Interviewer1

Okay, record instructors. So I'll start with the demographic question. The first question is, how old are you?

Interviewee

I'm 42 years old.

Interviewer1

Okay, and how do you identify your gender? Gender identity?

Interviewee

Female?

Interviewer1

Okay, and what is your highest level of formal education?

Interviewee

Master's in Computer Networking? Okay.

Interviewer1

And how many years of experience do you have in software engineering field?

Interviewee

20 years?

Interviewer1

And how many years of experience do you have in leading teams?

Interviewee

I am not a people manager per se, but I do. I have been leading for about like,five years, I would say,

Interviewer2

yeah, it's not, but people manage it more like even engineering manager engineering teams. If you've been leading teams, that's all that be?

Interviewee

Yeah. I mean, yeah, I'm not like by designation. I'm an individual contributor, but I have like, junior engineers who work with me on projects, and I help them and mentor them and those kinds of things. So that's what I mean.

Interviewer2

And how long would that time frame? So

Interviewee

I was with a startup. So I wouldn't like yeah, it's been like about, I think, five to six years or like little bit more. I've been like a tech lead for since about 2016. So perfect. Thank you.

Interviewer1

So these are our demographic question. I will start the interview question. So the first question is, when we use the word bag localization tool, what do you think about the functionality of this?

Interviewee

When you say bug localization, I assume you mean, more like Val grind? Anything that would help me like, has helped me figure out the issue, right? So I've used Val grind, and there is there. Mostly it's been like Val grind is what I am not able to recollect there is more there was one more similar tool. More like I've used tools for memory analysis is what I have used bug localization tools. Other than that, any functionality bugs, we typically do not use any like tools outside of our own analysis.

Interviewer1

So in this study, we defined bug localization tool, a tool that can identify potentially buggy files, given the bug report or stack trace, or both. Suppose a repository contains, say 700 source code files built on the bug report that will do a shortlist a few files potentially responsible for data that goes output is a ranked list of files sorted based on their probability of being responsible for a particular bug. And then the developers can start further investigation or their fix procedures from them. So that is our definition of bug localization.

Interviewee

Okay, so in that case, I would add, like we have these coverage reports, like static analysis tools, those help as well. Yeah, I think, yeah. And then we have I mean, this more like a CI CD, where we use so we have our own tests that we use, so that that helps uncover like on a daily basis with every code commits, whatever bugs come out, so.

Interviewer2

So in one of these things, so like, in the research area, bug localization would mean more so that, you know, when a bug report comes into an issue management system like Jira, or Bugzilla, from the text of that report, whether that has just natural language text or stack trace, or anything like that, can we then point to a file or a method or something like that, usually a file where we think the fix for that bug is going to be

Interviewee

okay. I mean, yeah, go ahead. Yeah, no, no, that's what Okay, so, yeah, stack trace would definitely we, I mean, typically, if it is, if it's like an assertion that we hit, it is StackTrace would directly point to the file that you're looking at. I mean, that is where anyone would start and then you analyze whatever is the contents and stuff, maybe. I mean, typically, that's that's where we start from for, if it is a, if it is on the other hand, if it's a memory corruption, then wherever it is localized is not going to help us because it could have happened like way before we actually hit the problem, right? So these are the two. I mean, in those cases like maybe if if software is already enabled with some sanitizers, and those kinds of things might actually add value to localizing the bug. Because those actually point when the issue happens, they could ask hit assertions. So that would help in those cases.

Interviewer2

Yeah. And sometimes we don't have stack traces. So from the English language text, we kind of say, you know, by doing some kind of analysis, assume this is a black box analysis, okay. We say where a developer should fix the bug, not where the bug happened, but where the bug should be fixed.

Interviewee

Okay, all right. So

Interviewer2

the researchers have been trying to build this black box tool, which then takes you to the location where you start debugging or start fixing the bug. Okay.

Interviewee

Yeah, that I mean, yeah, if somebody can, yeah, pick it out that that that would for certainly, like reduce the lot of effort. Because I mean, even even for simple assertion failure, it takes a developer because assists test or any like testing, when they bring the report to us, they, although they understand the functionality of the product, I have been working with mostly product companies, right. So I although they understand the product or functionality from networking standpoint, everything they they do not understand what the code does. So even for a simple assertion failure, when we hit like a case, we have to be looking at things. So this I mean, yeah, this would be this would mean, a lot of developers time would be saved.

Interviewer1

Okay, so our next question is, what do you think will be the benefits of using such a bug localization? Like you've said that, then?

Interviewee

Yeah, yeah. I mean, when we have like a release cycle going on, we typically like so there is, the way I have seen things happening is there will be like a first pass where you have like, mostly functionality, bugs that are that gets, you know, and those times we have like a lot of them coming in, because although although we have, like developers writing unit tests, and whatever the integration tests, it's, it's not typically the environments, it's not the same as when, when a system test would do it in a bigger scale. So those sometimes at the beginning of the cycle, we get, like, really simple bugs, which are like, which can be like if there is a tool like this, the at reduced the developer cycle to fix those, and it would definitely like, you know, help the cycle move much faster than normally, developer has to be involved in.

Interviewer1

And what do you think will be the harm of using such a tool?

Interviewer2

Do you think there's going to be any disadvantages of using this tool?

Interviewee

Um, I mean, maybe he like, less number of people, less number of engineers. That's, that's what I see. But yeah, I otherwise I don't see. Disadvantage person.

Interviewer1

So previously, you have you are mentioning about some tool, some kind of static analysis tool or something. So do you use any kind of automated tool for debugging purpose?

Interviewee

Or? Yeah, I mean, typically, we have we have discovery reports generated and like, we also integrate the I've had products integrate LLVM into, like build those kinds of issues like that before even we check in any code, those exposes some LLVM exposes all those static analysis issues. And also, like there are Coverity reports that get generated. And to your point before bug localization, I mean, yeah, I know, I don't know how this blackbox magic will work. But then in some cases, what I found is even these tools that we see, not always they are suitable for all the environment. So I think that that could be one of the challenges too.

Interviewer2

Would you also say that you can mention this before, and we all kind of included here, like Valgrind is a tool that you use for debugging as well?

Interviewee

Yeah. Yeah. There's one more of that. Yeah. Similar to Valgrind. I'm not getting any Yeah.

Interviewer2

But well, then gdb things like that, that you would use. But you but you are driving that tool you are telling where correct the tool should be looking at and things.

Interviewee

Yeah, that's true. Yeah. Also use a perf for it. any, you know, loops are like any stalls, purpose, another very useful tools, scalability.

Interviewer2

And typically these bug localization tools will not replace any tools like gdb or Valgrind or Perth. Because they will say where you should start looking at, right? It's not going to say how to fix

Interviewee

right, right. I mean, yeah, even these tools that we are not going to like, they are just giving us clues, right. We are just looking at clues which human eye cannot like, immediately, spot. Yeah, so yeah, exactly.

Interviewer1

Yeah, so do you have do like any special functionality of that tools that you are mentioning before?

Interviewee

like I said, it varies from problem to problem, right? Like GDB, we always use an anywhere where there's a code dump, of course, GDB is like lifesaver, right? So and you have Valgrind, it is very useful in in terms of, but Valgrind is not always useful. Like if the problem is reproducible. Yes. Valgrind is useful. But if it is not, then we are left to our own analysis. And you have perf Of course, again, those those are like Valgrind and perf. I would say they are useful if I can run it in a system and then analyze it. But yeah, if it's if it's something like an output that's given to me without much of information, even for that matter, gdb without a core, I cannot do much so. Yeah.

Interviewer1

So do you dislike any of the features? Or do you feel lack of any features?

Interviewee

Um, I haven't like really thought of any specific feature that maybe,

I mean, something, I'm just like, making it up. So right now, we do a lot of this packet analysis. So the way we look at it is, whenever we hit an issue, I'm, I want to dump the packet that actually hit the issue. And then I take it to this online hex packet decoder, which actually gives me an if we have something like a scape, scape integration, that can actually like display the brackets for me with gdb. Just throwing it out there. But yeah, okay.

Interviewer2

Okay, so.

Interviewer1

So our next question is like, according to your perception, what should be the minimum required functionality of a bug localization tool, like we said in our definition, the bug localization tools return a set of potential files? So what do you think about the maximum size of that set.

Interviewee

Can you repeat? I'm not I'm not exactly following

Interviewer2

that. The the, when, when the issue comes in, we will return a set of files. Okay. Okay. Which are potentially where the fix should happen? How big should this set? How big can the set be? Now? The bigger the set, the more accurate we can claim? We're right, right. Like, what do you think is an ideal size for this set? I mean,

Interviewee

I think you can target them, like zip them, as long as it is in some few kilobytes. I think, like, hundreds of kilobytes is also like, it is. It is.

Interviewer2

But I think what we mean is that let's say there's 700 files in a project. And for a particular issue, we may say that, you know, should we say that, let's say File A is where the fix is going to happen? Okay. Should we say only one file in our answer? Can we give five files in our answer? Can we give 10 files in our answer?

Interviewee

I mean, so from what I see if it is going to be, it's going to pinpoint if it's going to help me like point me in the right direction, then the more pointed it is, it's going to be more helpful. Rather than throwing, like multiple files at me, I'm going to be confused and doing similar work as normally I would do. So. Yeah. So probably one file is the best. Yeah, somewhere where I can start, right. Okay, at one or two, like not like five or 10, where I have to again, do the work. Yeah.

Interviewer1

And what should be the latency of such a tool? Like should the tool return the files, potentially buggy files in maybe minutes or or maybe our

Interviewee

minutes would be ideal. Like, I mean, I'm thinking if you integrate it with JIRA when someone files a bug, then yeah, typically minutes is like, okay, powers is like, I'm looking at much longer time, I think minutes is what? Yeah. Even even if the details can be like, a little bit reduced and you want something immediate, then yeah, then it's better. Okay.

Interviewer1

So our next question is Do you think other types of output beside sets of file would be helpful, for example previous bugs similar to the current bug report, or the best person to fix a particular bug,.

Interviewee

I think, author of the file or like, at least author of the you can even do a blame on that particular localized code where you're pointing to typically, that's, I mean, that could be useful that can be used or not used is up to the developer, but then you can that is something also we tend to do, because if I'm looking at something I'm not aware of the code path, I would typically do a blame and look at who committed that and go and talk to that person. So they are that that is that would be useful to the human

Interviewer2

because other information, like if we find a similar bug, or some other place by someone else,

Interviewee

oh, that that would definitely be useful. Yes. Yes. And then even if it may not be in the same project, it can be like in a different project as well, sometimes.

Interviewer2

And I don't know, Partha, did you miss the accuracy? Question? Are you coming?

Interviewer1

On? I'm coming. So what do you think about the minimum accuracy of such a tool? Like, what should be the expected accuracy? Do you think it should be 100%? Or close to 80% is also good for us?

Interviewee

I mean, I think it's, it's as good as we can get, right. I mean, yeah, definitely, like something better than nothing. So yeah, I think 80% I would say is like, really good. I mean, based on whatever tools we use today. I mean, the like I said, these are like, we are hope, hoping that they will give us something right? They are mostly I would say 50 to 60% is what they give us the rest of it. We do the you know, we do the code walkthrough and stuff. So yeah, 80% is a lot. Okay.

Interviewer1

So do you think an explanation would help the developers like this is what we are defining is kind of a deep learning tool. So it can get some kind of explanation why that particular file is here why this particular tool thinks that this file is responsible for that bug report? So what do you think about that kind of explanation

Interviewee

I would Yeah, I would definitely recommend adding that as well. Because yeah, I if you give me like two files, and then yeah. For like, how you arrived at it would definitely help.

Interviewer2

Yeah, because some of these tools are doing like word matches and things like that. Right? We could say that look, because of this method name that we found out from this issue. This is why we think it is right. While this is a city. Yeah,

Interviewer1

exactly. Yeah, yeah, yeah, definitely. Okay. So who do you think that this tool will be useful for? Is that good for newcomers? Or it would be useful for the experienced people?

Interviewee

I mean, like I said, both in the sense like, for a newcomer, definitely, it gives a starting point, right? Typically, that's where they are, like, even not be a newcomer like two, three year like, sometimes you, they just they don't know where to start. That's where we help them. Right. That saves a lot of time to like mentoring as well. And then not necessarily a newcomer, I would say like a junior engineer. And then even for senior engineers, it would be like, it will save a lot of time, like when you have tons of bugs to look at I'm

Interviewer2

Yeah.

And so but I have. So one of the other features that I wanted to ask about is, would it be helpful if let's say only 50% of the times we can get the correct answer. But the remaining 50% of the times would it be helpful for us to if we were able to accurately say in that 50%, let's say 30% of the times, we know that we don't have a good answer. So we say hey, we can't help you Sorry. And only about 20% of the times we may send you down the wrong path that would be helped Full if not like, could it be? Half the time? So we don't know where the the answer is?

Interviewee

Yeah. So I mean, it's better than driving someone to totally wrong location, then yeah, it's okay to call it out rather than like pointing to the lessor the error. I would say it will help more. Yeah, right. I mean, yeah, if you if you cannot, I mean, it's okay to call it. Or you can say, Okay, this is you can give a percentage of accuracy. Also, like you said, 50% of the time, I don't know 20%. So you can say, Okay, I'm thinking this is, like in a in a percentage wise, like, this is this could be it, right? I mean, if you can give a percentage, and based on the experience with the tool, people can then figure out how much to trust and not trust the tool as well. Right. So I mean, for for a bug. I mean, yeah.

Interviewer1

And What types of bugs would this tool e useful for? Easy or difficult? So do you think it is best suited for easy bugs? Or do you want to use it for solving difficult bugs?

Interviewee

So it totally depends on how much the bug the tool can offer. But I mean, definitely for easy bugs, if some simpler things can be pointed in the right direction. And even for like, I, it's hard to tell what exactly I would use it for. But wherever I can get, like you said 50% of accuracy, I would get I can depends on the accuracy percentage, I would definitely like go with it. At least start with it.

Interviewer2

Assuming this works only for simple bugs, and will not work for cross system bugs and things like that, would it still be a useful thing? Yes. Yes.

Interviewee

I think it would be useful while filing bugs in those cases, right? Yeah, it'll be an easy knock off.

Interviewer1

So the only thing that developers will be the most beneficial for using this tool, or that will be the QA engineers?

Interviewee

Um, I think it's it's sort of working both ways, right? Like developers, you're definitely reducing the time taken to fix a bug. And then for, I mean, when this like, we speed up the cycle, it leads in like, their timing, most of the time, they are waiting for the next fixes to move on. They're stuck in something. So it helps both ways. I

Interviewer1

would say, Okay, and what do you want the output of this kind of thing? Like, do you want it in an issue management system? Or in ID of the developers? Or do you want output?

Interviewee

I would say an issue management system, because any anyone I mean, of course, whenever whenever, typically, when a issue gets filed, wherever I have worked so far, it doesn't directly go to the Developer immediately. I mean, it is it gets filed in a system, and then people pick it up, right? So I think when the issue management system gets it, if you if the tool puts everything in there, if it's integrated with those kinds of tools, then it will be more useful.

Interviewer2

Okay.

Interviewer1

So this tool we are talking about. So these kinds of tools require access to the code base . So do you think that would be problem for, like your organization ?

Interviewee

not typically see, because I mean, we do integrate a lot of tools, as long as it is not like consulting a server outside and like throwing information out to somebody else. Like, I think, as long as it is used within the organization as a library integrated into something then I think it should be. Okay.

Interviewer1

This is the final question of the interview. So, okay, as, as you add a senior developer, would you pitch to your organization for using this tool? Do you think that it would be that a useful tool? Yes.

I mean, definitely, if it is, if it is promising, as promising as you're portraying it to be? Yes, definitely. It really helps. Right? I mean, people will start I mean, but for any tool, at least, so far, to be successful, at least I think, to some version of it has to be open sourced. And then like, I mean, when people like start adopting it more and more, I think it can be like, monetized.

Interviewer2

And like the current state of the art is about 50% is where they are. Right? Okay. And one of the big challenges is that it's very hard to get this tool to work on your product if you have not trained on your product.

Interviewee

Okay. So to start with, you're saying it will start at 50% and then it will go Keep learning on saying

Interviewer2

that it won't even get to 50% Unless we train it on your project. So kind of the like, it has to have some kind of context. Because the way you label, the way you name things, all of this kind of matters even like, not just a way, but just the words that are used, right, like if we train in, say, in a mobile app, but then try to apply it to, let's say, we train it on routers, big stuff, like number of bugs, all of this thing, then we try to apply it to a mobile app, it's gonna fail. Vocabulary is very different.

Interviewee

Okay? So so when, just to understand, so when you it's a piece of software that I would integrate in my product. So when it comes, it basically has nothing in its in its dictionary, right, like, and then it slowly starts working, we just let it learn.

Interviewer2

Yeah, so typically, what we do is we looked at, we look at the history, to see if we can connect the JIRA repository to The Commit repository, then we train on the history and then you can kind of start using it immediately. So in the past, were were bugs. So we look at okay, so this was the bug report, and this was where it was fixed. And if we do this for like, 1000s, of fire 1000s of reports, then we start getting some vocabulary. And so we can train, then we can immediately.

Interviewee

So in that case, maybe, would it be better if? Like, yeah, so giving, like a metadata file or something that someone who's integrating it can give you some kind of metadata that you want them to fill in and then start that would help it will it help it? Like, do it faster?

Interviewer2

Yeah, that is definitely helpful. Right? The now this is what is happening in the last I would say, five, six years. But this research has been going on for about 20 years now. Okay. And 20 years ago, there was no training. So it was more such. So it will work immediately, right. But they kind of capped out at like 30% 40% kind of thing. Because it was literally, if there was no common word between the bug report and the file, then it wouldn't find the file. Right, right. Right. Now, deep learning can make these inferences, but it has to know what inferences to make. There is some body of research where we go, let's just train it on billions of bug reports in open source projects, and maybe it will pick up the vocabulary. Now, it may be true, but we're not 100% sure yet. Okay.

Interviewee

So I think I think some kind of then the trial version has to be out there to show people off. It's like benefits, right? Yeah. Before before, it's fully like, you know, people trust it and integrate it.

Interviewer2

Yeah. So which kind of follow up to the previous question is, would you pay for this tool?

Interviewee

Yeah, so I, I mean, yeah, bigger organizations. It really like depends. But then the thing is, I feel if if someone really finds us for this tool, like, I mean, if it is prompt, if it shows promise, then definitely yes, it will be integrated, right. I mean, it's not like we do like, there are other like gdb and other softwares. We do like us, so I mean, of course some some of them are open source, but then yeah, JIRA at least Right. So.

Interviewer2

Yeah, okay. Okay.

Interviewer1

Do you have any more questions? No. Okay. So that was our final question. Thank you for joining us today. I'll stop that.