here. Ideally, it should also tell the data type of that variable. Yeah. Programming Languages getting very loose with Python and other things. I don't think anybody follows that now.

**Interviewer:** Yeah, I see. I see. So, my next question is: in your experiences, how do you define how you define these rules, like you design specific convention for a project or you follow some common practice or other things?

**Participant:** Normally, forgot the names, because it's been a while. That one is camel case something. Oh, camel case of writing code there. We we know how to write a variable name. Yes. Let me check. There are quite a few techniques of how do we write variables? Ideally, when we are working on a project, so key is the naming conventions are usually defined at a project level, by an architect, developers generally do not have the right to choose, because if we give that right to developers, everybody will write in their own standards. So, when a project is defined, and when we define the UML diagrams, and when the headers are defined, and the API's are defined, and only the skeleton is defined, there's no implementation. At that is a point when it goes to developer saying, these are the connectors, these are the endpoints. And here's your header, which means it takes these variables and returns these numbers. Now you go ahead and implement. So, and at that time, they are given the naming conventions or the coding style, follow this style, and they have to follow that style as much as possible. So it's not a developer's it should not be a developers call. It shouldn't be architects or project managers should decide on that.

**Interviewer:** And now if we expand our scope, not only limit in a variable name but also about other code elements and structures such as for-loop or while-loop. Do you think which properties will affect the naturalness?

**Participant:** I think that is the developers call. I think they free, should be free to implement that, that level of details.

**Interviewer:** Okay.

**Participant:** Because it is generally not possible to decide at a logical level. The logic is generally decided by the developer how he wants to implement.

**Interviewer:** I got this point, thank you. Do you have any other properties?

**Participant:** Review the code. Yeah. So, in practice, what is what we follow is, the code should be easily readable, not to add multiple statements in one line, try to break down the code into smaller levels. So that the code for is easily understandable. And also, if it has to be rewritten. So, we don't have to write one big statement with multiple executions in that. And also, we have a peer review, when we write a code. So the next person reviews that code, and he gives a feedback, if it's easy to read or not. And this happens at every checking before you commit. And this goes through the peer review. And the peer review will provide the feedback how the nature of the code is if he finds it easy, it's okay. If it's very complicated, he is asked to change that.

**Interviewer:** I got your point. It should be code readability. But, now back to our scope. We want to assess the naturalness of code transformations. Do you think code readability are related to naturalness of code transformation? Like if the code transformation reduces the code readability of the original code.

**Participant:** Oh, you're saying okay, so you're saying if there's an original code and the code transformation transforms that code into very complicated code, yeah. It is natural or not.

**Interviewer:** Yeah.

**Participant:** So, if it if it creates a very complicated code, then I would say it is artificial code.

**Interviewer:** Yeah.

**Participant:** And it is heart rate only if a code is transformed into a similar code with relatively ease of writing outset is natural. But if that if for-statement code is converted into one line code, our code are artificial. Can you just give me one moment I'm gonna bring my headphones to this room?

**Interviewer:** okay no problem.

**Participant:** Yeah. Can you hear me?

**Part 3: Context**

**Interviewer:** Yeah. I can hear you. Now, we will move to the next problem: context. Now we need to give it the the definition for the human annotator and ask them to assess the code transformation is a natural or not or not. So which context do you think is necessary for assessing the naturalness? Like which input should we present to them for assessing the robustness. For example, should we give them only a method or the whole file, or you need to see the whole project?

**Participant:** Can you please repeat your question like do we give them what? Again please?

**Interviewer:** Like, which context or input should we present to them for assessing the naturalness?

**Participant:** To humans to the human software developer?

**Interviewer:** Yeah, human software developer

**Participant:** Okay, which which input we should give them? Could you name them like Do you have a choice or you want me to tell?

**Interviewer:** Like, for example, in this example, now, we want to assess if this change is nature or not. And we have some properties and criteria, like code readability code naming convention and so on. Which inputs, human annotators need to deliver their assessment?

**Participant:** Yep. Okay. So, if if you want to define that criteria on how we can tell if this code transformation is natural or artificial, I would say code complexity is one thing. Okay, where too much of complicated code would would mean to me as artificial transformation and any random naming convention would also mean like a name like Q is a name, but the next thing if it means x y z, I would have defined that as artificial transformation and that codings I think these two are the ones that I can see in this current thing. But this is a very simple example maybe in the complicated examples, it might change. But yeah, naming conventions and the style of writing. Okay.

**Interviewer:** Yeah. I see. So you can even assess the naturalness with the current context. But do you think you need more infomation or inputs for assesing this one is natural. For example, you only need a pair of the code, or you need additional information like the file or project contains this code method?

**Participant:** So, give me some more context, like when you're doing this artificial transformation, you're transforming a whole project or to piece by piece code,

**Interviewer:** we only can transform a piece of code.

**Participant:** So if you're only transforming a piece of code. I think we not need the project details. Okay, I would just look at the two pieces of code to understand if it is natural transformed or artificial transformed.

**Interviewer:** But do you need to make sure that the piece of code either fit with surrounding context or not? Like if you change the variable name in this, in this context, is maybe natural, but when you change in another context is not natural?

**Participant:** So let me look at it from a software engineer perspective. Let's say you took a snippet from my code, and you transformed it into something else. Right? And if I should be able to tell you whether it's artificial or transformed or natural transformed? Because I know the project, I would see. This is on this question is assuming your transformation is very smart. And because if you say to me that your transformation model has scanned the whole project and then transform this snippet, then I would assume that the naming convention that you're written qmp\_async\_cmd\_handler could be an artificial transformation because you're scanning the whole project. But if you just took one snippet, and populated the other snippet, and if this naming convention comes I would say this is natural, because I never give that input to the transformation of the project. How would it know?

**Interviewer:** Okay, I see the point. Like you mean, like you asked, if we assume that we not only change the variable name here, we also change other variable name in whole project, we should be okay. But if you only change the variable here and do not change out the variable name, so it should be not natural?

**Participant:** Yes.

**Interviewer:** I got the idea new. Okay. Yeah.

**Participant:** I see. Sorry, my answers could be very complicated for you. But you know, I come from a different background.

**Interviewer:** I know. It is welcome. Yeah, we are, the purpose of interview is to hear from you. And we try to learn from you. Yeah, I think it's enough for me for the interview. Do you have any other suggestions or comments?

**Participant:** Let's me see. No for now. I will let you know if I have more.

**Interviewer:** Thank you for your help. Now, I will stop our recording first and ask you some personal information.