**Part 1: Introduction**

**Interviewer:** Okay, I started recording.

**Participant:** Right.

**Interviewer:** Now we will start. Hi, before starting our interview, I would like to briefly introduce about my research topic and the objective of our interview. First, let's talk about the AI4Code models. Recent years witnessed the significant development of artificial intelligence or AI. Besides, the development of software NZ provides entering and open support, I also create a very large code database from multiple resources like GitHub, StackOverflow or Gerrit. And as a result, researchers and practitioners propose AI4Code models that trained on very large data to automate the code-related tasks like bug detection, bug fixing or code completion. These model so very promising results on such tasks. Despite the promise, AI4Code models are not perfect. To understand the problem, let's look into the example. In this example, we have a model that take the input as a code function and answer for the question is: is this function vulnerable or clean? And now assume that I have the function here and it is predicted as clean by the model. And now we change variable name from queue to buffer. How do you think about the prediction of AI4Code model?

Participant: If the previous one was clean, and you just renamed the variable, it should also be clean the result?

**Interviewer:** Yeah, correct. But different from our expectation AI4Code model provides a different result that is vulnerable, because buffer maybe is more related to security than queue. So AI4Code model is bias, and they mispredict that this function is vulnerable. And we can see that, from our original data, we add a very simple code transformation, we would have a different result, the phenomenon is mentioned as the robustness problem of AI4Code model. Prior works showed that AI4Code model are not robust against semantic preserving transformation. Therefore, we need an effective way to test the robustness of AI4Code model. And a common way is to leverage a code transformation to transform from our original code into the new code and use the new code to test AI4Code model. This approach is motivated from the nature of programming in which we have different implementations for a same performing task. For example, three programs in the slide serve for a same purpose of calculating sum of two input number, but we have a different variant due to the different coding styles of the developers. But these programs are the same. So, our expectation is that AI4Code model performs same in these variants because they are actually the same. If know, we could say AI4Code model is not robust. The question here is which is a good code transformation? As I mentioned before, prior works mainly use semantic preservation transformation, the type of transformation transform from original code into the new code which is equivalent to the original one. For example, here, we transform from a to x and b to y. And now we have two programs that are semantically equivalent program but different regarding syntax. However, semantic preserving transformation is enough is not. For example, now we do not change from a to x and b to y. Instead, we change them into random strings. And now 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. So, my, the objective of my research is to investigate what is the natural code transformation. Towards this, we plan to first employs human annotators to classify the code transformation into artificial and natural transformations and use data analysis to learn from human data. However, we found that there is a lack of concrete and decidable definition for the naturalness of code transformation. Therefore, we conduct this interview with you and other senior developers to learn from your opinion about natural code transformation.

**Participant:** What do you mean about the naturalness?

**Interviewer:** It is kind of interesting question. Unfortunately, I think I should not mention my own definition about the naturalness to avoid potential bias. The actual objective of this interview is to understand you opinion about naturalness of code transformations. So, I think it is up to you.

**Participant:** I see.

**Interviewer:** And now, let's start with the example in the slide. 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:** Rename one variable from just a few to more meaningful names. So, I think it's, it's kind of natural. I think it is.

**Part 2: Properties**

**Interviewer:** Okay. Yeah. Why do you think is nature? Could you briefly explain how you assess the naturalness of these examples?

**Participant:** In this case, I would say it's more natural for the developer, because instead of using like, in the coding convention, normally, when we, when we write a program, we try to use the variable names that are meaningful for the developers, like for the ease of maintenance. So, let's say if there are a lot of people leave the project, and the new developer comes, and then they need to fix the fix the code or whatever, if they see the code on the right, I will say is like, it's natural for them to understand what the code is supposed to work. And like, if they need to fix let's say, some defects in this code, it should be easier for them to you know, just follow the variable names and understand what the code is supposed to do. Yeah, so yeah, that's why I think it's more natural.

**Interviewer:** So, let's me summarize a little bit. So, you mean, it is related to the meaning of the variable name?

**Participant:** Yeah. In this case. Yes.

**Interviewer:** So now, let's broaden our scope, that is not only the variable name, which properties of a program that you think are relevant to naturalness about code transformations

**Participant:** Oh, it depends on depends on many factors. Are we talking specifically about this, this these two code snippets or asking me in general?

**Interviewer:** yeah, we this example is only for getting familiar with the problem. And we talk in general.

**Participant:** Yeah. Okay. So in the broad context, that meeting could be the nature or the code change. Maybe in algorithm, like, if you are doing something simple, the code should not be more complex, like the, the nature of change the nature of transformation of the code and make it more complex. Let's say, if you're using for loop for something, the transform version of the code should not, let's say use the recursion, for example, like it should not make the code the result more complex than the than the original version of the code. So that's also a casual thing.

**Interviewer:** So you mean is code complexity? I want to clarify that, you know, how, you mentioned about the code complexity, or you mentioned about code readability?

**Participant:** In this case, I say, complexity not readability. Yeah. So yeah, so these are different. So, for the for the first case, for the renaming of variable that could be readability. That could be one of the readability. But what I just talked about the case of the for loop to recursion, I think, I mean, complexity.

**Interviewer:** But complexity here is mean like the complexity of the code, not complexity of the algorithm, right?

**Participant:** Well, what do you mean, what is the difference between complexity of the code and algorithm?

**Interviewer:** I mean, like, complexity in algorithm like O(n).

**Participant:** Oh, no, no, no. Yeah. Yeah. The complexity, okay. Yeah, Right. Right. So not the time not the time complexity. Yeah, the complexity of the code.

**Interviewer:** Okay. Yeah, I don't want to make it more clear. Thank you. Um, so do you have any other properties that you think are relevant to the problem?

**Participant:** Yeah, it could also have something to do with the complexity of the code like, you can transform. I'm not sure if this can be done in the AI4Code code, like, let's say you are calling some third party like, very, when you when you transform the code, it's like, replace the third-party library with something else. Like, using some other dependencies that might be safer, that might be less vulnerable. So, I'm not sure if we could consider that natural like, like comparing, okay, one thing that I can think of is like, instead of using the transformer, you have like, another developer, and you ask them to transform the code.

**Interviewer:** Yeah. Yeah.

**Participant:** So if they, if they change the code in a way that it's like, comparable to the original data, I would say that's the most natural thing because it's from the human. So the human developer, actually, so it's supposed to be natural. So what they would not do might be like, trying to change all the dependencies to the other libraries. That might not be the thing that human developers do. Because why would you do that?

**Interviewer:** Yeah. Like you implemented AI models on TensorFlow. And now you change the implementations of AI models to Pytorch?

**Participant:** Yeah, right. Yeah. Yeah. Something like that.

**Interviewer:** Okay. Do you have any other properties?

**Participant:** Right. What about this one? like the external storage? Or the how do you call this like, the resources of the program? Like how how this also has something to do with the code complexity, I think, like, but it's, it's kind of not exactly the same, but you can implement a program in a way that it use it use the memories to store some data, or you can implement it in a way that it used the file storage to like to keep the data. So the output of the program might be the same but Use the different set of resources. You know what I mean? I'm not sure how to define this.

**Interview:** Okay, okay. Yeah, maybe I could think more about it later. Yeah, yeah. But

**Participant:** it's like the output of the program is actually the same. Like, if you test it, if you run it with, let's say, you need tests are anything, it produced the same result. But during the runtime, it might use different set of resources. Like one of them might use more memory. The other might use more file storage.

**Interviewer:** Which like about, like everything to efficiency of the code.

**Participant:** Not exactly the efficiency like I would say, it's more like the resource.

**Interviewer:** Memory Okay. Okay. Yeah. Like, you have the original program only use 5Gb, but the new one require 10 Gb. Something like that, right?

**Participant:** Yeah, yeah. Right.

**Interviewer:** Yep. Yeah. But yeah, but it's required a runtime information, right?

**Participant:** Yeah.

**Interviewer:** Yeah. I see. I see. Okay. I got your point. So do you have any other properties? How about code convention? Do you think coding convention will affect the naturalness of code transformations?

**Participant:** Very much. Yes. So that's, that's why we, I think that's why we first talked about the renaming of the variable. Because that's like the most the most part of the convention in the project. So naming,

**Interviewer:** but I want to broaden the scope. It is not limited to the naming convention, I want to ask about other conventions.

**Participant:** other things. The other thing about a convention might be the design patterns. Like, yeah, how would you separate the classes? How would you organize the packages in the in the project? Like, are you having just one set of the common libraries that every every function will call for you having like, different utility class for the different package? So yeah, that could be the other things.

**Interviewer:** Okay.

**Participant**

Yeah, the organization the code.

**Interviewer:** Yeah. So, this is about something like, common thing. But do you think the project-specific convention is relevant to the naturalness, like if you have a code, and you change into another code, but the break the current convention of the project? Do you think affect the naturalness?

**Participant:** Oh, yeah, yeah. Well, when you when you talk about that, yeah, actually, that makes me think, like, when we define naturalness, is it? Is it project specific? Or is it universal? Like, yeah, that could be something that could be some naturalness that we can see as like the nature of things, but all the developers, but there should also be some nature on is that like, project dependent.

**Interviewer:** My context, currently, we will transform only one function inside a project. And we want to see if the transformation is natural, respect to the surrounding context, or not.

**Participant:** Right, so So basically, it's project specific,

**Interviewer:** But, for example, I would like to confirm that, like, if you change in only one location, it may be unnatural but if you change the whole project such as from Camel case to Snake case, it should be natural?

**Participant:** So the scope of this is just in one function, or you have multiple functions, and you put them into the transformer together, and you expect them to behave similarly.

**Interviewer:** It is only one function.

**Participant:** But if you only have one function and you want to test it, how would you change? Like, how many conventions are there that you can change? And you can compare? There might be just a few.

**Interviewer:** Yeah, we do not know now. We can't. we have many projects. So yeah.

**Participant:** But with just within one within one function, I think it's quite, it's not quite easy to imagine how much you can tweak the naturalness like, yeah, I still think that it depends on like, not only just in one function score, but maybe in the larger one. If you if you want to compare the naturalness of something. So you, you may need to think about the neighbours of it like, the other functions in the in the same package in the same modules? Are they? Are they being changed in the same way? Something like that. Otherwise, otherwise, if you just look at one function the scope of relation is might be a little bit limited, I think, like, you can you can think about replacing the names of the variables, you can think about changing the conditions and expecting the same result. But would it be natural? Maybe up to some point, like two, yeah, two or three.

**Part 3: Context**

**Interviewer:** Yeah, I got your point. Okay, now we move into the next question. Like, how do you assess a given code satisfy the aforementioned properties or not? Like you only need to rely on only the current method? Or do you need to rely on the whole project, or you need to rely on the files associating to transformed methods? In other way, which scoped you need to assess a code transformation is natural or not? Yeah.

**Participant:** Yeah, actually, this is this was the point that we just talked about. I don't think the functions could be now I would want to see the larger scope, like, the file would be good. But if you can compare with the other thing, is a project. It would be better.

**Interviewer:** Yeah, I see. Okay, so you think it’s acceptable? If we use the associating files, but it's better if we can do the whole project?

**Participant:** Yeah.

**Interviewer:** Now, assuming that you use the whole project. But a project may be evolved, right? It may have different versions. So how about just you only one version? Is it okay? Or you wish because it is like very hard for people to read all versions of a project. So, we could we assume that one version is good enough for this assessment?

**Participant:** So now we are we are talking about taking one function from a project in this version, and transform this function to the new version and somehow look at this version and see if it's natural, by using the whole project can take one?

**Interviewer:** Hmm. My question actually is once you assess a project, you need to read the whole project? Or you can only need to take a look on the project in the first time to capture everything of project like any convention, any style of project. And then, you can assess the different code snippets in this project later.

**Participant:** I think just one snapshot, just one version is fine. It should be fine. Because my reason is like if you have multiple versions of your projects, and like what you said the project can evolve, it might not be possible to come up with just one version of naturalness in order to test that transform function. So, sticking to just one snapshot of the project might be the better idea. It is my thought. Yeah.

**Additional Questions (for Part 2)**

**Interviewer:** Yeah. Okay. Yeah, I think it's okay. So, do you have any like, any other thoughts like about property or something? Ah, how about the code readability. Do you think it is important to the naturalness?

**Participant:** Well. That should be the fundamental, the fundamental property of it. I think, like, if you transform it and then there's no reason to transform the code like is unnatural anyways, right? Like, if you put the code into the transformer and the result is unreadable, like, let's say you have escaped the whole the whole file, and it becomes something that human cannot read, then it's unnatural. Anyways,

**Interviewer:** but sometimes it just reduce the readability and do not make it unreadable at all. Do you think it affect the naturalness?

**Participant:** Can you repeat the sentence? And again, I lost the topic.

**Interviewer:** Like, like. it does not make the code unreasonable at all instead it only reduce the readability Do you think it affect the naturalness?

**Participant:** Reducing the readability? Yes, I think it affects the naturalness. Oh, yeah. My, my scenario would be like, if you Okay. Let's say we don't have the transformer; we don't have the cord transformation. Again, I use this scenario before. So, if we ask the other developer in the project to to refactor the code, let's say so. So, it should be similar to be to what we asked the transformer to transform the code, right. So, we asked the developer, the outer developer to refactor that function, we do expect it to be less readable.

**Interviewer:** Okay.

**Participant:** Yeah, the answer might be no. Right. So you would expect the same level of readability?

**Interviewer:** Is it depend on how much it reduce? For example, it reduce a little bit, maybe okay, but it reduce too much is not okay, something like that?

**Participant:** I don't think so. Well, if you if you are other people to refactor the code, I don't think it will be less readable even even a little bit. I don't know. Okay. Yeah. Yeah. Well, there might be some case that the code is harder to read, but Oh, yeah. There might be some trade off. There might be some trade off. Like if the code is a little bit less readable, but it performs better.

**Interviewer:** Okay, I see.

**Participant:** Yeah, that might be that that might be acceptable in a way.

**Interviewer:** I think it's done for our interview. Last part is only I want to ask your personal information. So, yeah, I can stop the recording first because I will ask about your personal information. Thank you very much for attending our interview.