ATDD – An applied approach to continuous delivery

Practice Head:
Janaki Jayachandran
Independent Testing Services

Authors:
Vasu Swaminathan
Anil Sannareddy &
Jayanth Krishnan

Acceptance Test-Driven Development (ATDD) is a refined Test-Driven Development (TDD) methodology that helps in expediting the development phase and reduces the risks of unmet expectations or miscommunication.

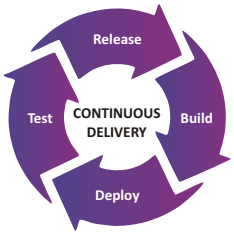
ATDD is a development methodology based on communication between the key stakeholders – business, customers, developers and testers. Each of these stakeholders comes with a completely different background, resulting in potential communication gaps.

In this practice, the entire team comes together on a common platform to discuss acceptance criteria elaborately. This is then distilled into concrete acceptance tests. It happens before beginning the development phase.

ATDD is one of the best ways to ensure that all stakeholders have the same shared understanding of the requirements. It also allows the development team to go into the project with a clear picture of the end user's needs, rather than on the interpretation of the individuals in between.

ATDD methodology is commonly used in agile teams. Agile testing methods embrace the principles of Agile software development, in which requirements and solutions evolve through collaboration between self-organizing, cross-functional teams.

ATDD – An applied approach to continuous delivery



WHY CONTINUOUS DELIVERY AND THE KEY CHALLENGE

In the competitive digital transformation era, for the business to stay it is evitable to have portions of workable pieces of software delivered in short cycles than wait for months or years together to realize a complete digitization initiate.

Continuous delivery is based on principle – **create a straightforward, repeatable and quick deployment process**. It is can be achieved through automating every process in course from development to deployment in production. That is, building a deployment pipeline.

However, this is a paradigm shift for IT teams to delivery because

- It is very well known that duration is short and only an incremental portion is being delivered; but how to ensure incremental pieces of system or function still useable by the business meets the business requirements.
- How to ensure reliability of incremental system or functions delivered to business.

This has led to new models of development practices like TDD (Test Driven Development), EDD (Example Driven Development), ATDD, etc.

HOW ATDD MAKES CONTINUOUS DELIVERY EASY

Though continuous delivery has led to an evolution of various development models many of them fail in practicality. This is where ATDD makes it easier for teams in non-continuous delivery models or other continuous delivery model to make a natural progress using acceptance tests which is framed in collaboration of all engineering teams - business, developers and testers.

As ATDD is customer and business focused, the outcome of each phase is very easy to understand by all the team members involved say development, non-development, non-technical, business, etc.,

Generally, testing is seen as one of major bottlenecks delaying the delivery of the software. This can be addressed through continuous testing by automating the tests to be done on software. As developers start with the code and QA get ready with automation scripts, having Continuous Integration in place can make things move faster. With builds getting deployed on the test environment through CI at regular intervals, an automated smoke test triggered through CI can check the sanity of the build.

Similarly, execution of acceptance tests through CI will enable test case execution and can find bugs at the earliest. This process helps in reducing Cost of Quality. It also increases overall productivity and all this happens unattended.

ATDD AND TEST AUTOMATION

Acceptance tests help regression testing and iterative development. Automation testing has proved to be fruitful when it comes to regression testing. Based on the business need, if the team has to execute the acceptance tests repeatedly for long period of time, it is wise to automate. This will help utilize the full potential of acceptance testing.

The acceptance tests can be automated using BDD and ATDD tools to act as testing, communication and documentation tools. It can also be used to publish reports with test results that non-developers can understand easily.

For example, Acceptance tests can be expressed as concrete tests in Given-When-Then style. Here's a requirement: "As an administrator, I want users creating accounts to be required to choose secure passwords so that their accounts cannot be hacked through a password guessing program." An acceptance test for this requirement can be expressed as below:

These tests can be implemented using frameworks like:

Cucumber/JBehave/Specflow/Behat: BDD tool that supports "Given-When-Then" tests for programming languages like Ruby, Java, .Net and PHP.

Fitnesse: a table driven framework that uses a wiki for displaying and editing tests

Robot Framework: keyword-driven framework that support texts or tables

Concordian: Java-based framework for expressing expectations in prose

THESE ARE THE CHALLENGES...

The general principle of CI - not committing the code that can break the build - applies for automated acceptance tests too. When people fail to follow this rule, it inevitably results in a continual stream of broken builds. It becomes difficult to maintain the acceptance tests, which are still 'in progress' and not yet complete.

HERE ARE A FEW TECHNIQUES THAT CAN BE USED TO AVOID THIS PROBLEM:

Tag the 'work-in-progress' acceptance tests.

Tools like JBehave, Cucumber and SpecFlow allow tagging the acceptance tests that are not yet complete. They also help configure the Continuous Integration build to only run the stories without the work-in-progress tag. For example, a '@wip' tag can be used to mark a scenario that is work-in-progress.

Branching:

Branching technique can be used to develop new f eatures. With GIT, this is fast, easy and common practice. For teams, that use Subversion branching and merging, it can be a painful process but can still be a viable option. The workaround here can be to not let the branch live for too long (say, more than a couple of days), because long-life branches can create a risk of integration issues later.

Incremental Implementation:

Another approach is to break down the new feature into small pieces that can be built and delivered incrementally. The smaller pieces can be completed quickly and merged back into the master.

ATDD – An applied approach to continuous delivery

TO SUMMARIZE, THE KEY BENEFITS OF ADOPTING ATDD ARE:

- Unambiguous and executable specifications in the form of Acceptance tests that are understood by every member of the team
- Higher automated test coverage, that leads to reduced cycle times & faster time to market
- Strong collaboration amongst Customer, Developer and Tester
- Involvement of test team at an early stage in development process
- Unattended test execution when build gets deployed
- Finding bugs at the earliest
- Reduced cost of quality and increased productivity

ABOUT ASPIRE

Aspire Systems is a global technology services firm serving as a trusted technology partner for our customers. We work with some of the world's most innovative enterprises and independent software vendors, helping them leverage technology and outsourcing in our specific areas of expertise. Our core philosophy of "Attention. Always." communicates our belief in lavishing care and attention on our customer and employees.

SINGAPORE +65 3163 3050 | NORTH AMERICA +1 630 368 0970 | EUROPE +44 203 170 6115 | INDIA +91 44 6740 4000 | MIDDLE EAST +971 50 658 8831

For more info contact info@aspiresys.com or visit www.aspiresys.com