opensphere

Software development in distributed environments

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.
Software development model

What can you say about a completely non-IT related company which hires two or three software developers to create a small piece of software for their very own use? And after that they keep coming up with ridiculous requirements just to keep themselves busy? To be politically correct, and not to use strong language, you could call that company mismanaged. Fortunately days when such companies existed are long gone, as managers' positions these days are occupied by more qualified persons who outsource IT projects to companies specializing in software development.

Software development has never been simple. At the very beginning it was the technology that made coding difficult. As time went by technology improved greatly, however software development didn't get any easier. It still requires a great deal of programming skills, knowledge of specific language patterns and much experience to produce quality software.

Completing a given project in a limited time requires a certain amount of manpower. The quantity of these resources is determined by the size of the project and agreed deadlines. Larger projects require tens or even hundreds of specialists split into teams which are given different responsibilities. Very often tasks of developing certain components are outsourced to external companies which have the expertise in given areas.

It's obvious that working on a large project in distributed environments is considerably different from developing a simple application by a group of three.

opensphere

Chapter 2.
Challenges

Developing different system components in parallel allows the completion of the entire project within the time schedule, which would have been unrealistic for a single development team implementing one system component after another. This isn't the only case where the product is developed in a distributed environment. Given the variety of programing languages and available technologies, it might be that the company doesn't have enough expertise to develop the entire system and has to outsource development of some subsystems to another company.

We've already established that a system comprises a number of components which communicate with each other. Each component requires some input data which is send back after being processed. This data exchange plays a bigger or smaller role depending on the particular case. Now the question is: how to design, develop and test a component which has to interact with other components that aren't available yet?

Availability of components doesn't always help either as they not only have to be available but accessible as well. And one doesn't really imply the other. Some companies have their production systems running 24/7 and they cannot afford to grant access to that system to a contractor company which has to develop and integrate another component. It's very rare that a company has a testbed system which they can share with a third party contractor. Making more or less an exact copy of the production system only for development purposes would then be out of the question as it would be economically unreasonable.

Either way unavailability of other crucial system components poses a problem more often than you'd think.

opensphere

Chapter 3.
Remedy

How to develop and more importantly test software components when the system components aren't ready yet or the production system isn't accessible.

Every system should be well documented, from requirements specification to user manual and interface documentation. The last one is especially important as it contains detailed information on how the component interacts with the rest of the system. These details include communication protocol specifics as well as messages which are exchanged between system components. With such knowledge we can try and simulate given components in our development environment. Implementing objects just to simulate a certain behavior can be time consuming, but there are frameworks which allow simulation of any component with just a few clicks.

One of solutions which implements such features is centeractive's Opensphere. It can simulate various objects allowing independent progress with implementation and verification of contracted system components. The specifics of the simulated application are determined by the protocol which is used by this application to communicate. As we are discussing IT industry solutions, supported protocols include JMS, SOAP over HTTP, TIBCO Rendezvous® and JDBC. Pretty much the only input data which is required to simulate an object is a file containing supported messages in a given communication protocol. Opensphere provides additional means to assist the user as much as possible. A built-in message detector allows the identification of certain messages which can later be edited in the internal messages editor and used as a source message template in a simulated component.

Implementation isn't complete before the application has been thoroughly tested. While on the unit testing level developers can use mock objects to simulate other components, it's not that simple in the case of system verification. Verification related tasks start in parallel with implementation. These include preparing test case scenarios, planning out test suites and regression test runs.

Opensphere can help with that as it contains a complete testing framework. It enables you to create and easily maintain test cases grouped in test suites, supervise test execution, and review and publish the test run results. All that wrapped up in a very intuitive and convenient user interface.

opensphere

Chapter 4.
Conclusions

In the IT industry time matters and it matters a lot. Depending on the contractual agreement not meeting deadlines can have pretty bad consequences. Trying to make excuses about the unavailability of certain components can be sometimes taken as unprofessional or even worse.

Fortunately there is Opensphere which can simulate non-existing components. As it supports major industry standard communication protocol, it provides a perfect remedy to problems that are very common in software development industry.

opensphere