So, we start with “Software Delivery in EPAM”. As you all probably know that in EPAM we have 5 different values, it’s officially documented and in case if you want to combine them all together in one sentence it will be “Act as a team that values the individual with integrity, focused on the customer — striving for excellence”.

And it is all about testing itself because testing is not only like a simple testing. It is a part of the quality assurance. The quality assurance is making sure that the process is correct and each stage of the process, which is actually teamwork and everybody is contributing to this with use actually the individual part.

And “integrity” is also important because being a hero on your own is not really valuable and it can be drastically increased value of the output when you collaborate together and you provide something useful.

“Focusing on customer” is really important and that is also the part of the quality assurance. The part of the testing, it is making sure that we are delivering something that customer is expecting with a quality that customer is expecting. Because just imagine you being a customer like going to the store buying a new pair of shoes and they look fine. But at the end of the day in case if they are of insufficient quality it is kind of a tricky and in case if you need a new pair of shoes after the first day it is not something that you really wanted, not something why you pay money for this.

The “striving for excellence” is exactly the same thing. It is when you want to make sure that your quality, quality of your deliverables is as good as it could be it's really improving each time.

So, what is a value of testing. First of all, value of testing is about cost of the bug. In case if you find a bug on the very earliest stage for example during the documentation process, when we are just documenting the requirement, it will save us a lot of time when developers will waste their time on implementing something that is potentially processing a bug.

We will see actually tester’s time on creation of the test cases and actually executing them. After all, everything should be deployed on production. It is also time for the Build Engineers that also valuable and then it is time for the customer to come step in, find the bug, report the issue. Everybody is not satisfied and there are tons of work to do before we can test this bug.

That leads to customers and team satisfaction. Why the customer satisfaction is important we all know and the customers are happy when we have less bugs, that is obvious thing, that is clear. And, what about team satisfaction in this case? That also very simple. When you do have lots of things to review and rework, lots of things to argue within the team, lots of planning, nobody likes when he is planned for something, for example. Especially for the bugs, again for the developers. Finding the bugs earlier, finding them prior to anybody else and fixing them as soon as possible is also good value of testing.

Sometimes testing is just about finding the bugs and making the informed decision. For example, making the decision about to go or not to go live it is also can be based on testing results. If you know that like a Project Manager or a customer that the product was implemented, this is functional or this is not functional, this is amount of the critical bug that we have, you can make an informed decision or postponing the goal like for a couple more days to be able to fix everything, to deliver the better quality to the end-users. And at the same time if you know that yes, there are 5 minor cosmetic bugs in your system you can make sure you are making the right decision in going life and fixing this 5 cosmetic issues maybe later someday.

EPAM reputation. Everybody knows that EPAM is delivering with the good quality therefore we are valuable, therefore we are winning the duels, therefore we are leading the market.

In case will start delivering something with questionable quality, I would say that will start with our customers. Testing will find points for your own improvement. Because in case if you know that tester is usually finding the bugs that are related to some specific area, where a good point for you to put some effort and increase your skills in particular area. And the last point but not the least, it is a work/life balance. Nobody likes working overtimes, especially on Friday evening, especially on the weekends when you have tons of things to do that really more interesting and sometimes more important than work.

{Slide 5}

But, at the same time goal of testing is a little bit different. So, the goal of testing is not to find all the bugs because it is simply impossible. It is not to find as many bugs as we can because it might be useless. I can say that every good tester knows that there is always one more bug in the application that we developed. But the question is: “Which effort is required to find this bug?” We can spend like 6 months on finding one minor cosmetic bug. But, would it be valuable to waste actually all that efforts? I really doubt that. So, the goal of testing is to find and communicate all important bugs. That called risk-based testing and it is actually smartest approach, which is widely used on all the projects. It is not the monkey job.

{Slide 6}

So, what are actually testers doing? Testing process is starting with understanding the requirement. First of all, we need to understand what are we going to implement. Based on that we are creating the test documentation. It might be test cases, test scenarios, some checklist depending on a particular case. After the functionality is delivered to the testing department, we are performing testing. After the execution of the tests we report bugs. After bugs are fixed or at least somebody thinks that they are fixed we are verifying this and the process starts from the very beginning. Each iteration it is one and the same.

{Slide 7}

It does not matter which process you use. Because the same approach can be used on Scrum, Agile, Kanban, Waterfall, RUP and V-model, does not actually matter which technology you use, it is applicable everywhere and everyone should do. It is something the same way. It might be the false thing when you are thinking that testing is that simple when it can be described in 5 steps. But, it is not actually true. It is just on the high level. Because speaking about the development, it can be described the same way. So, what is development about? Understanding the requirements, implementing this and pushing to production. Even last steps of the high level. But, we all know that it's more and more complicated.

{Slide 8}

So, testing should be done by everybody. And, actually, it can be applied to different roles, the question is that people test little bit different things. For example, Project Managers and other management that can be in the project as we know that for a small project there is just a PM and it’s more than enough, but for project with 15 more people there might be other management, for example Project Coordinators. So, the object of their testing is project process. First of all, they understand the reality, then they are creating documents, the guidelines, communicate to everybody, then they implement this process and they track risks and issues as soon as they implement the process. Based on that they are improving and fixing the issues that they found, verifying whether it is working or not and then re-tracking the process. That leads us to constant improvements - doing better with each iteration, with each process, but each project.

{Slide 9}

The same thing can be applied to the BA. They usually are the second gatekeepers for the quality. They are testing the requirements, they are understanding the requirements, they are documenting this, they are answering the questions from testers, developers, anybody else. Then they sometimes test actual implementation. Usually it might be called the user acceptance testing and they can actually act as end-users. They are verifying issues and improving the BA process, because during the testing it might identify that the process initially was not really good. For example, when they were understanding the requirements, they haven not identified all the dependencies or when they were documenting the requirements, they did not put enough information into acceptance criteria. And anyway, they are doing some process testing of the project itself.

{Slide 10}

Now more things that will be more interesting and more applicable for you. What the developers can do from testing prospect? They also do some process testing, requirements testing by nature. But, the main idea is that the developers should test their deliverables. But, as a developer you understand the requirements to the implementation and you do the testing. You can do that code review, you can do the initial testing and then analyze harm issues and improve the development process. You can do code review by yourself or in teams with your peers. But, do the initial testing before letting the QAs keep in touch with code and look at the pages or functionality. Because otherwise it might be just useless maybe cannot open the pages they just report one critical bug and they are going back to you and they have to wait for the next build. On some project it may be easier as they have build in the regular basis like once per couple of hours. On the other projects where the one build is delivered per week, it might be the huge problem.

?

{Slide 11}

So, what you can actually do. Take your responsibility. Don not say that testing is something that you have testers for. You don not have to do this. Because you are not responsible only for the deadlines, but also for the quality. Nobody needs something that was delivered in time, but does not have sufficient quality like I was saying before remembering the shoes. You have got the shoes on time but what is the point if you cannot use them. Bug that you find before testing decrease cost and save testing time, and by the way improves your KPIs. Sooner or later, Project Manager or your Resource Manager might come to you for your next focal review or your next 1-to-1 session and they say: “so, we need to set up the goal for you to improve your skills; and how we can measure this?” One of the option that is obvious is to measure the number of the bugs you have made. For you being smart here the small tips, tip from my say, try to get there not only measure the number of issues, but also their criticality. In case if you are producing lots of critical issues that means that you need some improvements and in case if you are doing really well and there are some minor issues everybody will be happier. Bug pin-pong takes time, increases cost and decreases morale within the team. What is the bug pin-pong? When you are actually do not want to fix the bug that was reported by QA, you are doing the declining thing - it's not a bug, it is a functional assert. But, sometimes to be honest it is done not only because it is really so, but sometimes it just takes too much effort to fix it. And then the QA goes and re-open the bug or somebody has not fixed that bug like or fix the bug directly, but has not done anything, any analysis around that and that fix one bug causes more bugs. And for this I can promise that QA people will be unhappy and they will come to you or they will come to Project Manager, to team lead, they say and you will also get your reputation. Last thing that I want to mention here is finding your own bugs helps you to learn and develop your own skills. Don not think about this as an issue or a problem, think about this the bugs that you made and that were found as points for you own improvement. It might keep you and put you an idea to going to the particular skills or areas where you might need the improvement.

{Slide 12}

So how can developers test? You can check the compliance for the basic requirements when you see 2 conflicting requirements you can raise it as soon as possible. You can do the unit testing. From my own experience the best developers that those who worked with the implemented unit testing on their project even if it was not required by customers of the process. They want to make sure that their code is really working on the unit level testing. As I have mentioned before you can do the code review. It can be done manually by you, by your peers, by your team lead or at least it can be done auto check if you have a CI on the project. But, in any case you should open your page and see if it at least looks OK. Because you will be terrified how sometimes, how horrible they deliver builds, sometimes when you can not open the page or everything just screwed up, turn the page and there are no buttons and there is no point of them, putting the additional testing, finding more complicated, more important bugs. But, I would like to say that it is really difficult for developers by nature to find their own bugs. I would say from my own experience when I was already a couple of years in QA doing testing and knowing lots of things about testing, I start learning Java. And I was doing my homework and actually programming. It was really surprising for me to see how many issues there were found after that. Because I was more focused on making the functionality, working directly with it was described in requirements. Not thinking about the boundary cases, not thinking about something else, how can end-users use it. Because end-users may be sometimes very, let it be, creative and pushing the boundaries and thinking about something else.

?

{Slide 13}

So, going back to the testers. They test everything, they are kind of overloaded. They are the last checkpoint before delivering something to the customers (no pressure), but they are actually the gatekeeper. They also help other people and act as other roles. Lots of time they try to be project coordinator, they are doing sometimes BA work and sometimes even the developers work. That is the rare case but it happens. So, understand the requirements, create test documentation, testing, reporting bugs and verifying them.

**Testing strategy** is a thing that describes: “What should be tested?”, “Why should be tested?” Because the tests may be done for different reasons. Where it should be tested? It should describe the environment, no other stuff. Who should test? Because it might be the testers, it might be developers. In case it is not the UI thing, it might be done by developers. When it should be tested? It is really important to define some milestones, deadlines and consequences. Because it does not make any sense to test the documentation after the implementation. Right? How it should be tested? There are tons of approaches how we can test one and the same piece of functionality. And we need to identify what will be more sufficient in this case.

So, testing strategy basically combines lots of things together answering the questions and everybody is responsible for each piece, for delivering his peace with sufficient quality, to be able to act as a part of the process, part of the team, but still have your individuality. That leads to really great results.

{Slide 16}

I would like to speak about the thing that how developers can help testers. I truly believe that developers and testers are not a little bit just the friends who can help you to move the couch, but the best friends who can help you to move the body. You can help them by providing the clean code to reduce tempo bugs submitting, verifying, doing regression testing again and again in communication that implies here. Give the hints where the bugs can hide. For example, checking code on the framework level and saying that it is not only for one screen, but it should be tested on several screens or vice versa that is not the framework page that specific for this particular screen. You can cooperate to work out test data together and find more bugs sooner. And sometimes it is easier for you and tester to establish really good communication when you can even test on your local machine before committing the code or they can tell you just giving the list of issues that they found. So, you can fix them before even without the official reporting. It might be really useful in some cases. For example there is no point in creation of bugs when that is on your local machine. At the same time you can also help them by improving themselves, so giving them the feedback: “this bug was described really good”, “this bug need some more information and please attach the screenshot to each bug that you reported”.

{Slide 17}

Testing is different on each project. It depends on the process, technology, vertical, some additional specifics. And, even though it can be described in 5 simple steps. It is not the simple and only on the high level it is a same. But there is no actual project, which can be successful without testing. But will be no silver bullet but the custom implementation.

?

{Slide 18}

Testers are working together with you and there would be a mistake to say that you need testers only when functionality is implemented. Because they start working before the product start, define testing strategy, they are testing the requirements as soon as they have the first draft before the implementation. They are doing test design in parallel or even prior to development. On my current project there are the cases when developers are doing their estimations based on test design, because only in test documentation it is really truly described, which functions should be implemented. Probably, one of the problem is poor requirements, but lately there is a big tendency in switching to more Agile process, which actually have no good thorough documentation. Functionality testing. Everybody thinks about this as just the only case, but as we talked before that is not the only case, testing can be quite on a different stage of the project. Testers can also do not functional testing and it can be done depending on the situation. For example, performance testing usually is done at the beginning of the project, at the beginning of the iteration and at the end of the iteration to be able to compare the results or it can be done like couple times per year. UAT/acceptance support. Usually users, end users prefer to hiking to testers and they can not sort them, help them how to report that issues or it is not the issues or just a configuration problem. And testers are also useful during the retrospective because they can identify what lessons should be learned and what should be improved in future.

{Slide 19}

Testing approach that we apply are really, really different. It is now more than imitation of the dumb end user testing like a monkey testing that was in the very beginning of the testing maybe more than 15 years ago. Right now we have mobile testing, automation testing that can be done only on UI, web service and database level. There is really complicated BI testing, performance testing, other non-functional testing like security or exploratory testing. Tons of different testing approaches keep out of our scope and I will tell you about them later.

{Slide 20}

So, my part is done and thanks everybody! I was very happy to be the part of this program and I really hope that it was useful for you. Thank you and have a good day!

{Slide 1}

Hello! When speaking about testing we usually speak about different approaches, techniques, tools, maybe methodologies. Why do we need so many terms? Why do we need so many beautiful words in order to just clarify that everything works correctly? Well, first of all, we need to understand that testing can be very different. And to show you this I can show this pyramid.

{Slide 2}

You can see here that there is so-called unit tests. Unit tests are usual tests that are made mainly by developers to test their own code. Each line one by one, each function call one by one. This is the down level, the downside of all the testing activities. You can not be more detailed, you can not be more specific but you are testing everything in isolation. When we are building a program, when we are creating a product, we usually need to test how different parts of our application interact with each other. Therefore, we need to go up. From component to integration, from system to user acceptance testing. On each level, we have different aspects of what we are testing. On each level, we have different number of sets and different number of conditions. Testing them one by one allows us to control quality on each phase of our production and therefore we can go on with another approaches. We can be more specific on each level, testing the exact functionality that we need to test.

{Slide 3}

It goes like this. When we're speaking about testing we usually speak about functional testing, usual testing, manual testing. This is a testing that we must perform on each and every application we are building. This functional testing sometimes may be so-called isolated or so-called exaggerated in terms of controlling the quality. Believe me or not, when you’re looking on different applications sometimes it’s really obvious, is it working or not. But, sometimes you need to have a deeper look, a closer look on what's really inside. And therefore we again speaking about different types, different types of functional testing. Each one is pointing and aiming on different conditions. Each one is end pointing on different parts of what's really built inside of our application, how our functions are working, how they're working together, aren’t they creating some kind of discrepancy in their outputs and so on and so forth. Making deeper look creating those deeper insights in our application allows us to catch bugs where no one will expect them.

{Slide 4}

Non-functional testing although is also important. Non-function means that we are not testing if our function is working properly or not. We are looking on can it be working for a long time, for example, or under stress conditions. The most known non-functional testing methodologies and types usually are security testing and performance testing. We are looking on how our application is protected and can our clients use it with a great amount of end-users. So, to do non-functional testing we need to look not only inside of our application, but also we need to make a wide range of tests outside, creating external options, creating external events that will somehow affect our application.

{Slide 5}

The main battle goes between manual and automation testing. What is better and what is outdated? Usually when we speak about some pros and cons of each type of this testing, we tend to ignore cons. Speaking about exact locations, speaking about some details when the choice is obvious. But do not be confused, both manual and automation testing are useful here. Both of them have situations when they are the only option available for you to choose. If you look closer on pros of automation testing you will see that usually it tends to be a long-term investment, a strategy investment. You can’t do automation on a project that will last for months or even a week. You just simply want to return your investments. While manual testing can be started in half an hour after a project assignment. Again, choosing the right combination of those assumptions, approaches and methodologies can win you a number one.

{Slide 6}

But remember about cons. There are different aspects in terms of pricing, in terms of time spent, maturity of engineers and outputs that will be provided to you at the end. All of those cons can lead you to defeat, so do not be a loser. Be a very accurate in choosing right testing on your exact project. Do not believe anyone who will tell that only manual testing is your saver and do not invest in automation testing if you can not see its benefits.

{Slide 7}

Each type has excellent conditions when the choice is obvious. But usually we are working in a grey zone where only a mix between manual and automation testing can bring you the highest available quality level on market. So, the right choice is almost 90 percent of your final win.

{Slide 8}

Front-end and back-end testing is the separation very similar to automation and manual testing. Usually when we speak about front-end testing, we mean user testing, UI, UX testing, acceptance testing and so, because we need this main perception, it is manual testing. When we are speaking about back-end testing usually, we are speaking about data flows, we speak about some algorithms and it can not be done manually just because the amount of data is huge and enormous. So, it is mostly automation testing. There is no single application built only in front-end or only in back-end nowadays. So on every application you will have at least several pages of UI or you will have several configurations items that can be changed by end-users. And you will need to test them, you will need to be sure that those testing can be performed, that those applications can be used by people with either skill and knowledge how to use or without it. And it brings us back to non-functional testing. Can your applications sustain long enough working with unprepared users? Will there be any unprepared user at all or not? These all have to be verified prior to going on market.

{Slide 9}

Mobile testing usually stands aside. Mobile testing usually is very connected with the type of mobile device you are using. And therefore there are great majority and variety in different mobile application devices, in different networks, platforms. Mobile testing is about diversity. You have to be very diverse and very different in both your approaches and tools you are using.

{Slide 10}

Mobile testing is basically divided into 3 parts. Testing of native applications. Application build, for exact one exact application platform. Hybrid testing, when we have both web elements and native elements. And therefore we can use the most out of both software and hardware platform. And HTML5 or usual web application that are built completely separately from the market. So, they are not using any kind of internal application architecture, they are not using internal resources in straightforward way. They use general resources. Therefore, they can be run on any types of devices. So, testing mobile application usually means not only testing application. It also means testing hardware, software, network, platforms, protocols, even batteries in your telephone can impact how your application is working. So, the number of conditions you need to count, you need to remember about is enormous and it is all about the diversity. Working with different approaches on different platforms with different applications and devices can bring a lot of problems to you if you do not understand how this is built, if you don't know how to test and how to make a proper choice.

{Slide 11}

On different projects you can use different tools to get the most out of your efforts of quality control and assurance. Working with the proper tools sometimes is hard and working with software recommended by EPAM can help you in terms of getting an expert to help you to start with. We have great experience in working with these amounts and these applications on different projects across the whole EPAM family on all continents and all countries. There are people who will assist you with starting to use them, there are people who will develop them like Reporting Portal that is entirely EPAM application. It will help you not only to perform testing, but to gather results, analyze them and to build report that can be used not only by you as a tester, but also by your management, your customers and maybe even end-users. Working with them can help you in understanding how process of testing is organized. Working with them will bring some new questions to you about how quality on your project is built, about whether your application is stable enough or not. And also it will guide you to a new areas and new questions.

{Slide 12}

In order to find answers to them you can turn to suggested reading of this chapter. You can start with EPAM Test Competence Center that will help you with getting the most up-to-date expert knowledge in testing across whole EPAM. You can turn to read world-known blogs like James Bach or Cem Kaner in order to get the understanding of what is core of testing and how it can be applied on your project. And also you can start with Wikipedia page, with a lot of cross-references that will bring you to any piece of information available on the world. Just remember that building the quality is a team task. And you alone can not do what a team can do. So, try to ingest quality on the very beginning, use appropriate tools, make wise choice on which approach and methodology should be tested and used on your project.

What is Unit Testing? Unit testing is a low-level testing focusing on a small part of the software system. Dependent on the particular language we can test methods, object or modules with the unit tests. Typically, unit tests are written and run by software developers. Not by QA. The goal is to validate that each unit of the software performs as it was designed. Unit tests are expected to be significantly faster than other kinds of tests.

{Slide 4}

Why? The first of all unit tests don't need external infrastructure. No need in database connections and integration with other systems. As a result, unit tests can be run during development frequently and scope may be smaller and there is no need to run a complete suite.

{Slide 5}

Unit Testing refers to tests that verify the functionality of a specific section of code. These tests cannot verify errors in integration or in deployments as these are out of their scope. That's why we different types of tests.

Component tests. Component testing is also known as a module or program testing. It finds the defects in the module and verifies the functioning of software.

Integration tests. Integration testing is the phase in software testing in which individual software modules are combined and they are tested as a group. So it as tight integration of those systems and now it only those systems.

The next step is system tests or end-to-end testing. End-to-end testing tests a completely integrated system to verify that it meets its requirements.

The last step is acceptance testing. Acceptance testing is used to conduct operation readiness of a product, service or system as a part of a quality management system.

{Slide 6}

Unit tests can be and should be one as a part of continuous integration pipeline. They should be executed during build time. It helps to verify that existing code is not broken by any new change. If for some reason unit tests can't be passed it may the code should be reworked. Only when unit tests passed we can run other types of tests.

{Slide 7}

What are benefits of unit tests? The first of all, unit tests make development easier. If you have unit tests you can get immediate feedback about code changes. It much safer to refactor legacy code. You might refactor your code and if unit tests pass, most likely you haven't broken anything.

The next benefit is that you can test complex scenarios much quicker.

One more benefit is that unit testing insure code readability. And it encourages developers to write clearer code. Because if developer keeps in mind that he/she needs to write unit tests, he/she needs to organize her code in a different way.

One more benefit is that when you have unit tests, you know what exactly was broken.

{Slide 8}

At first you can think that if you have unit tests your development time will be increased. At the result the cost of the development will be higher. But, eventually when you have unit tests and unit tests cover your system, I admit that we have unit tests, actually, helps you to document use cases at low level. They have to identify the root cause of bug and more, and more important that they have to decrease cost of bug fixing because they can help you find bugs much earlier. As a result actually unit test lower maintenance cost and helps you to write code faster with less bugs.

{Slide 9}

There are few principles for unit tests.

First of all, unit tests should be fast. They should run quickly. When tests run slow, you won’t want to run them frequently. If you don’t run them frequently, you won’t find problems early enough to fix them easily. You won’t feel as free to clean up the code. If you have fast unit tests, you will feel free to refactor your code, to improve your code, to good read of legacy code.

Next principle is Independent. Tests should not depend on each other. One test should not set up the conditions for the next test. You should be able to run each test independently and run the tests in any order you like. When tests depend on each other, then the first one's fail causes a cascade of downstream failures, making diagnosis difficult and hiding downstream defects.

Repeatable. Tests should be repeatable in any environment. You should be able to run the tests in the production environment, in the QA environment, and on your laptop while riding home on the train without a network. If your tests aren’t repeatable in any environment, then you’ll always have an excuse for why they fail. You’ll also find yourself unable to run the tests when the environment isn’t available.

Self-validating. The tests should have a Boolean output. Either they pass or fail. You should not have to read through a log file to tell whether the tests pass. You should not have to manually compare two different text files to see whether the tests pass. If the tests aren’t self-validating, then failure can become subjective and running the tests can require a long manual evaluation. This principle helps us to include the unit tests in continuous integration pipeline. We have it pass or fail. So if it passed, it means that we can continue and run other types of tests. If it fails means that we need to revert Java code.

The next principle - Timely. Users should cover each use case scenario; it is good to follow Test Driven Development because it is easier to write testable code if there are tests before.

{Slide 10}

Another principles came form Mr. Michael Feathers.

He says a test is not a unit test if

1. It talks to the database

2. It communicates across the network

3. It touches the file system

4. It can’t run correctly at the same time as any of your other unit tests

5. You have to do special things to your environment (such as editing config files) to run your unit tests

Michael added that tests that do these things aren't bad by all means.

Often they are worth writing, and they can be written in a unit test harness.

However, it is important to be able to separate them from true unit tests so that we can keep a set of tests that we can run fast wherever we make our changes.

{Slide 11}

So, let's look at an example. For example we have a method that adds a to b. The method is written in Java. We have 2 input parameters a and b and 1 output parameter.

{Slide 12}

We can write a unit test for this function.

In this example I wrote two unit tests for 2 use cases: one for positive numbers, second - for negative.

As you can see the test consists of 3 parts:

1. Setup

2. Exercise

3. Verify

During Setup, we are getting ready to a test. In our example, we create an object of class MyUnitTest that we are going to test.

During Exercise, we run our method and we pass parameters to the metod.

And at Verify step we compare expected result to an actual result. And Unit test tests the method as a black box.

As you can see we have a such equal method that compares expected result to an actual result. And it returns either even pass or fail. By the way, if those steps were not done correctly, unit test can have test bugs as well. So, how would developers run those tests?

{Slide 13}

For example, I use Eclipse to write my code and eclipse has a special plugin for unit testing. And when I use this plugin and can run my unit tests I can run whole suit and I can run it one by one. And I can see result. At this screenshot you can see that all my tests worked.

{Slide 14}

As I mentioned before, unit tests do not use any integration and no connection to database.

Unit tests are supposed to be executed in isolation.

To achieve it developers would need to replace dependencies to other classes, in other words to create Mocks.

{Slide 15}

A mock object is a dummy implementation for an interface or a class in which you define the output of certain method calls. So let's look at an example.

{Slide 16}

In this example, we have a method that returns next month number. Our method has no input parameters, but it has am external dependency to a system data. We can emulate (or Mock) behavior of the calendar class.

To do this, I decided to use Mockito library that helps to create mocks with just one line of code. We just say that if anybody calls calendar.getInstance method with any parameter, this method should return 9 all the time. As a result we get rid of dependency and we don’t need to rewrite our test every month.

{Slide 17}

One of the important KPIs for unit testing is Coverage.

{Slide 18}

Code coverage is a measure that is used to describe the degree to which the source code of a program is tested by a particular test suite.

A program with the high code coverage has been more thoroughly tested and has a lower chance of containing software bugs than a program with low code coverage.

This example is created with EMMA which is a code coverage tool for Java. Code coverage tools like EMMA can be used as plugins to IDE as a part of build process or in continuous integration.

What we can see on this picture it is the source lines decorated with the following color codes:

• green for fully covered lines

• yellow for partly covered lines

• red for lines that have not been executed at all

Now we have 2 lines which are not covered and 1 line which is partially covered by tests. All other lines are fully covered.

Yellow line as you can see, it contains some instructions or branches in some case. And it seems that the unit test that was written for this line covers just one instruction.

{Slide 19}

What is the best option for code coverage? It is tricky. Code coverage is not a primary goal for unit testing. And it doesn't guaranty quality. But it is recommended to agree on some level of code coverage in your team and to use rules and Continuous integration tools so they can either warn you or even fail the build job in case of having decline in code coverage.

Let’s see the example.

These screenshot are taken from a SonarQube server’s page. SonarQube is an open source platform for continuous inspection of code quality.

In the first picture - amongst other metrics - we can see the actual values.

The second one shows the evolution of the code coverage.

The current value is about 70 percent, which is relatively high and it shows an increasing trend which is also good news.

{Slide 20}

But it is not worth having 100% because it requires too much effort and sometimes it is not even possible.

So, general recommendation is don’t test trivial code (e.g. copy constructors consisting of only assignments, getters/setters, etc.)

The idea that we need to test code with algorithm. And there is no need, it might be really difficult and low level of testing, overcomplicated code that needs refactoring.

There are some benefits of having code coverage metric.

If you have code coverage metric, it helps you to identify dead/unused functionality. And it is a great help when you need to decide what to be covered in a legacy project.

But again this metric is misleading.

It is very easy to write test that covers 100% of code but won’t bring much value.

{Slide 21}

One more really interesting strategy for unit testing is a Test Driven Development. So, Test Driven Development is a special process that developers can follow to write tests. It consists of multiply steps:

1. Write a test

In test-driven development, each new feature begins with writing a test.

To write a test, the developer must clearly understand the feature's specification and requirements.

It could be a modified version of an existing test.

This is a differentiating feature of test-driven development versus writing unit tests after the code is written: it makes the developer focus on the requirements before writing the code, a subtle but important difference.

2. Run all tests and see if the new test fails (most likely it will fail because it is a new test)

3. Write the code

The next step is to write some code that causes the test to pass. The new code written at this stage is not perfect and may, for example, pass the test in an inelegant way.

At this point, the only purpose of the written code is to pass the test. The programmer must not write code that is beyond the functionality that the test checks. It's tricky.

4. Run tests

If all test cases now pass, the programmer can be confident that the new code meets the test requirements, and does not break or degrade any existing features. If they do not, the new code must be adjusted until they do.

5. Refactor code

The growing code base must be cleaned up regularly during test-driven development.So, you might want to refactor this code and to optimize the way how you organize your code, how you write it.

So the overall idea that you repeat this cycle over and over and every time when you start writing new functionality, you start with a test instead of code.

{Slide 22}

OK, now let’s talk about Tools&Libraries. I would mention today EMMA for code coverage, SonarQube for continuous inspection and Mockito for mocking. There are a few more.

{Slide 23}

For example, in Java:

• JUnit is a simple framework to write repeatable tests in Java. It is a simple and basic framework, but there are few more

• TestNG is a testing framework inspired from JUnit and NUnit, but introducing some new functionalities that make it more powerful and easier to use

• Mockito and EasyMock are the two most known mocking frameworks

In our example with the calendar class we use Mockito framework

• Cobertura and EMMA are free Java tools that calculates the test coverage of your code

In our example we use EMMA today

{Slide 1}

Hi, my name is Vitali and today I will tell you about test automation. So, first let us take a look at our agenda.

{Slide 2}

Today we are going to talk about Testing in Agile, Test Automation Goals and Metrics. Then we will take a look at the Testing Levels from Automation perspective.

Then we will switch to Automated testing tools and finally we will take a look at the Test Automation Frameworks and methodologies. So, let us get started.

{Slide 5-9}

To understand what is Test Automation, its goals and the purpose we need first to understand the testing role in Agile. So, let us take a look at the Agile plan.

Here we go. So, here we have product backlog which is split into sprint backlogs and during the sprint we have some activity for building the application to perform the full lifecycle.

So, during the sprint, every sprint, we plan, develop, test and rollout the new release. So, at the end of the sprint we have some kind of shippable product we deliver to customer.

So, let us take a look at the typical sprint. It is too big sprint. Here we can see it on the slide.

So, let us go deeper and let us try to understand what kind of activities we can perform inside the sprint.

{Slide 10-14}

So, basically, every sprint is started from planning and finished with release and some kind of delivery for customers. So, in case we have 2 weeks sprint, it is 10 days without day-off.

So, during these 10 days we usually spend one day, first day, for planning and the last one - for rollout, for delivery of release.

So, in case we are developing new features let us imagine we spent 5 days for development and 3 days for testing. Before Agile we had much more time for testing.

But in Agile we have something about 3 days. So, let us now imagine the growing of the application and the life cycle of the project.

So, first at the first release we will have several features. Let us count 10 features per sprint. And at the end of the sprint we need to test those 10 features.

We have 3 days, 10 features at slides. But imagine at the end of the year we have sprint number 26.

And this time we need to test 10 features went to the sprint and in addition we need to test all the features went to all the previous sprints.

So, that is a new number and that is about 260. But we still have 3 days and we need to feed our testing face inside 3 days.

So that is a challenge and the Test Automation is the answer. So, the Test Automation allows us to run those scenarios, I mean regression tests, much faster than it takes manually.

{Slide 15}

Let us take a look at the automation engineer activities during the ordinary sprint. So, while developers creating new features, we can create test automation scenarios.

So, we can pick up some tests from regression scope and make them automated. Then as far as new features ready, we ready to start manual testing. In rare cases we can automate them.

But usually we need to test them manually, because we still have not enough time to automate those features.

So, the days number 7, 8 and 9 we spend for test automation scenarios execution and result analysis. And then in the new sprint we shall start from the automating regression.

So, that is the typical workflow of test automation engineer on real project. In addition to these activities, we usually collaborate with other team.

First, we need to understand the business requirements. So, that is why we communicate with business analytics and customer.

Then we need to understand the test cases, the steps to reproduce and the expected results, that is why we communicate a lot with test engineers.

And finally we need to understand the technologies stack, the application is being built and how it is better to trigger some particular controls in this application,

so that is why we communicate with developers. So, test automation engineer is usually able to talk with every role in the project.

{Slide 16-22}

So let us now talk about test automation metrics. We have several metrics in test automation. But let us take a look at the main, the most popular.

One of the most popular metric is Return On Investment (ROI). It came from the economy and it indicates how the profitable any investment is.

So, let us take a look at the following chart and see the Return On Investment for automation on a project. So, here we have one axis, it is Time, and another axis, it is Cost.

Blue line is Manual Testing. The cost for manual testing is increasing during every sprint because the amount of the test cases we need to test is increasing.

So, we need to spend more time. So, the cost is increasing permanently. So, lets a straight line. The Test Automation is the orange line.

So, it is growing much faster than manual testing at the beginning because we need to test the features manually and in addition we need to invest in automation,

we need to develop frameworks, some scripts. So, that is why the automation takes more costs than manual at the beginning.

Let us take a look at the 6th sprint. So here we have the automation line is crossing the manual testing. So, what is happened.

The automation started to return investment back. So, some steps, some cases, some test cases got automated and we are ready to run the scenarios in automatic mode.

So, we do not need to test them manually. So, that is why we reduce the time spent for testing phase. And after the 6th sprint the automation cost is continuing to reduce.

So, following this chart we can make the following resolution. In case of automation, cost is less than manual testing.

So, in this case the automation is profitable and in case the automation cost is still higher, for example as for 5th sprint, in this case probably we do not need to automate something.

{Slide 23-29}

Another metric is Coverage. Coverage of manual tests by the automation. So let us take a look at the following chart.

On the one axis we have the Test Number covered by automation and on another - we have the Costs for the scope we have got automated.

Definitely, we can automate everything. But let us take a look at the Costs. In case we have 100 tests, the automation would cost 100 units.

But some cases are really expensive to automate. Some of them are cheaper at the beginning, but some of them are really expensive.

And in this case we can cut some really expensive and complex scenarios and do not automate them. We still can execute them manually. Let us cut 15 % of the most expensive scenarios.

So we still have coverage of 85 percent, it is high coverage. But the cost for 85 % of scenarios is not 85 units but 50.

So, it is good to find the point where we need to stop automating. So, that is another metric and it’s called coverage.

{Slide 30-33}

Now let us take a look at the levels and where we can perform automation. So, every application can be divided in different levels.

So typically, these Unit level, Service level and User Interface (UI) level. On the Unit level we have automated unit tests.

They usually written by developers and automation engineers basically do not need to write them and they are extremely fast. The next level is the Service level.

So, the service tests also called API tests and they have middle speed. They still fast, but not so fast as the unit tests,

for example 100 of service API level tests can take several minutes. And at the top of the pyramid we can see the User Interface scenarios.

They are the most brittle, they are the most slow. They have slow speed and for example, 10 test cases from UI level can take several dozens of minutes.

And this pyramid shows that we do not need to automate every scenario at the UI level, because it is cheap and the scenarios are still brittle.

So, the test count in this pyramid should be decreased when we shift from the bottom to top.

{Slide 34}

And finally, let us take a look at the tools used in automation. The same pyramid for the different level will have different technologies and instruments.

For the unit tests, we usually use such frameworks as JUnit or TestNG for Java language and NUnit for .NET framework.

For service level we usually use SoapUI and Spring WS. And for user interface scenarios we use different tools depending on the technology stack.

For example, if it is web tests for automating some kind of web application we use WebDriver basically. For mobile applications one of the best tools is Appium.

For desktop applications it varies from different tools depending on the targeted platform for example NetBeans forms or Lion X and so on.

{Slide 35-41}

Let us take a look at the evolution of the Test Automation Frameworks. At the beginning of the automation there was Record & Playback scenarios.

So, we have just recorded some activity and play it back when the release came.

So, their scenarios almost unmaintainable so it is really difficult and it is really expensive to maintain them. Next step was Reusable Scripts.

So scripts can be parameterized and they can be combined in different packs. The next iteration was Keyword-Driven frameworks.

They can allow to distribute the automation activities between test automation engineers and non-technical persons.

So, every step in our scenario is divided into atomic business steps. They called keywords and we can combine keywords in new scenarios.

So, the new test cases can be created by a combination of existing keywords, so we do not need to implement new feature in code.

One of the evolution of Keyword-Driven was Data-Driven frameworks.

Data-driven means that we have one test case but different data sets, several data sets and by combination of data sets we can test another behavior of the application.

For example, if we have an insurance application we can feed the test case with different users, with different health and produce the different application behavior.

And one of the most popular trending nowadays methodology is Behavior-Driven development.

So, it can be applied into test automation and the point of Behavior-Driven development is to writing the behavior of the feature instead of tests.

So, the behavior usually described in English like Gherkin language which is easily readable by developers,

by test automation engineers and which is more important by business and customer.

So the customer or guys from business can write the behavior in Gherkin language and the test automation engineers can create scenarios which would able to read this language

and perform the steps from it.

And the final type of the framework is Hybrid frameworks. This is combination of different types mentioned below.

So for example, a Keyword-Driven framework with Data-Driven support is Hybrid framework.

So, basically in EPAM we use Hybrid Frameworks because we can pick up the most, the best features from every methodology below.

{Slide 42}

Okay. Basically that is it. Thank you for your attention and bye!

Hello, my name is Marharyta Halamuzdava and in this video we will look at the importance of non-functional requirements.

Slide 2:

Non-functional requirements (NFRs) describe how a system works, while functional requirements describe what a system should do.

The main NFRs are: Performance and Security – these will be overviewed within the course, Reliability, Maintainability, Usability, and Supportability;

there are many others, as well. These requirements are just as critical as the Functional requirements, as they ensure the usability and efficacy of the entire system.

Failing to meet any one of these can result in systems that do not meet internal business, user, or market needs,

or that do not meet mandatory requirements imposed by regulatory or standards agencies.

Slide 3:

So, what is performance?

Performance is that aspect of a system which indicates its ability to be as powerful, fast, stable, and scalable as required. Performance is first of all:

Response time – the amount of time a system takes to react to a user request. This is an indicator of how fast the system is.

Capacity – the highest level of load a system can take and handle without a significant increase in response time or a decrease in stability.

This is an indicator of how powerful the system is. Stability – a system property that indicates the system’s ability to maintain its level of operation;

and Scalability – a system property that indicates the system’s ability to be readily enlarged.

That means that the capacity of an ideally-scalable system should increase as fast as the power of the system resources increases.

Slide 4:

Performance testing is an iterative process, which typically starts with engagement planning that includes:

information collecting, requirement analysis, test plan preparation, creation of scenarios and workload, selecting approach and testing tools, and setting the SLA metrics.

The second step is Preparation. Here test scenarios and scripts are rolled out, the data required for the tests must be created, and testing environment must be prepared.

The third step is Test Execution. All the collected results go to analysis, where the collected metrics are correlated with the SLA and the issues found during testing analyzed.

The final stage is preparing a test result Report, Code Fixing, (and) Tuning and Hardware Reconfiguration.

The Fix and Tune stage is very specific and will not be discussed in this video. Now, let’s take a look at each step of the process.

Slide 5:

Engagement planning is the first stage and moreover is not a full part of the iterative cycle. The performance test plan is the main document in this stage;

it includes all the necessary information for scale testing and process description.

Slide 6:

The Engagement stage includes: Requirement Analysis – this means reviewing the application functionality under testing,

identifying performance requirements such as response time, normal and peak load, common traffic patterns, and expected or required uptime;

Scenario Creation – a combination that defines the number of virtual users of every virtual user types, start time, duration, and some other settings of the performance test;

Choosing a load-testing software tool (the choice depends on the customer’s preferences re using freeware or a commercial tool for testing);

Developing a Test Script – a special program code that simulates typical real user behavior.

At first glance everything is clear and simple, but things aren’t always what they seem. So let’s look deeper and prevent the most evident mistakes.

The following examples are just the most widespread and far from being the full list of missteps you may face.

Slide 7:

In order to assess the performance of a system it is not enough to state that “the system should be capable of supporting 80,000 customers” or

“the system should be able to support 3 pages/sec”, for example.

These statements are often good metrics at a high management level but do not define the work that the system must support.

Slide 8:

Functional testing and performance testing are fundamentally two different beasts.

Focusing on detailed single-user interaction with the system misses how a system operates for multiple concurrent users following valid business rules.

Crank up the load using functional tests as the model and, yes, the system load is significantly increased, but it's doing a lot more of exactly the wrong kind of functions.

Testing system performance under load requires specifically-designed performance test cases: re-using functional test cases and executing them under load is completely the wrong approach.

Slide 9:

Different tools are better for different purposes, they require specific knowledge and have their own advantages and disadvantages.

Selecting the right one can have a significant impact on the testing process.

Slide 10:

Record-and-run solution is not a suitable approach for load testing, as it can give false positive results in the event no assertions are involved,

thus the script may receive successful 200 http code responses with an invalid 500 http code stack inside.

The best practice is a script that is written manually with full parameterization of everything.

Parameterization is not an automated action, but the complex analytical process of studying the application’s behavior.

Slide 11:

The test preparation stage includes all the activities to get ready with the scenarios, scripts, load model and testing environment.

Slide 12:

In the test preparation stage we build a test environment and pre-generate necessary data.

Slide 13:

Extrapolating load test results is a very risky business. You should replicate every aspect of your production environment:

machine profiles, configuration, database, network architecture, load balancers, firewalls, and so on.

The best practice is to create complete images of the production machines to be duplicated in your test environment.

Slide 14:

It’s very likely that an application tested on lower database records might fail in the production environment.

Therefore, it’s one of the responsibilities of the performance engineers to make sure that the test environment system has an appropriate number of test records

to accurately test the system.

Slide 15:

Testing and collecting test results begins after the environment is ready and the testing scripts have been created.

Slide 16:

The testing stage involves test execution, monitoring, and result collection.

Slide 17:

Performance monitoring should be part of the infrastructure and take place at the time of setting up the environment.

Very often it is completely ignored and is taken into account just the results provided by the load testing tool.

System resource utilization should also be analyzed to track how busy various resources of a computer system are, when running a performance test.

Slide 18:

Load generators are the machines that run the scripts in your scenarios.

They are part of the infrastructure, so just like you collect performance measurement from the system under testing,

you should collect performance measurements from the load generator machines in order to determine their status and avoid bottlenecks in this part.

Slide 19:

GUI mode consumes a lot of memory and other resources, which in turn negatively impacts your scripts and tests.

Using the non-GUI mode of JMeter helps to reduce both resource requirements and potential errors.

Simple Data Writer logs the results in a flat file and does not display the results on JMeter's GUI mode.

Slide 20:

The stage of analysis begins when the stage of testing has been passed and all the results have been collected.

Slide 21:

All the results are to be analyzed properly. As an example, on this slide, if the monitored period includes a system warm-up,

the results could be a part of the accepted range by mistake. Worse still, the test would be considered as “successfully passed” and

the application version would be endorsed for production. This could have dire consequences.

Slide 22: After all the analysis is done the stages of reporting and discussing begin.

Slide 23: All your efforts regarding performance testing may be in vain if you do not know what exactly you are measuring.

Make sure you allow sufficient time for completing your analysis and for preparing a comprehensive report aligned to the performance targets for the engagement.

Slide 24: As you can see, performance testing has many pitfalls, so if you want to provide load testing on a high level, you should know exactly what to do.

If you think you need help on your project with performance-related questions, here at EPAM we have a special group of professionals in the area of performance testing and analysis.

You can visit the Knowledge Base page to get more materials on the subject as well as contacts for addressing your specific questions to.

And you can also send a question or thoughts to a distribution list of the performance community in our company – it’s called Project EPM-PERF.

The list collects professionals in performance testing and analysis across our entire company, so the likelihood is great that your request will be answered promptly.

Thank you for your attention!

Slide #1:

Hello,

Colleagues, today I’d like to tell you about Security Testing.

Before we go too deep, I’d like to make a small remark. During this presentation, we’ll be talking about Web applications. Only.

Network or Infrastructure security, mobile applications security are out of scope for this presentation.

Slide #2:

Application Security Testing is the generic term.

For the sake of simplicity we'll say that application security testing is the set of activities focused on the analysis and testing of applications for security vulnerabilities.

There are three main styles of Application Security Testing:

Static Application Security Testing, or SAST.

SAST is focused on the analysis of the application's source code and/or compiled versions of the code for security flaws.

The second type is Dynamic Application Security Testing, or DAST.

Testing is performed against a running application. The engineer who is doing the DAST performs attacks against the application and

analyzes the way in which the application responds to these attacks.

The main idea of Interactive Application Security Testing (IAST) is to provide a combination of SAST and DAST approaches,

getting the most out of the pros that each approach offers with fewer cons, and improving the overall quality of the testing results.

Slide #3:

SAST is focused on analysis of the application's source code. The primary purpose is to identify security flaws early.

The most popular are the automated and semi-automated SAST approaches.

In this case special tools to analyze source code and/or compiled versions of code to help find security flaws are used.

Tools can be integrated into the development environment, preventing developers from making security errors that could produce vulnerabilities.

What are the pros of the SAST tools?

• Scale well -- can be run on lots of software, and run repeatedly (as with nightly builds or continuous integration).

• Useful for things that such tools can automatically find with high confidence, such as buffer overflows, SQL Injection Flaws, and so forth.

• The output is good for developers -- highlights the precise source files, line numbers, and even subsections of lines that are affected.

But there are cons you should be aware of:

• High numbers of false positives.

• Many types of security vulnerabilities are very difficult to find automatically, such as authentication problems, access control issues, insecure use of cryptography, etc.

• Frequently can't find configuration issues, since they are not represented in the code.

• Difficult to 'prove' that an identified security issue is an actual vulnerability.

• Many of these tools have difficulty analyzing code that cannot be compiled.

Slide #4:

Now let’s talk about Dynamic Application Security Testing.

Here we test for various types of vulnerabilities against a running application. The engineer attacks the application, as if he was a real attacker,

from different angles and with different knowledge about the application.

I’d like to focus on two different types of DAST that are often confused - Vulnerability Assessment and Penetration Testing.

So, let’s start with the Goal of these two types.

Vulnerability Assessment aims to find as many security issues in the target application as possible but not to exploit them.

The identified issues are turned into a prioritized list with advice on how to remedy the discovered vulnerabilities.

The Penetration Test has a specific goal. During the Penetration Test the engineer can find one vulnerability and exploit it to obtain the goal.

In this case none of the other vulnerabilities may be displayed in the Penetration Test report.

A second big difference between the Penetration test and Vulnerability Assessment is that Vulnerability Assessment can be automated to some degree.

We’ll talk more about the tools later during this presentation, but there are many scanners that can be used for performing an automated scan of the application

that can be compared to Vulnerability Assessment.

The Penetration Test is hard to automate due to the fact that this is often a creative process that heavily depends on the experience of the penetration tester.

Slide #5:

There is one more, new approach - Interactive Application Security Testing, or IAST, that combines dynamic and static techniques.

The main idea of this approach is that a special piece of software – it’s usually called a software agent – is added to the application.

It allows one to collect data about different application and security events.

The pros of such an IAST approach are the following:

\* It enables one to reduce the number of false positives.

\* It allows one to catch security issues that the other approaches may miss, for example, identifying vulnerabilities that do not provide any response by providing the agent

with access to an intermediary service that monitors the application over time for vulnerabilities.

\* And it allows one to see what part of the code contains the security flaw.

At the same time there are some cons you should be aware of:

\* Currently not all programming languages are supported.

\* Depending on the solution, it may take a significant amount of time to configure the tool to work properly.

\* As with SAST, the cost of a license for the tool should be considered.

Slide #6:

Now it’s time to talk about the classification of risks.

The most popular is the classification from OWASP – the Open Web Application Security Project.

OWASP Top 10 is the prioritized list of the ten most critical web application security risks.

So what is in the list? First place – Injections. This is a generic term for different types of attacks, such as LDAP Injections, OS Command Injections and SQL or NoSQL Injections.

The main idea is that an attacker injects commands into forms or input fields and then these commands will be unintentionally executed by the application.

Second is Broken Authentication and Session Management.

Application functions related to authentication and session management are often not implemented correctly, allowing attackers to compromise passwords,

keys, or session tokens, or to exploit other implementation flaws to assume other users’ identities. The third risk is Cross-site scripting or XSS.

It allows an attacker to inject malicious client-side script (in most cases, it’s JavaScript) into a web application which is later executed by the victim.

In the fourth place, we have Insecure direct object reference. This is a flaw in the design of the web application where access to a sensitive object, such as,

for example, a directory or any file in the application is not fully protected and the object is exposed by the application.

In the fifth place we have Security Misconfiguration. To ensure the security of a web application it is important to secure the configuration of its components, web server,

secure the operating system and ensure that it is always updated with the latest security patches.

6th – Sensitive Data exposure.

Sensitive data stored in databases or any other object should be protected. Sensitive customer details, application details should not be disclosed.

7th is Missing function level access control.

If the web application fails to perform proper access control checks the attacker may craft requests and get access to functionality that is supposed to be not available for him.

8th - Cross-site Request Forgery.

A very serious and widespread vulnerability. This is an attack that forces an end user to execute unwanted actions on a web application in which they're currently authenticated.

9th - Components with known vulnerabilities.

Applications using components with known vulnerabilities may undermine application defenses and enable a range of possible attacks and impacts.

10th - Invalidated redirects and forwards.

By modifying untrusted URL input to a malicious site, an attacker may successfully launch a phishing scam and steal user credentials

Slide #7:

So now let’s briefly go through the methodologies that can be used to perform security testing.

1) OWASP Testing Guide

It can be treated as a good framework that you can implement in a project and as a guide that describes techniques for the testing of the most common web application

and web service security issues.

If you are looking for checklists that can be used, please refer to the OWASP Application Security Verification Standard.

2) PTES

A detailed description of how Penetration Testing should be performed. Phase by phase. Without going deep into details of how certain actions should be performed.

3) OSSTMM

A manual covering many aspect of security testing. Good for big applications and complex solutions but needs tuning to be applied on small projects.

4) ISSAF

This is also a framework that is mostly for organizations that plan to enhance the security of their information assets. Not for small projects, for sure.

5) NIST 800-115

This document provides guidelines for organizations on planning and conducting security testing. A good document that contains a lot of information.

Slide #8:

To address a typical project’s scope and goals during security testing you can use the following phased approach:

Preparation phase: discuss the objectives of the testing, clarify scope, testing restrictions, reporting format and contact information

Information gathering: during this phase we analyze different information sources (forums/blogs, search engines, newsgroups, articles, social networks, etc.)

performing open source intelligence (OSINT).

Vulnerability analysis and exploitation: during this phase we perform automatic scanning along with manual tool assisted security testing

to discover security vulnerabilities and to exploit them.

Reporting: during this phase we summarize and classify all disclosed vulnerabilities and provide a Security Test Report.

Present the report to stakeholders to make sure they have read it and understand what is written there.

Slide #9:

Now let’s talk about the tools that can be used to perform security testing. Tools cannot cover all aspects of the application. There is no silver bullet, no all-in-one tool.

The engineer MUST perform certain tests manually to cover areas where tools are inadequate.

Currently EPAM has licenses for the following commercial tools used for security assessments:

\* Acunetix Web Vulnerability scanner

\* Burp Suite Professional

In addition to commercial tools, other different open-source tools like w3af, skipfish, OWASP ZAP and others can be used.

Each tool has its pros and cons, and is good at finding certain specific vulnerabilities.

Scanners are good at finding the following flaws:

\* User-input validation on the client and server side;

\* Analysis of requests and responses for configuration issues in (like missing cookie flags);

\* Known vulnerabilities.

Scanners are not good at:

\* Password policy management issues and, in general, authentication checks;

\* Multi-step issues (such as injections);

\* Authorization issues;

\* Finding hidden files and folders;

\* Identifying session management issues;

\* Application logic issues.

Slide #10:

There is one thing you should always keep in mind - Security Testing activities are destructive by nature.

So, please, DO NOT PERFORM ANY ACTIVITIES BEFORE YOU HAVE PROPER AUTHORIZATION.

And please, make sure that you have discussed the following topics with Project Management before starting any testing:

1. What environment will you use?

2. When are you allowed to do your tests?

3. What actions are you allowed to perform against the target?

After you have provided the list of what you plan to do, MAKE SURE that everyone understands the risk. Remember to keep all approvals in written form.

When you do the testing keep the following in mind:

\* Make sure you understand what the tool or script does and what you expect to get as the result.

\* If you get results from automated scanners, please check them for false positive results.

\* Start creating the report from the very beginning of testing.

Slide #11:

If you have any questions about security testing you are always welcome to contact us. For this you can either send an email to our distribution list or find us in Yammer.

If you need more information, please review the KB about security testing and secure development.

Thanks for staying with me!

{Slide 1}

Hello, my name is Dzmitry Zmitrachonak and I am a Senior Solution Architect at EPAM Systems, and now we are going to speak about EPAM internal services in terms of CI/CD.

{Slide 2}

So, we will speak a little about “Services used in CI process”, “Services interaction schema” and what services EPAM has at this purpose.

{Slide 3}

Speaking about application development, we need to understand the application lifecycle. Each application that we develop goes through the same repeatable process with the same set of steps that we can call an iteration. And it all starts from collecting business requirements from your customer, then you write a code, you save it to the version control repository, you work with your EDEs, you can build directly this code in your local machine. But also at the same time it is built by automated service. And finally it is put to some binary storage or even deployed to the production or staging. And also you can use some Wiki pages to save the deployment steps and use some issue or ticket tracking system like Jira. And when you get the first release done all starts from beginning.

{Slide 4}

This application lifecycle. Schema can be easily transformed into the pipeline. And Continuous Integration and Delivery pipeline usually mean building code from Version Control Systems, testing code with unit tests, Sonar executed tests, publishing some artifacts to the binary storage. And delivery steps it is like deployment, QA testing, some other automated testing like User Acceptance testing or regression tests and getting the release or release candidate. The development team gets feedback from each step and if something fails, they fix it and the pipeline runs again. At the bottom of the slide you can see the EPAM services that implement those steps and we will speak about the services a bit more.

{Slide 5}

Every service in the pipeline tightly integrated with other services. And you can see the Services Interaction on this slide. For example, the Jenkins Service works with Version Control Service to get the source code, with Artifactory service to save the binaries, with Sonar to make a static code analysis and with EPAM Private Cloud to deploy entities. The developers work with Sonar services to get the reports, build reports or static code analysis reports, with Jira for task or issue management and with of course Version Control System to save the code into.

{Slide 6}

So, let us speak about each service itself and it all starts from Version Control System – the place where you bring your source code. EPAM has 2 version control systems. It is GIT (distributed VCS) and Subversion.

Git consists of 3 servers across worldwide. And homepage looks like this. I can show this. You can login here using your SSO credentials and create up to 3 personal repositories, for example, for developing some game with your colleagues or experiments, but also you can see the projects you are working with.

And another VCS is Subversion. It is implemented with 5 servers across worldwide: Western Europe (Budapest), Eastern Europe (Minsk), Russia (Saratov), Asia (Karaganda), North America. The Subversion is simpler to use, it is centralized version control system but still it allows you to keep your code. I can show this. And in addition to just storing the source code we have the Trac Wiki added to each subversion repositories, we can save the deployment steps or some other useful information here or just easily browse here visioning history more easily than just seeing the code. And one thing that I should mention here that please do not save your binaries to the source code repository. It is not allowed and it is strictly prohibited.

{Slide 7}

For this purpose you might use next EPAM service called Artifactory. It is our binary storage. Originally, it was a public Maven repositories’ mirror inside EPAM if you know what I amm talking about. But nowadays every project can order personal space to save any binary entities you want. It is a jar file, war file, zip file or any other kind of archive you would like to version. And also some dependency for your projects can be downloaded from this Artifactory service.

{Slide 8}

And as soon as you get your source code and binary entities saved into those 2 services that we are talking about there is a continuous integration service called Jenkins. Jenkins.epam.com is our centralized build system. This tool is really nice and powerful. It has many plugins and can implement any build steps or deployment steps that you can imagine. It has EPAM SSO integration. It supports many build tools. And each project also can order personal space here and you can see this space if you are the member of this project. And I can show you this, how it looks like. Here is the main page of this service. You can see how many jobs we have here. And it is pretty much easy to use and helpful for development. The best practice for using this service is to write universal build scripts and save them into the source code repository and of course be ready for release in any time. So, when you run your scripts on this service you will directly get the release done.

{Slide 9}

And as soon as your code is built you of course need to verify it with other tools like static code analysis. And one of those popular tools that we provide, the one of the services is Sonar. Sonar.epam.com is tests runner and reports aggregator for static code analysis. It runs many different tools like Checkstyle or PMD or FindBugs and saves code metrics here. It shows you code issues that you need to fix and it is integrated with Jenkins. So, if you get to this service at your browser you will see also the projects registered here and you can browse issues and every new project can get the registration on this portal by a quest.

{Slide 10}

In addition to static code analysis we also provide the code review service. It is nice tool, where you can discuss the code that you have composed, it supports threaded comments, inline discussions, it supports distributed teams. You can show your code to key developer, to development key manager to share your best practice, to review the code coverage and this tool not so automated, but still it is really useful during the code development. And of course each project, each new project can get access to it. It looks pretty much simple. You see the source code, you see the discussions, you can create new code reviews and make a final decision is your new piece of code good or not, should be integrated into the main line or not. So please do not hesitate using it.

{Slide 11}

And one more service I need to mention is a BlackDuck. It is our Open Source code analyzer. When you write your code of course sometimes, you search for the best practices on the internet and sometimes you copy-paste those pieces of code into the code that we write for our customers and this is not good. So, we need to somehow to find out does this project use some Open Source code injections or not to prevent EPAM possible losses. And we do this scan automatically on regular basis like month or two or by request. We might find all license conflicts, describe it and also let you know the ways of how fix these conflicts. We have a special team that might help you with some recommendations.

{Slide 12}

And not only license conflicts might be interesting for the developers. We also have a service for identifying, tracking and repairing the security issues. We have bought the tool called CHECKMARX. It allows you to find the technical, logical flaw issues, compliance issues, business logic issues and also get the recommendation how to fix it. This tool scans un-compiled code and builds like logic structure of how the data flows works inside. For now this service is not public and we provide it only by demand but I think it is only matter of time when we will start use it on a regular basis.

{Slide 13}

More information about EPAM services for CI/CD can be found out on the Knowledge Base or you can get any help about these services at EPAM Support Portal. If you go to this portal and go to the catalog of requests, browse to the Technical Services, you can find out the list of technical services, support requests. So if you have any questions please go there, write your question and our special team that is responsible for this exact service you want to have or use will provide with any recommendations or consulting in it.

{Slide 14}

So, that is it about EPAM internal resources for CI/CD. It was Dzmitry Zmitrachonak. Thank you for being with me this time and bye!

{Slide 1}

Hello, my name is Dzmitry Zmitrachonak and I am working as Senior Solution Architect at EPAM Systems, and now we are going to speak about Continuous Integration.

{Slide 2}

About Continuous Integration prototype, about services used for Continuous Integration at EPAM and about Continuous Integration tools.

{Slide 3}

So what is Continuous Integration? Let us imagine that your work as developer on some project. You write some new piece of code that need to be integrated into the main line. We need to make sure that this piece of code is good and it does not affect any other work. So, for this purpose we build special process that automatically builds, tests your code and notifies if something is wrong. So, Continuous Integration is a development practice that requires developers to integrate code into a shared repository several times per day. And each code check-in is automatically verified, build, allowing teams to detect issues early. And the main CI principles are: - to create process this way so the code is verified as often as it can be, and - every change must be integrated, and - the automatic, static and integration testing must be included. So, this allows us to catch defects as early as possible.

{Slide 4}

And how we can build this CI process? Let us prototype it. It is all start from getting in touch with development team. You need to get the information about what they already have if they have, who is responsible for this, and discuss the CI needs with key persons, key developers, with project management. As soon as you get this knowledge, please choose your toolset. And if the customer or team does not have any preferences, follow the way of the simplest and most useful tools that easy to use and easy to setup but also cover all these needs. One more important thing, let us start with building automated build. Then you can add other pieces like automated testing and do not forget to keep your scripts in repository and make them unified and create test pieces for each part that needs to be tested. You can also disable some modifications. For example, if you test your new additions to the CI process on live system. And you need to be close with development team to discuss the CI issues, automation questions and usually it is enough to get in touch with team once per week but feel free to do this more often.

{Slide 5}

So, when you finally get your Continuous Integration process, you can see it, something like pipeline like on this slide. And this pipeline will consist from the next phases: building the code, testing and publishing. And each of these pieces is implemented in EPAM by additional, by separate service.

{Slide 6}

We can see these services interaction schema and the pieces that I am talking about. This is Jenkins - the CI assistant for building, Version Control Service for storing the code, Sonar for code verification and static code analysis and Artifactory for saving the binary entities or downloading the dependencies. So, this is the CI system based on EPAM internal services.

?

{Slide 7}

So what tools are usually used for Continuous Integration? One of our favorite is Jenkins. Jenkins is one of the most powerful tools nowadays. It is Open Source and it is really rapidly developed, the new versions come out like even 4 times per month. Long-term support releases come out one time per month but still very often comparing to other competitors. And, this tool has a lot of plugins, more than 1000. It has a huge community. And it is flexible, you can build as complex solution as it possible. So let me show you how Jenkins looks in EPAM. So, it is the main page of our Jenkins instance, where all the projects can have their own tab for storing jobs or processes that they run. And you can login here using your Active Directory crFntials and see jobs for your project.

{Slide 8}

And another tool for Continuous Integration, Jenkins competitor is Atlassian Bamboo. It is not so free. It is free only for open source projects but paid for others. But it has a really excellent integration with Atlassian products. So for example, if you have Atlassian Crucible or Atlassian Stash or any other Atlassian products, Bamboo is one of these exits you might consider for a Continuous Integration process because it is really nice from the position of integration. And it looks like this, I can show you the example. Pretty much similar, we can see jobs, the last build that was run, how long it takes and some details the same as on Jenkins pages.

{Slide 9}

Another tool similar to previous two is TeamCity. It comes from JetBrains, the company that brings us the popular Java IDE, code IntellJIdea and it is also free only for some small projects, but has good dependency management and really nice professional support for paid version. So if you are looking for some good support please consider this tool. And this tool looks like this, also almost the same job list, some reports, some status trans. Pretty much similar but it is different.

{Slide 10}

And in addition to in-house CI tools you can use Travis CI. It is service that is very similar to GitHub. It is hosted on the internet, it is tightly integrated with GitHub. And many open source projects use it for automated building of the code that is stored on GitHub. And it has multi language support and it is tightly integrated with the most popular development services and tools. So you can use this for some open source project or by the prescription if customer pay for example. And this tool also looks really pretty much similar, you can see the dashboard, you can the projects that you are building. For example, some Jenkins plugins use this tool and you can see their build status and even configuration.

{Slide 11}

Another interesting tool that you might consider is GO Continuous Delivery. This tool is specifically based on modeling pipelines. So, if you build some complex project where some processes go in parallel this tool can be really helpful and it allows you to eliminate bottlenecks. But it is not free. It is a little example of how complex processes this tool can cover, so please go to this tool homepage, download examples and see. So this was about the continuous integration main tools, it is scheduler that run, build and deploy steps. But under the hood of those services there are some other tools that we look a bit a later.

?

{Slide 12}

This slide shows you the comparison of those schedulers. And as you can see the Jenkins is the most popular tool. But if you build scripts and write them correctly, independently and universally, without hardcode, without any special things that are relevant to only some selected CI tool, it does not matter which tool you use, Jenkins or TeamCity, you will be able to easily switch from one tool to another without issues.

{Slide 13}

Under the hood of Continuous Integration main tools there are set of automated build tools. Those tools allow you to describe and run build steps, test steps and deployment steps. And why we should use it? Because sometimes simple BAT files or common LAN files are not enough. So, those tools have a lot of integration with compilers, linkers and many other basic development tools that you use for code development compilation and so on and so forth. So, what are those tools? It is a Maven, Gradle, Ant or Grunt. And I can show you little examples of those tools, scripts. And for example the Apache Ant with Ivy dependencies management, script looks like this. This XML file with description of code dependencies you need for code building. And this is a build.xml file that describes build steps. Pretty much the same description from Maven perspective looks like a bit different XML file. Those tools are pretty much similar, but Ant is more flexible, Maven is more strictly, has more strict descriptions or definitions and more complicated from some point of view but still covers all your needs. And Gradle is more Java like. You can write the scripts very similar to Java files. But still those tools usually used in addition to CI tools to define the build, deploy and testing steps. And when you write those scripts, you save it to the code repository, you run it from Jenkins or TeamCity or any other CI tool.

{Slide 14}

And in addition to automated build tools there are static code analysis tools that are used from Sonar, another EPAM service. And those tools usually help to standardize the code, to catch bugs in early stage, to get code metric, to verify how your code looks like or what static issues it can have. And I can also show you an example. This is one of popular EPAM resources: source code, EPAM Heroes Suit. You can that they have some issues here, but it is not so critical. But still we see that some things need to be verified, checked and fixed.

{Slide 15}

So, that is it about Continuous Integration and primary tools used for this purpose. It was Dzmitry Zmitrachonak, thank you and have a nice day or night! Bye!

{Slide 1}

Hello, my name is Dzmitry Zmitrachonak and I am working as Senior Solution Architect at EPAM Systems. Now we are going to speak about Continuous Delivery.

{Slide 2}

We will speak about “What is Continuous Delivery (CD)”. We will speak about “CD prototype”, about “Continuous Delivery integration with EPAM internal services” and

about “Continuous Delivery tools”.

{Slide 3}

So what is Continuous Delivery itself? As we have already discussed, the Continuous Integration is a development practice that allows developers

to integrate the new pieces of code several times per day and every check-in is automatically verified and this allows you to detect the issues earlier.

And Continuous Delivery itself is a software engineering approach in which teams keep producing valuable software in short cycles and ensure

that this software can be reliably released at any time. So, basically the Continuous Delivery is Continuous Integration plus automated testing and automated deployment.

And Continuous Delivery usually follows the same principles as Continuous Integration. You create the development process,

every change must be integrated and you need to be ready for release in anytime;

the automatic, static and integration tests must be included and you need to catch defects as early as possible.

{Slide 4}

CI/CD process can be prototyped in the same way. So, first you need to get in touch with the team, you need to get to know what they already have and who is responsible for this.

You need to get in touch with key persons to discuss their needs. As soon as you get this information collected, you need to define the best tool set that covers those requirements.

And if customer do not have any preferences follow the way of this simple solutions, but powerful at the same time and useful and easy setup and easy implemented.

First try to create the automated build, then add automated deployment, automation testing and keep your scripts as simple as possible.

Keep it in the source code repository and create test and deployment processes for each piece that need to be tested and deployed. But, do not make it complicated.

And at the same time you can even test some pieces on life process, just to know the notification that make sure that the new additions do not break anything.

Also, do not forget to be close to the project team. You need to discuss the automation or integration issues with them at least once a week,

but you can also get in touch more often if you need.

{Slide 5}

So, the CI and CD process finally will look like the pipeline similar to this slide. And again Continuous Integration is build step, testing step,

then you publish something to Artifactory. The deployment itself, the Continuous Delivery itself is a deployment and QA testing. And finally, you get the release.

Each of those steps in the pipeline is implemented by a different EPAM service that we will speak a bit later. And development team gets feedback if any step fails,

they fix the issues and the pipeline runs again.

{Slide 6}

Every service that is used for Continuous Integration/Continuous Delivery usually works with other services. And CD system based on EPAM internal services usually looks like this.

We have Jenkins for building and running other automated scripts. We have Version Control Systems that is used by the CI/CD system to download the code.

We store the binary entities to Artifactory. We deploy ready builds to EPAM Private Cloud, we save the test reports Sonar, we use BlackDuck Open Source code violations running and so on.

{Slide 7}

Let us speak about Continuous Delivery tools. And one of the most popular tools nowadays is Docker.

Docker implements nice idea of packaging the application with all dependencies into one “container”.

So, you do not need to take care about dependencies when you deliver this application to a customer. You just bring him the “container” and the customer runs it.

You also can use several “containers” to build more complex application.

Anyway this idea perfectly implements the idea of platform as a service which is a category of cloud computing services that provides a platform allowing customers to develop,

run and manage Web applications without the complexity of dependency management and building and maintaining infrastructure of course.

I can show you a little slide, a little page about Docker itself under the hood. So “What is the difference between Docker and virtual machine?” you might say.

As you can see on this picture virtual machines do not share the same operating system, they have their own system, they have their own binaries and libraries.

So, it is like a machine inside machine while the Docker itself is a separated environment. They still use the same operating system. But they are separated, those containers.

And it is easy to run, easy to manage, it is less strict, tighten to system resources, more flexible. So, that is Docker itself.

{Slide 8}

All right, so, what else we can use for Continuous Delivery? Another thing that you should take is configuration management.

And one of the interesting tools for this purpose is Ansible. Ansible is really nice and simple tool.

It works by SSH, you can define the configuration as some script, as a code and infrastructure will be built from this code.

Ansible can manage machines in parallel and it is really nice for Unix-based or Linux-based operating system. So it is not cross-platform but still unique.

And I can show you an example of the code written for this tool. Here we define the service groups. We define the Playbook. For example, we want to install Jetty server.

As you see that we define “What to do?”, “How to do?” and “What we need to start?”

And when you run this Playbook or like receipt of building something you see the output of this script, what it does,

you see that those steps will be running and the process is so running. So, really powerful nice tool.

{Slide 9}

And in addition to Ansible some might consider another tool called Chef. Chef is more powerful, more flexible and cross platform tool.

It is client-server oriented. It has Web management console. It also follows the same idea. You write the definition of the infrastructure,

the tools you need to have and you run those Playbooks. But they here cook, it Cookbooks like when you are on the kitchen and there tools also called the same like “knife”

and it is really nice from terms perspective. But it is still the same, it is configuration management tool.

And I just can show you the Home Page. When you install the Chef server you can define the nodes, seats, save them to the server.

See that I do not have the Cookbooks, but still you can upload it here.

{Slide 10}

And speaking about Chef we also might consider its competitor. So Chief competitor is Puppet. It is another configuration management tool.

It also follows the client-server architecture, have Web management console and cross platform. What tool use, it depends on your needs. For example,

I like Ansible more because it is more simple, but if you need something more powerful you might consider the Chef or Puppet.

{Slide 11}

And if you compare those tools they are very close by functionality and usage frequency. Those tools help having minimal differences between environments.

You just define the configuration and you can get pretty much the same configuration in minutes. So, this makes environments easy to export, to build and use.

{Slide 12}

If you want to know more about Continuous Delivery you can get the information either at our knowledge base pages, either on website of “Thoughtworks”,

they have nice articles about Continuous Delivery or also, we recommend a nice book from Jez Humble and David Farley, also called “Continues Delivery”.

So, you can find it, buy it and read it.

{Slide 13}

So that is it about the Continuous Delivery and EPAM internal services and thank you! It was Dzmitry Zmitrachonak. Have a nice day!

{Slide 1}

Hello!

My name is Anton Komarovsky, and here in this video we are going to talk about yet another project role: the business analyst.

{Slide 2}

You will have a closer look to business analyst as a role: who are they?

You will really know why their role is important and what they do.

Also we are going to briefly discover how they work and which results they deliver.

{Slide 3}

First, let’s define who is a business analyst.

According to the definition from Business Analysis Body of Knowledge®, a business analyst is any person who performs business analysis tasks described in that Guide, no matter their job title or organizational role.

I’d like to underline, that some projects may not have a separate person with business analyst role, but it doesn’t mean that business analysis is not performed.

{Slide 4}

Business analyst is a liaison between team members, customer and management. A business analyst is able to talk customers language and transform customers need to the structured requirements.

We can call someone a skilled business analyst if they at least:

Understand domain area as good as customer or even better

Ías system thinking and able to work with large volumes of heterogeneous data

Able to use the tools and techniques of business analysis

Has a strong communication skills

The last skill is underlined because you might be a real expert, but if you can’t communicate with people – business analysis is not your way.

{Slide 5}

So, why a business analyst is an important role. Let’s see what we are doing:

At the very beginning, a business analyst identifies customer problems and needs.

Together with the team they model and offer the solution to meet business requirements.

A business analyst breaks down the whole functionality into small stories or tasks.

A business analyst supports the team with clear requirements, answers questions during the implementation.

Also they help with scope management and build up right customers’ expectations

{Slide 6}

Actually what a business analyst does, depends on the project phase.

During INITIATION PHASE analyst studies a new domain area, provides strategy analysis, understands the problem “in general”. As a result, vision and scope document, business requirements document is delivered.

During PLANNING PHASE while collaboration with stakeholders is planned, requirements are being elicited, modeled and specified. Finishing planning stage analyst had project backlog and specification.

During EXECUTING PHASE the analyst is responsible for requirements management, clarification of details, solution evolution, scope management. User stories, presentations, change requests appear at Execution stage.

While CONTROLLING PHASE the analyst takes part at acceptance testing, provides user trainings, demos, delivers manuals and user guides.

During CLOSING PHASE they prepare a final documentation package.

{Slide 7}

During our work, we use a variety of tools and techniques.

We use Skype and Outlook to communicate with the client and team members, cloud storage for sharing files, Microsoft Office to create documents and presentations, Jira and Confluence to manage requirements and tasks, various applications for BMNN and UML modeling, special software for creating prototypes.

Also different techniques are used within daily project activities, such as interview, brainstorming, business process analysis, functional decomposition, mind mapping and many others.

{Slide 8}

A business analyst faces large amount of information every day. Information flows run from customers , users, the team – any stakeholder.

Good business analyst should not flush this unprocessed data on the team, because it will distract developers, QA engineers from their tasks, and reduce team capacity. Distributed information often leads to miscommunication.

Working with stakeholders’ needs, requirements, analyzing customer problems, studying the domain area,

a Business Analyst makes information structured and classified. They create Vision and scope documents, the product specifications, product backlog etc. A business analyst constantly makes an order from chaos.

{Slide 9}

Business analysts prevent their teams from doing wasteful work, making sure they do not devote resources to tasks whose value is low or absent.

Business analysts simplify team’s work, allowing them not to be buried under tons of customer requests and requirements.

On the other hand, a business analyst helps the customer in successful communication with the whole team.

This makes it possible to achieve valuable results using less effort, which ultimately increases team velocity.

{Slide 10}

That’s it, thank you for watching! See you next time. Bye!