**Part 1: Introduction**

**Interviewer:** To start our interview, I will introduce about a little bit about my, my, my research topic and the objective of the interview. First, I will share my screen. Can you see my screen?

**Participant:** Yes, I can. Yeah. Yeah.

**Interviewer:** So, first, let me talk about, about AI4Code models. Can you see the screen now?

**Participant:** Yes. Okay.

**Interviewer:** Yeah. First, we'll start with the AI4Code models. Recent years witnessed a significant development of artificial intelligence. Besides, the development on the software engineering and open-source projects also creates a very large code database from multiple resources, such as GitHub, Stack Overflow or Gerrit. And as a result, researchers and practitioners tried to use the AI to solve the code-related tasks such as bug detection, bug fixing or code completion and show promising results. These models are mainly referred as the AI4Code model. Despise its promise, AI4Code models are not perfect. To understand the problem, we can start with an example. Now, in the slide, we have a AI4Code model for the vulnerability detection task. The model takes as input a function, and answer for the question: is this function vulnerable or clean. Assuming that I have the first program and AI4Code model predict the function as clean. Now, if I change the variable name from queue to buffer, how do you think about the prediction of AI4Code model for this new program?

**Participant:** Also, it's the same right? This function is that simple. Yeah, if nothing is in, yes.

**Interviewer:** Yes, it should be clean, but the prior works showed that is existing AI4Code model do not predict this function as clean. Instead, they predict that it is vulnerable. These observations showed that AI4Code model are not robust against the semantic preserving transformations. And therefore, we need an effective way to assess the robustness of our focus model, and its purpose of my research. And one effective way to do this purpose is using code transformations, in which we will transform from original code into the new code and use a new code to assess the AI4Code model. The intuition behind the approach is nature of programming in which we have different implementations for same farming tasks. For example, we have three programs in the slide and these three programs serve for a same purpose of calculate sum between two input number. But due to different coding styles, we have different implementations, and we can even have more. And now the by using transformation to transform from the first one into the other one, we want to we can assess the robustness of the model. And we expect that AI4Code model to perform same as these programs are actually the same. Towards this, we need a good code transformation. Prior works mainly uses semantic preserving transformation, as I mentioned before, in which they use which information to convert from original program into new program, such that the new program is semantically equivalent to the original program. For example, here we change variable name from a to x and b to y. And now we have a program with different syntax. But they have the same semantic, so we can expect that the two programs are the same. But in my opinion, semantic preserving.

**Participant:** I think your voice was one. Oh, sorry. I think your voice was cut off like occasionally.

**Interviewer:** Okay. Yeah, can you hear me now?

**Participant:** But I can hear you. Your voice now is perfect, but sometimes just because hold for like one or two seconds, and then it comes back.

**Interviewer:** Okay. Yeah. So let me know when I have any problem.

**Participant:** Yeah. Okay.

**Interviewer:** Now we will continue. Semantic-preserving transformation is not enough. Let's see the example. Now, we do not change variable name from a to x and b to y. Instead, we changed from a and b to random strings. And now we have a very odd implementation and rarely happen in a real-world coding and now is create the false alarms the about a robustness of AI4Code model. So, in my study, I want to focus on second property is naturalness. But we found that there is no prior work on: "What is naturalness of code transformations". Therefore, the objective of my research is to investigate a concrete and decidable definition for the property. Towards this goal, we conduct the interview to talk with you and as other developers to understand as your opinion about the naturalness as a code transformations. Yeah. So, it's the purpose of interview. Do you have any questions?

**Participant:** Question? Sorry, how does it relate to vulnerability detection?

**Interviewer:** Can you repeat again, because if your voice also have some problem, yeah.

**Participant:** How does this transformation help with vulnerability discovery?

**Interviewer:** It does not help with vulnerability discovery. Instead, it will be used to assess the robustness of AI4Code models. For example, here. Now, if we have two functions, the same semantics, but now one function is predicted as clean, and one function predict as vulnerable. So now, we cannot trust the AI4Code model, because they are not robust, and they create a false positive and also false negative. So, we want to make sure that the AI4Code model is robust against these transformations.

**Participant:** I say, I see. So, the focus is on this AI4Code models.

**Interviewer:** Yeah. It focuses on AI4Code. Vulnerability Detection act as a simple example, yeah.

**Participant:** Cool. Cool. Thanks.

**Interviewer:** And now, so do you have any other question?

**Participant:** No, that's all. Okay.

**Interviewer:** So now we presented a brief introduction. Next, we'll focus on a naturalness of the code transformation. To begin, I want to ask you a question. Now, I have the original program here and we have a new program in which the variable name queue is changed into this new variable name here. So, do you think this change is natural?

**Participant:** I'll say depends on the context of the second function. If it has a, if you use the variable qmp somewhere else, that can be natural. But if that's very new and not related to anything, then that's not.

**Interviewer:** So, if you have the context, how do you identify if the variable name is natural or not?

**Participant:** Also, if a substring of the variable name has occurred somewhere, for example, if you define the variable called qmp, and or if I have a class named that, then this can be natural.

**Part 2: Properties**

**Interviewer:** So, it depends on the context. Yeah. Yeah. I got it. Now, which properties of a program that you think are relevant to naturalness about code transformations?

**Participant:** I'll say it relates to the code readability. If the name is very easy to be interpreted by human like another programmer, then it's very natural. If the name is weird, and people can't understand it. That's not natural at all. One way to measure this, as we just said, is to look for the context of this program. If that mentions the name, it is if a if Name It was never mentioned elsewhere. That is probably not natural.

**Interviewer:** I see, I got this point. And now if we expand our scope, not only limit in a variable name but also about other code elements and structures like for-loop or while-loop, how do you think which properties will affect the naturalness?

**Participant:** Your voice was cut off a little bit. So, you're talking about expanding this from variable names to other code elements and structures?

**Interviewer:** Yeah, it is Correct?

**Participant:** I mean, I'd like to explain the structures name and something

**Interviewer:** like about the while-loop or the for-loop. And, also about other statements? Like, yeah. It's not about only variable name. I mean, like about everything is in the source code of a program.

**Participant:** I see. I see. I see a point here. Oh, yeah, that's a very good question. I guess this is very hard to define. Normally, people can only tell when they actually look at the code. So I'll try to answer this one by one. So all for structure. Also, for structure, it depends on all the relation, or between the structure name and its content. Or if the content of the structure is not related to the name, then it's unnatural to me. And it also depends on the complexity of the structure, if there's normally programmers try to avoid having a very complex structure, that's at least what I think we should. If a structure is over complicated, then it's also unnatural to me, is the function for function. I think that depends on the statements inside the function, like whether these statements have occurred. Together in other places, or whether the, like, intentional, those statements are like semantically or, or, or very close to each other in terms of their meaning, or in terms of their intentions. Or, for example, if you have a function that says do A, B, and C, right? If those types normally occur together in most places, for example, if I want to save a file, then I probably want to define the name and open the file and write its contents, right? These three things together seems natural to me. But if you have some other steps inside this function, for example, or like, oh, I don't know, like. How do I have a good example. All right, for example, defining a new variable does not use inside this function, then that's very unnatural to me.

**Interviewer:** I see. I see. Yeah. It got your point. Before, you mentioned about complexity of a structure, could you give me an example about this?

**Participant:** Right, also referring to the number of lines in the structure and the depth of this structure.

**Interviewer:** Okay, number of lines

**Participant:** If you have a lot of attributes in this structure, you probably want to consider splitting it into several structures instead of grouping all of them into one right.

**Interviewer:** Yeah. I see.

**Participant:** And if you have a look for the second point, if you have a very deep of structure, like you have a structure of structures of structures, then you probably want to consider it by itself. Like a data structure, for example, using plus, instead of doing that, I guess the essential point is still human readability. Right? If it is very easy to be understood by human, then that's natural. And it's also good for code interpretation. If it is very complex and human. Not normally. It's very difficult for humans to understand is intention. That's bad, and it's unnatural.

**Part 3: Context**

**Interviewer:** I got this point, thank you. And my next question is, do you think like, code convention and naming convention is related to naturalness of the code transformation?

**Participant:** have so what's the question?

**Interviewer:** Do you think coding convention is related to naturalness of the code transformation?

**Participant:** Oh, it does. But I think for most of the time you given a programming language people will assume a given naming convention, right sometimes use camel case, sometimes you use those underscores what you have here. But the most important part here is to make sure the code is consistent with the rest of the code. For example, if if you have a file with like two, two spaces as intention as an indentation, and then you have another file with four more spaces indentation, they will see a difference in their New York considered as an actual probate the other file is from some other codes or is written by machine. So, the consistent here, consistency here is a key.

**Interviewer:** Ah, I see. Yeah. Yeah, thank you. So, my next question is, before you said, when you went see the sample you said about the context. So which context do you think is necessary for assessing the naturalness like, you need to see the method, or you need to see the whole file, or you need to see the whole project?

Participant

Good one. So it depends on how much effort we want to put into this. Seeing the context of a function is certainly the minimum, right, we want to see the variables defined before they sign and the function calls after this. And the second level is to see the like the symbols used in the current file, because logically speaking, they should be very close to what's worth doing, right. It's everything we do within the file should be, by and closely together. And the next thing is to see the files imported by the current file, for example, do they use functions or variables from those files? Yeah, the largest scope is to see the approach the whole project, you may want to look into other parts of project that is important and why this one, but I think that's too complex.

Interviewer: Yeah, I see, I think, yeah, but for you, what is the minimum scope? Like we need to find acceptable context for the our annotation process, but keep the quality of the assessment is acceptable. So what is the minimum context that you want to have when assessing a naturalness.

Participant

everything inside this function including the parameters

Interviewer

Everything inside a function including parameter?

Participant

Yeah,

Interviewer

Yeah, I think it is enough for the interview. Thank you very much for your help. Now I first I want to stop my recording first and ask you some your personal details.