

video

Sun, Jul 21, 2024 2:12AM • 36:35

SUMMARY KEYWORDS

tests, metrics, uptime, quality, project, bugs, requirements, part, business, people, regulations, implemented, software, team, research, frameworks, mentioned, limitation, issues, feature

00:02

Hello there, good to see you.

00:03

I come in Nice to see you as well.

00:07

Thank you for

00:10

taking this opportunity and helping us in our research.

00:16

It's great to help you and support you. So,

00:22

then let me start with the, our paper, basically, I will inform you about the research first. So what we are doing, we are basically analyzing securing quality in software development process. But we try, we have convention to make this research related to the FinTech or financial industries. That's why we basically decided to interview software engineers, project managers from financial industries.

01:01

Is it okay for you?

01:04

Yeah, it's okay for you, Hey, it's

01:06

okay for me.

01:10

Okay, then, let's go through the consent form first,

01:15

because we didn't have the opportunity to talk about it.

01:20

So in the case you didn't signed consent form, I will just walk through you through the consent form and ask your approval, basically,

01:30

or verbal consent. So

01:34

as an introduction, I would say that I am master student of University of Tartu. And this is my master thesis research. And the objective of research is identifying how software quality is, is implemented in model financial products. So, and this study aims to develop a framework for basically securing data quality in this industry. The framework will be organized by help you software engineers, project managers, software engineers and test. So mainly with the people who design or develop software. So we have this participation requirements that you should be at least 18 years or older. I know that you are older than 18. You need to have a work experience in the financial sector at least one year to have it.

02:33

Yep. I'm 25. And I have worked as a software engineer in banking, mobile banking application. And I think it fits this requirement. And

02:50

great, and what about the experience with software quality? In your basically industry,

02:57

it was a highly required skill. And then we implemented software quality while building our products. So I think it also fits

03:08

great. So it basically means that you already have all the expert, all the participation requirements covered. interview will be approximately one hour. And what about the risk and benefits, I would say that we have the privacy and confidentiality in a high level, we protect our interviews. So what will be here. So after this recording, we will basically take the transcript out and we will remove any reference to the company to the position that you worked in. Maybe any kind of information that that makes you targetable. So everything like potential identifiers will be removed and aggregated. Do you agree to take part in this participation? I need to say that it's completely voluntary. You don't need to do that. If you don't want

04:15

yeah, I understand it. Yep. I would like to support you and help you with this research process.

04:21

So yeah, we can continue with the process Sorry,

04:29

what else should I do?

04:32

Nothing you basically right now consented for recording. And I I will record this session in order to

04:43

have a recording to transcribe.

04:46

Okay, great. Thank you.

04:48

Let's go through the interview questions. Thank you for being here. So, first of all, I need to say that the Our main focus of research is the security and quality, like software quality or software development process in financial industries, so we're not analyzing customer data quality or other things, we only analyze the software quality. And nothing else I see, I can start with questions. So our first question is, so you, how is software quality of the projects planned in early design or planning phase of the project? Okay,

05:42

so, we do Agile development.

05:47

Like we create user stories,

05:51

start working, thinking about what we need to do, we will have for this project. And then in this process, we also consider how we can test the implemented features. So like, from the customer perspective, we have to be sure that the implemented feature is what it is, how and like, it works as promised. And like we deliver the high end product without having any bugs any issues. So in the way, when we have these epics, because we discuss it with our QA people, as well as our engineers, and then we try to think about edge cases, what kind of issues can happen there, and then have some buffer for finding out these issues and then fixing them before delivering the project. And then while doing this planning phase, consider some buffer timeline for our project.

06:59

Okay, sounds great.

07:02

It's actually answers most of my questions. What about like, when you do the requirements gathering before the working project, like designing project, do you take into consideration like a prioritized software quality or you basically prioritize it in the development process itself.

07:25

So like, as you know, if you have,

07:29

like 1000s of

07:31

more than that users, then you need to make sure that the product that you delivered is high quality. And it's really hard to fix mobile application bugs in the client devices. So you have to make sure that there is no box because you have to release a new mobile app, and then it should be delivered to the clients. And it takes a lot of time. For the end users. So our stakeholders, that's why require us to build flawless applications. Like they understand that it's really important, and it's high important. And we have to make sure that whatever we deliver is flawless can work without any issues. So it's high priority. And that's why we consider software quality as an important artifact, and as an important point, and we try to dedicate quality assurance people to test out our products. And taking the requirements from the business side. Yes, we don't have such like clear requirements that like how we have to make sure the product is high quality, but they say that like it should be high growth and without bugs. And then our technical people should analyze every detail, every integration point and understand that okay, what we are doing is right, it doesn't cause any problems.

09:00

Okay, so how do you define the quality goals of the project? Like is it like non functional requirements, acceptance? criteria is how it's happening generally.

09:12

So from business level, that's like, we have this uptime of service and also the number of bugs that reported in the quarter and also the time that how much time does it take for us to respond on the box or on the customer issues. So this is mostly the quality goals for the project that actually should be manageable. And depending on the of course, type of the project, like they might vary, but this is the overall quality goals are defined in this way. Like we have automatic automated tool that collects all these metrics and and reported, visit the time spent at Hey, there are number for example 500 errors, and it is much arrows, it's more than the

10:13

special threshold that we have to report it.

10:17

Okay, what about do you have any specific methodologies or frameworks, like from the beginning of project like, do you have some checklist by which you define the goals or

10:32

what we follow agile and then we use Scrum framework. Okay? Agree we have user stories and every user story, you have acceptance criteria, and then where we define this requirements, like certain code quality goals, or like feature related targets are mentioned, as

10:52

well as

10:56

we have quarterly goals provided from the business and technical people so that, let's say we are gonna increase our test coverage for quarter or like, we are gonna have certain thresholds for integration tests, this kind of technical quality, things are added to the project.

11:21

Okay, that actually makes sense. What about my functional functional or non functional quality metrics, generally used by your team? Can you give a few examples like which ones generally in almost every project, and they needed?

11:43

Okay.

11:45

from a functional point of view, we have,

11:50

as I mentioned, we count the number of defects

11:53

the process, so we have revenue to finish the development, you will put your tasks in the quality assurance part. And then our create starts testing out everything in the business flow. And then they evaluate, they find defects. And then the number of defects that found in the project is one of the metrics measured for the creatives that like they're really good at. And for the developer teams, it sounds a bit mindless that like your code has defects. And there is also bug bugs that come from the production, then there is a ratio that like, how many of the bugs and defects exist. So like, let's say five bugs exist in the production and then 20 of defects have been found in that release cycle. And this ratio, like how many of the possible issues are found, before it goes into production, and it is, it comes from the production and non from non functional point of view, again, we have like up service uptime, that is the metric, the number of arrows happened is different, like 400 Arrows, 500 arrows, and

13:17

also a performance evaluation like how,

13:21

how much time does it take to respond to requests per second, per millisecond? And this kind of technical, quality metrics? And as well as also we have code quality metrics, such as how many percentage of the code is covered by unit tests? And how are they like, do we have integration tests or not? And if we have integration tests, which parts of the source code is covered by integration tests? And also there is sonar coop plugin which checks like flows and arrows, like some places that happen? And this kind of coverage is shown?

14:05

What about the challenges? Like How challenging is like what challenges did you face when implementing these metrics? For example, 99% uptime, or response time, lead time, basically?

14:19

Well, like

14:26

for example, I remember one case that like there was in the source code, some error, some search part error codes, were mapped to the 500 arrows. It was not our fault, it was a third party's fault. And then like this, and there was one out gauge with the third party, and it automatically hit our error count. We have lots of 500 errors. And then from the business side it was looking at like our services are problematic, but actually the problem part of Third Party, and like this metric started causing troubles for us. And regarding the challenging part, again, making sure that like you have 99.999%, uptime is really hard that you have to be, you have to play with your technical architecture a lot. And like, you have to analyze everything before doing the implementation, and it costs you time and time to delivery. And also, we hired the ops person, dedicated DevOps person to the project to implement those tools, and measure these metrics and then create nice dashboards to see these reports. And then creating those dashboards with the limited option was also hard.

15:55

I agree with you, I also saw this problems.

15:59

How the mic, these metrics differ from project project, for example, like, if it's mobile application, it should have different metrics. And if it's web application should have different metrics, internal tools, or different metrics, how it's happening there.

16:15

A I don't have that much experience with the other teams and as applications in that sense, I cannot answer it exactly. But up to my knowledge. The key is that, like, Yeah, we were actively looking at the mobile application and as well as back end part, and backend oriented goals were separate. So for each microservice, we define the topic, different uptime and different metrics. But overall, like we were measuring the same as I mentioned, same properties. And for mobile applications, again, like for each UI page, for, let's say, about test coverage, like some for some UI page, we have test coverage as well. So for some of them, we don't have done to go, Okay, let's add, let's say 50 percentage test coverage for this use cases, but for other cases, like it should be 100 percentage covered, because it's very likely

that some issues can happen there. And we want to detect it earlier. But in general, yeah, like, more or less, these metrics are similar in every project, I will say,

17:23

Okay, what critical aspects of software quality are generally prioritized by your company, for example, test coverage, or uptime, or something else, which is, which has more priority in the company.

17:43

I mean, they all are important, actually. But if we consider business level approach, if we have already delivered the project, and it's already in production, then this uptime, and then the number of bugs and performance and response time is highly valuable,

18:10

quality metrics for the company.

18:13

Have you ever come across with, let's say, software quality, like aspects that conflicts with business objectives, for example, team says that we should measure this and this person that should be there. And business has a different idea about it. Generally, CONUS, vice versa business say something and deaf team says we don't have that much. Like, we can't really wear this.

18:41

Well, that kind of situation has happened. Like business says we want it for example, in a week. And when you discuss it with the team, you realize that like it's not possible to deliver high quality product in a week, you can deliver something that works the product, there is likelihood that there will be lots of bugs. And also, it's impossible to test the implemented features within this timeframe. Yeah, this kind of situation happened.

19:09

So generally, business wants everything faster than under.

19:14

Exactly. They just say that, hey, it's just one day work, why takes one week and then you explain a day, you know, we have to consider certain credit attributes. And then we have to do testing, we have to do proper testing, especially we initially if you have just manual testers, you don't have automated tests, which also takes time. This manual testing again, testing error flow in the application also takes time. And vice versa. If you implement automated tests, it will take more time and again, they will not be happy. So if you want something that meets the quality requirements, it takes more time and then businesses not happy about it.

19:55

Yeah, I agree with that. I remember you mentioned about summer What other frameworks tools methodologies? Are you using? Like vital for the project that you have in financial industry?

20:16

Well, that's how can I share it without leaking the information? Okay.

20:25

Like, we have this sonar cube.

20:29

And in every developer machine, we have sonar, lint. And in the mobile part, there is also similar frameworks, I think it's called Kaylene, for Kotlin. But I might be wrong, as I'm not doing the mobile development right now. And like in general, you have some tools that checks that your code is properly

20:54

used. And then

20:58

we use Kibana,

21:00

ag Akustik stack, and like this, every micro service micro service logs, the issues or errors, and these error counts come from that logging service. And again, request response in this network layer. The measure this 500 400, another. Arrows and discounts are also shown on the dashboard that we have.

21:32

What about the

21:35

Yeah, sorry.

21:37

But from metal side, I mean, this is mostly measured by the

21:43

team leads, and also

21:50

agile the office, they collect these requirements. And that's under the quarter we discuss it.

22:01

More about this like?

22:04

So basically, what key advantages and disadvantages do you see in automating basically tests, or quality metrics in software development? One disadvantage, you already mentioned that it takes time. Yep. So

22:27

it has, as you mentioned, advantages and disadvantages. First of all, like automatic, previously, when we started, for example, the project, it was manual tests. Let's say in the first time, yes, it works. And then the second time, if you do the second release, again, you have to test everything from the scratch. And that enormous amount of time and effort if you check everything manually, but if you have automated tests, this you save some time, and then you have you are sure that like this testing will take so short period of time. And you can use continuous integration, continuous delivery. And like you can add them to your pipelines, for example, in the Git lab, or like in Jenkins. And those tests can be run in every commit or in every release cycle, and can save you a lot of time. So these are the positive sides. But writing those tests, and then maintaining them and making sure that those tests are actually covering the correct quality requirements is hard, especially dedicating time to the testing. It increases the delivery time. And then business stakeholders might not be happy.

23:50

I agree with that.

23:52

Have you? Like

23:56

have you ever

24:00

seen any limitation you encountered in automate automation part of the testing or say problems that comes with automating? I can tell one use case that in my projects, I saw several times that so end to end tests take enormous amount of time. That's why they need to run it at night automatically. This kind of limitations.

24:31

Well roll Yep.

24:32

I mean, it also depends on the level of tests. For example, if you are taking like unit tests, they are straightforward. They're very quick. But if you have integration tests or end to end tests, then you need to set up a dedicated environment. And then it takes an amount amount of resources that developers cannot check it on their machine, then we have to provide the environment And I remember, like Previously, we had like development setup, and also production setup. And testing was also happening in the development setup. But the productive part is that if multiple teams work in the same development environment, which is, as it's mentioned, it's development environment, but it's used for testing. Time Zones, automatic automatic tests, were failing frequently, just because of instability in the environment. So we had to dedicate a new environment for it, which cost hardware resources, also, some DevOps time and things. But it increased the overall test quality, or like, stability in stability of the results of tests. And, and as a

25:51

limitation was that

25:53

in certain cases, you have to do some operations with the third parties. And for testing purposes, you cannot do it. And then they don't if they don't provide the testing environment, you have to set up that testing behavior in your own environment. And that's sometimes really tough. Like, it's kind of building a new project, just make sure that your testing environment works perfectly. Like I remember, it took us really a huge effort to have such feature for one off services in our in house. So these kinds of limitations are there. And even with this effort, it was really hard to implement fully end to end tests in this environment because of certain changing variables that were dependent on other practice.

26:47

properties.

26:52

It gives me flashbacks when you're talking about this.

26:58

One about, like, what about priority or software quality on new future releases? Like, is there like some metrics that as acceptance criteria that if you don't meet, you cannot roll the future, basically, to production?

27:19

Well, yeah, definitely, like we have two week release cycle. Every release, you have to make sure that whatever is included in the release have been tested, like manual tests have been done. Automatic tests are like definitely in your in the pipeline, all the automatic tests should pass. And as well as there are certain scenarios that take it should be manual tested, all these steps should be done before the release, otherwise, you cannot do it on that release. And then you have to move it to the next one, which is also headache. And like, we try to make sure that whatever we have included in the release have already been tested. And then like we have enough time to ensure that the last items will also be tested and then can be part of the release.

28:08

Okay, got it. So you said actually, one one great thing that if if you don't meet the requirements, software quality requirements, you cannot prove the future and you need to move it basically to the next release. How much time did it happen? Is it pretty common saying?

28:32

Well, like when we move to this system? Yes, it wasn't a bit tough. And it happened a couple of times. But once our team started getting used to this system, we started changing our planning, based on the on average timeline that like we are sure that okay, is in two week of period, it's possible to deliver this much stuff to test these things. And we have this many people in the team. Once we learned it, then it

started decreasing like once in a quarter or something like that. Okay, the calls on NVMe it's our data good times.

29:17

Great, actually, you might want great point here that like you said, there is a decision making process behind prioritizing quality over the new features basically. So it came needs to decide, do we do this quality? Can you describe this decision making process behind prioritizing quality over speed for new features?

29:48

Hey, like it comes from the top management like this. They say that if we deliver the project, we have to be sure that it works for works nicely. And then if there is there is no bugs, troubles. But like mostly we try to fix every issue that we came across and then do the deliveries. But of course, like, if something is business critical, and bug or issue is like very small doesn't if there is no such big impact on the system experience, then in certain cases, like it might be possible that we release this thing, and then we fix the issue and then we do the next few days later. So that existing peak new feature is already in the production. And users can get benefits from the system, but are all generally taking pure Being quality oriented from the management, and as well as team members are interested in this approach shows that whatever we deliver, we deliver nicely. And I'd like it's measured from the management side. And they always, they're always interested in to increase this performance and uptime and increase user satisfaction. So all the decisions come from the management side, and then teams are helping them to decide if it is really a high impact issue or like, it's okay to move forward with it and then fix it later on. It's not at all it's not a bug at all, it's feature

31:54

pretty common thing to say. It's not bad, but it's feature. I have one one question left about regulations. So, I know that financial industry is under heavy regulations, because it deals basically with manual people and how the organization ensures compliance with local or global regulations related to the software quality of financial projects. And you seen that

32:35

in general, like

32:40

a development team is mostly aware of the features and and what should be done, but like product owners, and like business analytics, business analytics people have this information that like what are the new compliance requirements and like how the system should behave based on that they provide this user cases use cases scenarios that like hey, the system should work this way this information should be logged this should be accessible for limited people, this kind of requirements comes from the product owners and then business analytics. Also there is like dedicated department for the regulations that they do our audit in our system and also they analyze this regulations and then provide the guidance also there is a security team who has this ongoing security analysis and then they also provide detail these these these things are not meeting their quality criteria is in terms of security, our like this can be improved, like this regulation comes from the different perspectives and also there is

local authority and local financial institutions basically share their guidelines and then also requirements, so that you have to comply with these these these requirements and regulations.

34:13

Got it. So, basically, if we are talking about GDPR or some other regulation, there is so hierarchy starts with business people they like when they design Yeah.

34:29

Yeah. Like basically a business people sense that like there's this, like, it comes from the government institutions or like responsible organizations, basically, to the financial institution. And then the business people already have this information. They analyze deeply, what are the low and what are the change and how it's going to affect our system. And then they share it with the team that we have to build this thing. And once we build it, again, they check it with our team members with the quality assurance people that

35:13

is it really correct.

35:16

Okay, great.

35:19

So it basically starts with business people and ends with software developers testers. And in the middle there is security demo checks, basically, how the team is doing, how the features would go to work basically.

35:37

Team internal audit, which also have these

35:43

regulations

35:45

on various the regulations, the law, like the to audit system, and then they also mentioned that like, yes, you meet these criterias it's fine or they find something they say that you haven't met these criteria, please make sure that in your system, this is also implemented.

36:07

Okay, got it.

36:09

I already got all the answers needed for the research. Thank you for being here saying thanks for participation and investing your time and our research.

36:20

Thank you. I hope you will get valuable results.

36:23

I already got them. I would say bye bye.