opensphere

TESTING SYSTEMS INSIDE OUT

Index

OPENSHPERE - Developing large projects in distributed environments is never a simple task. Being dependent from other teams makes it hard or sometimes even impossible to develop and test parts of the project under one's responsibility. Opensphere can simulate system components which aren't available yet, allowing to progress with development on schedule and independently from other teams. The built-in testing framework enables executing regular regression test runs making sure the product is thoroughly tested before the delivery.

opensphere

Chapter 1.
Introduction

Most people aren't aware of how much their lives rely on various IT systems. They usually realize it the hard way, when they try to access their bank account only to find out that due to a system failure, online access isn't possible at the moment. Or when they go to the administrative court to obtain a copy of the land registry of a property they own and are told that the system doesn't work and no one can tell when it will become operational again.

System failures are usually a result of either a human error or system malfunction. The first one is something that cannot be eliminated as we are only humans and it is human to make mistakes. System errors on the other hand can be minimized by implementing fault tolerance procedures and thoroughly testing systems before delivery. This is especially important in real-time mission-critical systems in which case any potential error may have fatal consequences.

It's not that every system malfunction causes serious consequences. The impact of a failure depends on its severity. Sometimes a system owner gets away with a minor malfunction when it isn't serious and it's taken care of right away before anyone really notices it. But in some cases when the system failure lasts longer and concerns a large number of people, the system owner loses business. And it isn't only the system owner who loses business but also the company responsible for developing, pre-integrating and delivering it as its reputation is impaired.

Even though not every system requires 99.9999% availability it still has to be tested to some extent to eliminate major software bugs and errors. It doesn't matter whether the project is worth millions or thousands; software quality should be a real concern as it costs 10 times less to fix an error detected during the development phase rather than after delivering it to the client.

opensphere

Chapter 2.
System testing 101

The importance of system testing cannot be questioned as it's the only way to ensure that the system behaves according to designer's and users' expectations. It may seem that testing is the very last stage of system development but this is far from the truth as it is present from the very beginning of the development process.

Testing is present before the development even starts - in the planning phase, when the requirements are taking their final shapes. Each implemented functionality requires a set of test cases which should cover all possible use case scenarios. These should be included in the requirements definition.

Actual testing starts from the very first piece of code. This is called unit testing and it ensures that the results of the code execution are exactly as expected. Unit testing allows early detection of errors which is quite important since the earlier an error is detected the less effort and attention is required to correct it.

Whenever there is a piece of software ready which is somewhat usable, the functional testing starts. While the unit testing is usually performed by the software developers, functional testing is carried out by dedicated software testers. Functional tests cover previously prepared use case scenarios.

Functional testing often covers only a given part of the software's functionality which has been implemented, thus it doesn't ensure that the previously implemented features still work flawlessly. This can be ensured by automated regression tests.

Components are tested separately but there are certain test scenarios which require interaction with other system components. While there are ways to simulate another object in functional testing, the real tests are run after system integration.

Functional tests are executed on rather limited data sets.What also has to be tested is system's behavior upon exceeding certain volumes of data. This is called volume testing and it ensures that the system remains operational even with large amounts of data. Volume testing can be paired with scalability testing which is about making sure that expanding the system will allow to effectively increase processing power. Both of these are considered non-functional tests.

opensphere

The last stages of system development are the acceptance tests. These are suites of test case scenarios agreed with the customer which prove that the system was implemented according to the requirements.

Drafting and developing test case scenarios, planning testing campaigns and supervising test suite runs are very responsible tasks. There is no piece of software which could do it for you, but centeractive's Opensphere will support in the whole process from the very beginning.

Chapter 3.
Software testing

Every piece of software or hardware comes with support which is included in the final cost of the product. Ensuring high quality during development will be of benefit in the future with fewer bug reports from the client. Bugged products will require a lot of attention from the technical support staff and probably software developers as well. Sooner or later it may turn out that the profits are long gone and you start losing money on given contract.

That's why software should be tested and it should be tested as early as possible. Unit testing allows every line of code to be covered, to make sure that the results of code execution match expected values. Code coverage is actually one of the software quality metrics and that's why today's continuous integration environments facilitate means for static code analysis.

Proper continuous integration environment not only compiles and builds the application, but also executes unit tests making sure that changes in the software did not impact the state of the application. Unfortunately unit testing isn't enough to ensure high quality. It's the functional tests that play the key part in quality assurance.

opensphere

While unit testing is about static machine thinking, functional tests pertain to user experience. 100% code coverage with unit tests won't guarantee that the application is error free. This can be ensured only by covering all (or at least as much possible) use case scenarios in functional tests. Functional tests should be planned before the implementation starts and the test case scenarios should cover every possible option. These tests are usually executed manually by system testers, but with proper framework they can be automated and integrated in continuous integration environment. Executed with each build they are the best way to make sure that the application works correctly. Even though they are a part of software integration, they can be considered regression tests as they make sure that the changes in code do not impact previously implemented features.

Opensphere testing framework supports development process from the start, when the requirements are discussed and the test case scenarios are prepared. People involved in the planning can use Opensphere to create test case descriptions which are later taken on by system verification specialists who implement these test cases.

Chapter 4.
Smoke testing

Unit and functional testing discussed in previous chapters concern single components. System components can work flawlessly in testing environments but it's the smoke tests run after system integration which make sure that the system behaves according to design and can be subject to further testing before being delivered to the customer.

Smoke tests are very first on the agenda after integrating the system. They are a subset of test cases covering system's most important functionality. By this definition unit testing and functional testing can fall under that category. This way the integration specialists can make sure that the integration was somewhat successful and can proceed with more specific testing, including volume, stress and scalability testing discussed in the next chapter.

centeractive's Opensphere can be helpful right before the integration smoke tests as it can simulate system components, giving us an idea how the system will behave and what kind of problems can we expect after the integration.

Chapter 5.
Volume, stress and scalability testing

Every system is designed to handle a certain amount of concurrent requests and store a certain volume of data. Regardless the initial design and final implementation, it's certain that sooner or later the system capacity will be reached. That's why before a system is delivered, it has to be tested how it will behave when its capacity is reached. This is called volume testing and it's about putting the maximum amount of data that the system can store and running the entire test suite. Under such circumstances the system can lower its response times, deny accepting new requests or even crash. Recovering from a crash can take many hours and that's why quality assurance requires running volume and stress tests.

Properly designed system should be able to grow as the business of its owner grows. Scalability tests can show how the system's capacity increases with additional processing hardware and storage devices. These tests should be paired with volume testing when the system is already running at its maximum capacity. Customers may require a detailed analysis what kind of scalability the system offers so they can include necessary investment in their CAPEX and OPEX planning.

Chapter 6.
Conclusions

Quality assurance in software business can either make or break, no matter whether we're talking about real-time mission critical systems or less demanding solutions. The only way to ensure high quality is proper testing. Such testing should be well planned, precisely implemented and thoroughly executed.

Opensphere assists in that process from the beginning allowing to create test cases descriptions, grouped in test suites and implementing them as the implementation of the functionality moves forward. Opensphere's ability to simulate system components which aren't available yet also proves to be very useful on many occasions.

opensphere