Summarized Transcrpt

Theme 1: Technical Debt Identification

4. Response: In their experience,

the stages in software prototyping involve understanding the software's purpose, designing, implementing, testing, and deploying.

Key challenges include continuous learning due to evolving tools, standards, frameworks, languages, and conventions.

5. Response: They become aware of technical debt when repeated changes or additions are needed in the code, indicating shortcuts or poor coding practices.

6. Response: Indicators of technical debt include code complexity, duplication, quality issues, and a lack of modularity.

Theme 2: Technical Debt Measurement

7. Response: They prefer measuring technical gaps by analyzing the time needed for fixes, conducting code reviews, and using version control tools like Git.

8. Response: Currently, they measure technical debt through code analysis, reviews, and stakeholder communication.

9. Response: They prioritize addressing technical gaps promptly to prevent slowdowns.

Theme 3: Technical Debt Impact Evaluation

10. Response: Unresolved software gaps negatively affect software quality, such as reliability, performance, and maintainability, by causing development delays.

11. Response: An example of how technical debt affected a project is neglecting design initially, leading to extensive changes later, impacting timelines and quality.

Theme 4: Early Debt Repayment

12. Response: To encourage early debt repayment, they practice error troubleshooting and seek assistance from peers or experts.

They plan to involve department heads for guidance before project presentations.

13. Response:

Incentives include supervisor guidance and using platforms like Stack Overflow.

Educating students on these resources motivates them.

Insight:

14. Response: In a technical debt management framework, providing tools for students to communicate with experts, guidelines for contacting senior experts, and education on using platforms like StackOverflow and ChatGPT would help them overcome technical debt effectively.

Full transcript:

Interviewer: I'll start the recording.

Interviewee: OK, so I'll introduce myself. Again, my name is Mugoya Dihfahsih, and I'm doing a master's degree in software engineering and data communication from Makerere University. And I'm carrying out a research. Basically, the main objective of this interview is to identify metrics for measuring technical data in software prototypes, involvement processes. And also, maybe to bring you to speed, what technical data is, technical data can be understood as consequences of taking shortcuts in involvement in software that compromises your prototype, right? Because we say it could be related to a financial debt where you have to pay after taking that debt. And this time, you pay with interest. The more time you take to repay the debt, the more interest you pay. The earlier you repay the interest, the less amount you pay as interest, right? So which would you prefer to pay? Is to pay as early as possible, right? So technical debt, for it, what it will do, it will slow down your involvement. It will require an extra mile or an extra effort to solve the bugs. First of all, it will also introduce errors or bugs in your code. So first of all, maybe it also, first of all, leads to poor coding practices. Maybe you're fond of just copying code, pasting it in your project. It works. You don't care about running the processes. You move on, so something like that. So it's really, it really affects students. Imagine you're a student and you've been used to copying and pasting past exams. Now you go to working world where you're given a project, yes, but you keep taking these shortcuts. You don't automate your code. You don't modularize your code. I assume that you're at the end of it all, you pay. Because imagine after five months, this code has grown big and you can't understand what you did previously. You have to go through again, so you take more time. And you need to document. So that's why they're taking the code out for you. You have to repay. You may repay in terms of maybe losing even a job because they're giving you money, they're giving you resources, and what basically you're doing is to learn something that you've already developed because you're easily taking, copying and pasting. You're never following the right development procedures. So I hope you have gotten a clue about technical debt.

Interviewer: Okay, great, so this interview is going to be themed in the themes we're going to have. First of all, technical debt identification, technical debt measurement, we're going to have technical debt impact, then we're going to talk about the repayment of technical debt. So you said, okay, I'll start by you describing what is your name, your course, and your role in the project that you're currently doing.

Interviewee: Okay, I'm Ssekamanya Julius. I'm a third year student. From Kyambogo, correct?

Interviewer: Yes, I'm doing BIS.

Interviewee: BIS, first semester, third year, first semester.

Interviewer: Okay, currently, I won't say, because there is a project that we are starting, but we have already divided the roles.

Interviewee: Okay.

Interviewer: Maybe what I could say.

Interviewee: What you're doing, because in social development, you say, I could be an advocate, you can be maybe a project manager, you can be maybe something like that. So, different roles, so you feel like, which role can you play in a software development?

Interviewee: I can do that, the overall.

Interviewer: Like overall.

Interviewee: Great, so I'm going to now ask you like over things, right?

Interviewer: Okay, so you're still so, and then you said, what is the, you said you're coming up with a project already?

Interviewee: Okay, I'm still learning some, there's some flat I'm learning, but I won't use it, use my app that I will use here. I'll use my project in third year.

Interviewer: Okay, so let me, in your experience, what are some of the stages that you go through when you're working a software product?

Interviewee: Okay, since I didn't know anything about Flutter, I just started learning Flutter like a month ago, during internship. Yeah, so you have to study Monday, study Monday, Wednesday, and Friday.

Interviewer: So I mean the stages, you know the stages of developing a software?

Interviewee: Yeah.

Interviewer: Yes, so what are those ones that you normally say, if I'm given a software, I start by doing this, and then doing this, and then do this, and then do this?

Interviewee: I start by like knowing what the software will do.

Interviewer: Yes.

Interviewee: And knowing if I can actually do like the coding of that software.

Interviewer: Yeah.

Interviewee: So.

Interviewer: Then I get the designs.

Interviewee: Afterwards, then you can implement.

Interviewer: Of course, the design, yeah.

Interviewee: Then you test and then deploy, right?

Interviewer: Yeah.

Interviewee: Okay, so what are some of the key challenges that you normally face, related to tools, standards, frameworks, programming languages? What are some of the challenges?

Interviewee: The challenge is that I don't know everything about the programming language. I have to always go back and watch and research.

Interviewer: So you said you're learning Flutter, right?

Interviewee: Yeah.

Interviewer: So what are the challenges you normally encounter when you're learning Flutter?

Interviewee: Yeah, just like I've said, like the instructor who teaches us, so he doesn't give us everything. Yeah, so I have to like go to the online and search, and those things are kind of complicated.

Interviewer: Complicated.

Interviewee: Yeah.

Interviewer: So, you've said you've understood what a technical data is, right?

Interviewee: Yes.

Interviewer: So how would you normally identify these technical gaps in your software product?

Interviewee: Like I can design something, like today, and then like tomorrow, I realize I have to do it. I have to like maybe add something there, and it will involve like repeating the work. Maybe what I'm talking about, the techniques, it could be system review, it could be maybe contacting the stakeholders, it could be maybe doing the code reviews, their programming, as well as maybe directing all the supervisors.

Interviewer: So what are, what of those do you normally use when you're doing the code, when you're identifying technical gaps in your product?

Interviewee: Maybe the code review.

Interviewer: Okay, so you normally do code reviews with your fellow students?

Interviewee: No, maybe with my friend, I do it Flutter, so I'm going to use, like you can get code for charge

ability.

Interviewer: Yeah.

Interviewee: Then you put it there, but you don't necessarily put the code there sometimes.

Interviewer: Okay.

Interviewee: Okay, so these are indicators. We have code duplication, we have code complexity, we have code quality, we have maybe, and then code ownership.

Interviewer: So what are the indicators that suggest that you have a technical gap in your project?

Interviewee: Code complexity.

Interviewer: Code complexity, when that, you can't understand the code that well?

Interviewee: Yeah, when it's too long.

Interviewer: With a lot of codes in a single block?

Interviewee: No.

Interviewer: Okay, so let's look at the code measurement. How would you normally, how would you wish to measure these technical gaps?

Interviewee: Measurement, I look at analyzing, maybe using version control, like Git, maybe contacting the stakeholders, maybe doing the code reviews like you've said.

Interviewer: So what are some of those, how do I call them, how do you normally, how do you want to measure the technical data? This is how big my technical data is. Can you measure, like, what did you time? The timing? Like, used to fix it.

Interviewee: Yeah, because I normally do this in class, because I don't have a lot of code.

Interviewer: So you know GitHub or Git for version control?

Interviewee: Not really, but I use it very often in GitHub. I just have an account there, but.

Interviewer: You don't use it?

Interviewee: Yeah, but they won't teach us how to use it today.

Interviewer: Oh, okay, then that would be great. So which, actually I was going to ask you which kind of tools you normally use for measuring technical data, but you told me.

Interviewee: Yeah, I think it's time, because I waste a lot of time trying to fix those things, I understand it. Yeah, because I take it upon myself to, even though I get the code from somewhere else, but I really want to understand what it does. It takes you a lot of time, man.

Interviewer: So how do you normally prioritize technical gaps to address first in your software development?

Interviewee: Prioritize? Prioritizing, I mean, which errors to, we talked about technical data as being shortcuts, could be errors in your system, so at which phase do you normally want to address technical data first? Immediately, immediately I realize one. I don't want to waste time, because the more that you do it, the longer you spend.

Interviewer: So in software development, we have what we call SDLC, you know about it?

Interviewee: Yeah.

Interviewer: Software Development Life Cycle. We have requirements gathering, requirements analyzation, design, implementation, testing, deployment, maintenance, so at which of those phases do you want to measure technical data?

Interviewee: I think it's during design.

Interviewer: During design, that's when you're doing the coding. That's where you should be measuring technical data.

Interviewee: Okay, that's good. So we go to the impact of technical data. So how do these unresolved software gaps affect your quality of the product, maybe in terms of reliability, performance, and maintenance?

Interviewer: Interviewee: First of all, they slow down the development process. Because you have to go back always and work and understand what you are doing or try to fix. So they slow down the development. Yeah, they also need production of duplicate software because many people use those things, your resources. So if you go to somewhere, you pick code, you just choose, and you end up producing the same thing. The same thing, okay. So that is a form of maybe how reliable, maybe how performing your code is because you just put it somewhere and it's not performing as expected, something like that. Yeah, because there is, okay, what I usually do, if I put somewhere, I tend to learn it, and then I apply it. I don't use that very thing most of the time. I just apply the knowledge and then I maybe change it to something else. Oh, that's great. You do something that is wonderful that most people even don't do. People just copy, put it in. Yeah, but from the same source, but now you change something. Okay, then that's great. So can you provide an example when you're coding, maybe you're working on a project, that you realize this was a real technical data? Yeah, they look very, when Patrick was working on my UI, the first design I created, very fast, because I didn't think about the effects, we wanted to put some effects in the layout, but I didn't think about them. So it came to me, this week again, and he said, like, we're doing a lot, we don't want them, you know? So until now, I'm still doing that same thing. So you realize it was a technical data? Yeah. Yeah, okay, so let's go to repayment, because we say, we talked about that everything, even if it's a financial one, you have to repay. So are there any practices or strategies that you know of that encourage you to fix these technical data as long as possible? If it is errors, most times, if they encounter an error in my code, they first try troubleshooting, then if they fail, they forward to those three years, or there's some cut-apart from that they usually go to, they submit queries there. I mean, like, in an academic setting, like in this university, do you know any strategy that you know of that encourages students to fix their errors, their projects, as soon as possible? Maybe in the DI, they heard that when they're developing projects, they communicate with their supervisor. So they keep in sync with their supervisor? Yeah, and he guides them in which way to go. Ah, so I believe that if an error is sensed, then it can be... Okay, actually, that one is, I think it's contacting the supervisor to help them identify, yeah. So what are some of the incentives or mechanisms that can motivate you as a developer to fix your technical data as early as possible? I think... I think... Maybe, like, on this project I'm doing, I'm thinking of introducing it to someone in the department, maybe the HOD, so that he guides me earlier, before what? Before the actual presentation of the project comes, and next semester. So that's how, and when they reach that time, their project is in line. Okay, oh, that's good. Then maybe, what could be, maybe remember this is going to be like a framework, it's going to be like a set of guidelines to guide students. So what are the things that you would wish to have in such a framework? Maybe... Like, something that will help students to overcome those technical... I think a tool that enables students to communicate with experts. Because experts really help. So, like, you should include guidelines for students to contact maybe experts who are senior in what they are doing? Yeah

, and that's where errors can be resolved. Because something you are facing right now, there's someone who has already solved it somewhere. Maybe also, you can also try platforms like, they usually use Stack Overflow. Stack Overflow, okay. Yeah, it can provide solutions. Maybe like, what they're saying, they can call it guidance, or students should be guided on how to use such platforms like Stack Overflow, how to use ChatterJB to solve their problems, not just copying, copying and pasting. So students should be educated on how to use these platforms, right? Yes. Okay, thanks so much for their... Actually, they've been a lot of insights in what we're going to be doing. It's going to be helpful for the research they're working on. So they just want you to provide me maybe their name, their contact, that name, because, yeah, hoping for the follow-up. Okay. Thank you so much.

---