Interviewer1

Okay, the recording started. So I will start the interview with the demographic question. So the first question is How old are you?

Interviewee

39

Interviewer1

Okay, and how do you identify your gender identity?

Interviewee

A woman, female.

Interviewer1

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

Interviewee

PhD degree

Interviewer1

is that from CS as PhD in

Interviewee

computer science. Okay.

Interviewer1

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

Interviewee

Probably 12. Yeah, 12 is decent. Okay.

Interviewer1

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

Interviewee

Five.

Interviewer1

Okay, so these are all of our demographic question. Now we'll jump to our interview question. So the first question is, when don't we use the word bag localization tool? What do you think about the functionality of this tool?

Interviewee

Bug localization? So I have heard about that in a few discussions that I had with some of my colleagues and other friends and other companies? That primarily, it's it's a tool that it's kind of a debugging tool that whenever a report is submitted, which is more in regular language, like English, or human readable language, how can that tool takes in that very text based report and kind of point you towards the right place where you should look for bug, I would admit that I have never used it myself, I did have an opportunity to look at some features in Bugzilla. That was a tool that our team was exploring, wants to use. So I think Bugzilla had some some of these features implemented in that. So yeah, that's my general understanding of that, that it points you towards the right place where to look for blog. Now that right place could be the exact files or it could be a potential commit that was made in a series of like 20 commits or squash squashes? Yeah.

Interviewer1

Yeah. So to our definition of that is exactly right, that so in this study, we defined a bug localization tool, the tool that can identify potentially buggy files, given the bug report, or sectors or both. And suppose the repository contains 700 source code files. And based on the bug report, the total shortlist, a few files potentially responsible for the bug, that was output is a ranked list of the files sorted based on their probability of being responsible for a particular bug. And then the developers can start further investigation or procedure or fix procedures from there. Okay. So given the state of the art of such a bad localization tool, would you use such a technique

Interviewee

I would be given I have a PhD. So I have the curiosity bug. So I would be very willing to test out new features and new tools like this. So I would say that right now, because most of the experienced developers or I would call myself semi experienced as well, we are kind of used to doing the deep dive on our own. So if we have been with the software for longer if we have been primarily developers, or maintainers, then we kind of know which module or which files are related to which functionality and a bug report comes in. So people are usually pretty quick at identifying where to look at, where I do see a lot of utility for a tool like that is for people who are just joining a team, or who are just starting on a particular project or a product that they are not that familiar with, that could certainly help getting them better acquainted, saving them some time. So certainly a utility there. And also for fresh developers who have less experience in programming or software engineering, they haven't built that muscle memory on where to look for when a bug comes in. So those kinds of tools can certainly guide them well.

Interviewer1

Okay, so you have mentioned about these tools can , save the time, do you think any other benefits that can have from using the such kind of tool?

Interviewee

So I'm a little skeptical on how the exact tool would work. So I think the few discussions that I was a part of people were discussing about various probabilistic techniques, text matching techniques, machine learning techniques. So it really depends on how how good that tool is, what's what's the false positive rates, what's the false negative rates, how many times that tool is actually helpful, and how many times that it's a false positive, which it points you towards something which has nothing to do with the bug, that that will just add to the frustration of whoever is using that. So how useful that will be I think it depends a lot on what type of technology it's built on, and what type of not, what's probabilistic guarantees we can make around it that 70% of the time, it will point you to the right file, something like that.

Interviewer1

Okay, so just a follow up question. Do you think false positive and false negative are equally bad? Or do you think one is much worse than the other one

Interviewee

So if it directs me, what false positive would be that it says that these files have the bug when they do not have the bug? So that would be going in one way. And the other one, false negative? How would we define that in this particular situation?

Interviewer1

The tool is saying that this particular file is responsible for the bug, but that is

Interviewee

the it's not there. So, yeah,

Interviewer2

it's the same output for both false positive and false negatives, really, because when there is a false positive, it's inherently a false negative to you. But you could think of it as let's say the first file is a false positive. And the second file can be a true positive. false negative is when all the files are false positives.

Interviewee

That's, that's a good way to quantify that. Yeah, yeah. So again, I would say it depends on how the overall experience is, okay, fine. If the bug is not in the first file that pointed out, but it's in the second file, and we have reasonable confidence that this tool points towards the right files most of the time, maybe not the top, its top recommendation, but the second or the third recommendation. So people do tend to get a clip. acclimatized, I think was that's the right word, to the to the tools that they are using. So as long as the expectations are set, right, initially, I think the tools can be helpful.

Interviewer2

Okay, I think part of you should ask some of the because the show kind of touched upon a lot of things that we were going to ask in the future, some things now.

Interviewer1

Okay, so the other thing, and the next question is, like, you have mentioned about one tool about in Bugzilla. So, but you haven't you haven't used that one. So is there any reason you can share with us that is that wasn't that helpful for your project or any other reason?

Interviewee

So at that time, we were actually evaluating Bugzilla as the bug tracking system overall. But then we decided to not go with that. So it was not particularly for that feature, or lack thereof. But it was that we decided to go with another system rather than the bug tracking system. But but we were evaluating it, I think it was called one bug localizer or something. I think that's what's the name. So at that time, we were looking at that, and we were thinking that could be useful. But currently, we don't have that. I think I just, it's my personal opinion that a lot of software developers who have had years of experience actually pride themselves in finding bugs. So giving them a tool sometimes makes them feel that okay, probably people don't have enough confidence in our abilities to track bugs. So that we have that they are suggesting to use external tools. So it's I think it's a it's a matter of a little bit of pride there as well, that this is something we have been doing for years, we can do continue doing that. But landscape changes, new tools become available, I think we should have an open mind about what can help us. Yeah.

Interviewer1

So do you use any other kind of automatic debugging tool in your process?

Interviewee

In my day to day? No, not at my current company? Okay.

Interviewer2

Not necessarily bug localization? Like what to do? Like when a bug report comes in? You're trying to fix that bug. During that process? Do you use any kind of automated tools?

Interviewee

Sure. So we do use a simple text matching or string matching, because we put a lot of emphasis on putting in what are those called? The the assert statements for lack of a better word. So a lot of very descriptive assert statements that that that logout, what classes, these are, what fires these are, what functions these are, and when the bug reports come in. Also, on the front end, we make sure that those errors are appropriately surfaced. So there is a lot of investment that happens both at the time of programming, and then also the front end projection whenever things don't go right. So that kind of assert messages or log messages help us tie back to the exact files or modules where the errors are might be coming from. And for that we use simple text matching. So there is the day I think a few of us use this script bash script that a couple of our developers have created, which basically does the simple this kind of like string matches, depending on these assert statements and log statements and tries to give the starting point of investigation.

Interviewer2

Excellent. Thanks so much.

Interviewer1

Okay, so do you think that is the lack of feature of your current automatic debugging process you just mentioned?

Interviewee

I'm sorry, again, I didn't.

Interviewer1

So do you think there is a lack of feature? Or do you think any kind of feature that would help you to speed up the debugging process in your current debugging system?

Interviewee

So the two things, one is that there is a problem, which is so pronounced that you feel the need to change things or speed things up. And the other is, you're okay, in the status quo, if there is nothing, which is glaringly in your face wrong, but you are open to evaluating newer methods to improve things. So I think we are more in the second bucket that there is no inherent need as yet. But if something comes up, and it's like, it's promising, then why not? Okay.

Interviewer2

And do you want to ask this performance question?

Interviewer1

I'm going to discussion,

Interviewee

performance questions. Okay, so

Interviewer1

the first question is, like, according to your perception, what are the minimum required functionality of a bug localization tool like, currently, they can just rank files based on the probability or maybe it text matching? So do you think that is enough? Or do you think there should be other kinds of output for the developers?

Interviewee

Good question? So I think something and it's a good question, what what part of that should be part of the bug report versus what part of it should be part of the the localization tools output, but the if we can move ahead from just the the bug gets moved beyond just the bug itself, that, okay, this line or this file, has incorporated this bug, if something could go a step beyond that, and really focus on what was the what was the user's use case? What were they trying to do? And what didn't happen? And the workflow that they were hoping for, which files or which functions are which modules could have could have ideally worked together to make it happen? And where did the break exactly happened in all that chain? That was that was supposed to work correctly? I think that would be really helpful. So it's not just what we have right now, like a text matching tool, like this file. This is our statement. Okay, cool. But what exactly went wrong? So if we can have kind of that workflow kind of visibility, this was the original workflow, this is how things were supposed to be. This is where it broke. And probably this is what has happened. And I think a lot of that could actually be improved bug reporting, as well. A bug reporting could be as simple as the assert statement or the the status code that's thrown out. But that could also list that what what happened, what was the expecting to happen? Any force, the error codes and other messages are part of that as well? What could you do to reproduce that bug? So work reports and localization tools? Both can kind of work in tandem to strengthen the overall visibility into what went wrong? Where? Yeah.

Interviewer1

So other types of output, like typical bug localization tool, I can give me his like web, similar bug reports that were fixed before that is from the history of the bug reports, or the best person in the team who is better fit to fix a particular bug building this kind of output is helpful, or do you think these kind of output will be helpful ,

Interviewer2

you know, I think the history could be very useful that something like this was experienced, so and so that that time or that person fixed it, so that giving them a name of a team or a person who might have experienced something similar, I think that will be very useful. And then what I was mentioned at the very beginning of our chat is for git based development, if the point somehow could also be pointed out to which particular commit this might be coming from not just the file, but commit as well. That would be helpful too because sometimes have this 100 commits squashed together and then like a nightmare to figure out, okay, which particular commit to look at it, revert or what to do. So that that kind of versioning information, if that could be added to a localization tool, I think that will be extremely helpful. Okay.

Interviewer1

So the next question is, what do you think is the lowest acceptable performance of a bug localization tool in terms of accuracy?

Interviewee

That's a tough one. Depends on I think depends on a few factors. One, it depends on how many senior more experienced resources a particular team has. If there are if there are newer resources, and there are not a lot of mentors available, that even a lower accuracy tool would, would would be acceptable. If there are a lot of senior resources available who already are, as I said, they have a lot of muscle memory. And then something but something would only be adopted if it has a higher accuracy threshold. Similarly, how, how easy the tool is to adopt how easy the tool is to train people on, those kinds of factors will eat up into the accuracy probability as well shouldn't be zero. It's difficult for me to just say an exact number.

Interviewer2

Yeah, just a conversation like this so that we get a rough idea about it. Right. And I really did not like what I think this is probably your next question is, how many files do you think is should be in the output? Like? is five? Potential files? Okay. 10. Okay, or one? Okay. Now, given that the more files that we can give, given the output, the higher accuracy number can be right? Yeah, yeah.

Interviewee

Your error is in your repo. 100% accurate? Absolutely. When you say how many files are you talking about? Like, how many files in the top guess because there could be multiple guesses as well, right.

Interviewer2

Or, so we'll make we'll go in, we'll get output will be that, like a Google search, it will give either only one link five links, or 10 links for a file?

Interviewee

I would say three to five, I think anything more than that might be might be too much. And even lesser if the whatever probability threshold is less than that, that if you can only find two good matches, then two good matches. But yeah, three to five, I think is, is pretty much it. I'm just thinking about myself that if I have this tool, or after five, I would be like, yeah, no, I'll search myself.

Interviewer2

And on the flip side, let's say that we let's say that we, from our from our approach, we kind of find out? We don't think so. So when you search in Google, you will get some results, always right. But they may not be helpful results. Google tries to make sure that it's not they try to give you helpful results. But maybe it's not. Right. So what if we find out that maybe we know that we're not going to get a good result? And we tell you, Hey, we don't know which file rather than actually give you a file with a low probability? Would that be better? Or would you still want to have like an output always,

Interviewee

I would personally appreciate more transparency, that if a tool is not able to predict a file with high probability than it says, so it can even be more transparent and say, Hey, I, the only result I was able to find is like, this is low probability result. But here it is, as well, if just if you're curious, but I would, I would appreciate more transparency here. And the tool, if it's not able to figure something out, it's, it displays that message that will increase my confidence in that tool. Okay. Okay, so

Interviewer1

our next question is about the latency. So what is the maximum acceptable latency for the output of the tool? Do you think the output should be in in orders of minutes? Or maybe hours is also okay?

Interviewee

Um, I was thinking more like in the order of minutes might be better. But then again, of course, how, how complicated your repository is, how deep it is, how many files it has, how many commits it has endured, that may increase the time to, but a few minutes should be okay. I don't think that's a deal breaker. I don't think we need anything which is in the order of microseconds or seconds.

Interviewer2

Because there are some approaches that we are looking at the literature where even for a repository of say, at the order of 700/1000 files, it can take several hours to output an answer.

Interviewee

Yeah, that might be that might be painful. But then again, if it's if it gives really, really good results, and that's the way to then market the tool, again, with with full transparency. And that's the amount of time you take. So you started when you are leaving for the day, and the next morning, you will have your results. So that could be something someday it could be used. Yeah, but transparency and messaging exactly what you're offering and what what the developer should expect is very important.

Interviewer1

Okay, so you are talking about transparency. So if the tool there's some kind of explanation about why this tool thinks that this particular file is responsible for this particular bug, so do you think that kind of explanation will be helpful?

Interviewee

So I'm kind of on the fence about that feature, because the excuse me, for the developer, excuse me, the for the developer, it matters whether he's being he or she is being pointed to the right files or not. It's a kind of like a binary vote for them. That okay, this tool is working, this tool is not working, why is it working? Or why is it not working? Maybe some curious minds might be want to learn more about that. But I don't think it will be helpful in the day to day usage.

Interviewer2

I think let me rephrase that a little bit, not why the tool is working. But other for a particular bug report, we say that file a is the top result. And along with saying file as the top result, we also say why we think file as the top result for this bug report, because some keyword matched or some, some other keywords, are there and file a? Or can we point to some something in the file that we say that this because of these keywords, or these variable names, or these method names, that we think that the fix is in this file for this bug report?

Interviewee

Yes, I have two thoughts here. One is that whoever is creating that tool, and whatever internal logic algorithm they have, will they be comfortable surfacing it to the user like telling them exactly how they are doing the internal calculations? Is there something they want to be patented and, and a secret or not? So that's one thing that I'm thinking. The second is, if we are talking about probabilistic models, or machine learning models, if you are moving towards higher and higher and higher accuracy, probably you're moving towards more and more complex algorithms, maybe deep networks, neural networks, who knows? And those are very difficult to explain. So it will be difficult to explain how your algorithm made that particular choice because neural nets, deep nets are not experts. I don't care what people say that they are not explainable as yet, maybe in 10 years or so. So those were the two tours. But even if it's a simple algorithm, would you want the world to know how you are ranking? Do you want it to be a secret or patterned and to us? Is it even explainable?

Interviewer2

That's actually two good points. I mean, for the first point, we did this, we did doing research, we would make everything publicly available. So like, most point, but the the second point is like, that's certainly something we're thinking about. Right. But so we may not be able to say how the algorithm works. But we might be able to say, what are the keywords in the file that pointed to this?

Interviewee

There's a little bit more visibility on the features that have been used to make that determination. Sure. Yeah. I would certainly find that information. Interesting. Okay,

Interviewer1

so what types of bugs Oh, this will be useful for do you think the tool should focus on the easy bugs, or it should focus on the difficult bugs.

Interviewee

I can see usage on both fronts. Yeah, even for four simple words, if it were simple bugs, it will be appreciated. If it's very quick, like if in front of you seconds that can tell us how these files, these us files, these Ruby files, go fix them. And for more complicated words, maybe some algorithmic changes, even if it takes a few hours, it can it can give you that kind of a pointer to where to start from. I think I can see utility both in both cases. Okay.

Interviewer1

So both in this bug location tool will be useful for developers. So do you think it would be best fit for QA engineers?

Interviewee

So I'm, in fact, think of the developer lifecycle in a traditional sense that you develop something QA engineer comes in and test testers break things, and then they produce the bug reports. And then those bug reports can then go into that tool. And the tools output can be used by developers. That's the workflow I'm thinking.

Interviewer2

Perfect. And a relative note Where do you think that the tools output should be should be should it be in the bug repository or in the IDE?

Interviewee

I don't have any immediate inclination on that. I mean, I cannot think of any pros and cons either way.

Interviewer2

Okay, yeah.

Interviewer1

Okay, so this is the final question of the interview, if a bug localization tool is a paid tool, so would you ever pay for such a tool in your organization?

Interviewee

If it's a paid tool? So most of the paid tools that come with like their trial periods and everything, right, so something that that is documented well, that how it's doing what it's doing, it sets expectations, and SLA is in terms of time and performance are laid out. Well, the the term that I used earlier, the transparency, that aspect is very clear from its documentation from its sales agents from its researchers, I would certainly be willing to give it a try the free trial or try. And then of course, the how much money can be or should be paid as that business comes in. And all the market factors come in. So who knows? Okay.

Interviewer2

Kind of like a tangential question to one of the things that you were saying, Do you think multi stage algorithm kind of thing where we try the simple solution, simple algorithm first, takes a few seconds, we see if we can find a high probability file. And if we don't, we give them the option saying there's this other very heavyweight algorithm that you can potentially run and let it run for a few hours and then give you the output?

Interviewee

I think giving levers to the audiences, I'm proud that I think it's always a good idea. Again, it tries back to the theme of transparency, you tell them that, okay, if these algorithms will approximately take this much time. This is the accuracy, we expect false positive and all that good stuff. But if you increase the time, we can try these algorithms. And then these are the metrics we can expect from that, and let the developer choose pick and choose. So yeah, I think that would increase the confidence in the tool, it will certainly increase mine, if I know the options and the grid I have available to choose my point in. Okay.

Interviewer1

Okay, it's actually a follow up question. So this kind of tool actually needs access to the code base of the organization. So do you think that that could be a problem in using this kind of tool?

Interviewee

No, as long as we can probably have an option of local deployment or deployment to their own cloud cloud, then I don't think it should be an issue. If it's going to a third party cloud or a third party service. sure people will will not like that. But a local deployment or deployment to one's own cloud, that should be okay.

Interviewer1

Okay, so that is the end of our interview. So thank you again for giving us the opportunity. I will turn off the recording now.