TRIGENT

Continuous Build, Deploy & Test: A Worry Free Product Release Methodology

Srikanth Ramamurthy
Technical Consultant

Trigent ■ 2 Willow Street, Suite 201, Southborough, MA 01745 ■ 1-877-387-4436 ■ www.trigent.com
Trigent Software, Inc. All trademarks, marked and unmarked, are the property of their respective owners.

Introduction

The software has to be released today. Its late evening, pizza has gone stale, the programmers are still feverishly putting in the last minute fixes, the build engineer is struggling to get a clean build to hand over to the test group which is waiting to certify it clean before it gets released. Somehow the team does succeed in releasing the software that night but barely so. The next morning bugs have started to fill in-boxes of the programmers before they straggle in late and realize that in the last minute rush a number of necessary features and bugs were missed out.

The above is the most likely scenario that is prevalent in the software world.

Now let's look at a desirable scenario below:
The software has to be released tomorrow. A code freeze is planned today after careful review of all the components and checked in after proper verification. Build starts the next day morning and a completed build is given to the testing group during early afternoon along with the release notes, a list of fixes and features for the release. The final product is certified shortly thereafter, finding no major bugs to hold up its release. The day ends normally by evening hours. As a bonus, the inboxes are not overflowing with complaints the next morning and the team has started work on the next release as its normal course of schedule.

This second scenario may seem like a dream, but it is not. It can be the new normal provided proper planning and practices are put in place.

At Trigent we achieve this second scenario by using what we call "Continuous Build, Deploy & Test". This paper describes how this practice can be constructed, the tools and methodologies to be used and the team responsibilities defined.

Test First Code Next

This concept of "Test First - Code Next" may seem counter-intuitive. However, if the requirements and design for the solution are known, one should actually think about how to best go about verifying these requirements in the completed software. Each functional requirement or feature and design components will need to be verified and hence the actual test approach and test cases can be created. By wearing a tester's cap one can get an outside-in view to the functionality to be implemented. This improves the overall understanding of the requirements and can lead to better design and more robust implementation.

The test cases also help in identifying "pre-state" data to be used for the creation of test database which

Trigent Software, Inc. All trademarks, marked and unmarked, are the property of their respective owners.

supports in generation of test data to be used for testing programs.

Trigent has developed a proprietary XML format and schema which are used to represent this test data. This schema is also used to identify pass and fail conditions in a flexible manner.

This activity not only helps get good test cases but helps compile these test cases as regression test suites. This self-documenting feature allows the software to be tested repeatedly and consistently, identify regression errors. It also supports a level of automation commensurate with the technical and budgetary demands of the work.

Automatic Code Review

While Trigent follows the practice of regular code reviews for its well known benefits, code reviews generally suffer from subjectivity and biases of the programmers and the reviewers involved. Hence, for code reviews to be effective in an organization, uniformity of standards and consistency of results are necessary. These can be obtained if code reviews can be automated to a degree. This becomes especially necessary when the code base is very large.

Trigent automates the code review process in order to achieve a high level of uniformity and consistency. This process prevents syntactically bad code from getting into the source repository.

We use one of the following open source tools for automated code review:

Hammurapi

This tool from http://www.hammurapi.biz is an open source code inspection tool. This tool checks the source code for compliance with a pre-defined set of rules or best practices. Trigent has selectively chosen some of the rule sets provided by this tool.

FindBugs

This tool from http://findbugs.sourceforge.net is a tool to find bugs in Java programs. It looks for instances of "bug patterns" -- code instances that are likely to be errors. FindBugs uses static analysis to look for general code quality problems.

EMMA

This tool from http://emma.sourceforge.net is an open-source toolkit for measuring and reporting the Code coverage. Trigent uses this tool to measure the effectiveness of the test cases. If the code coverage is below 60% then the test cases are enhanced to increase the coverage.

Trigent Software, Inc. All trademarks, marked and unmarked, are the property of their respective owners.

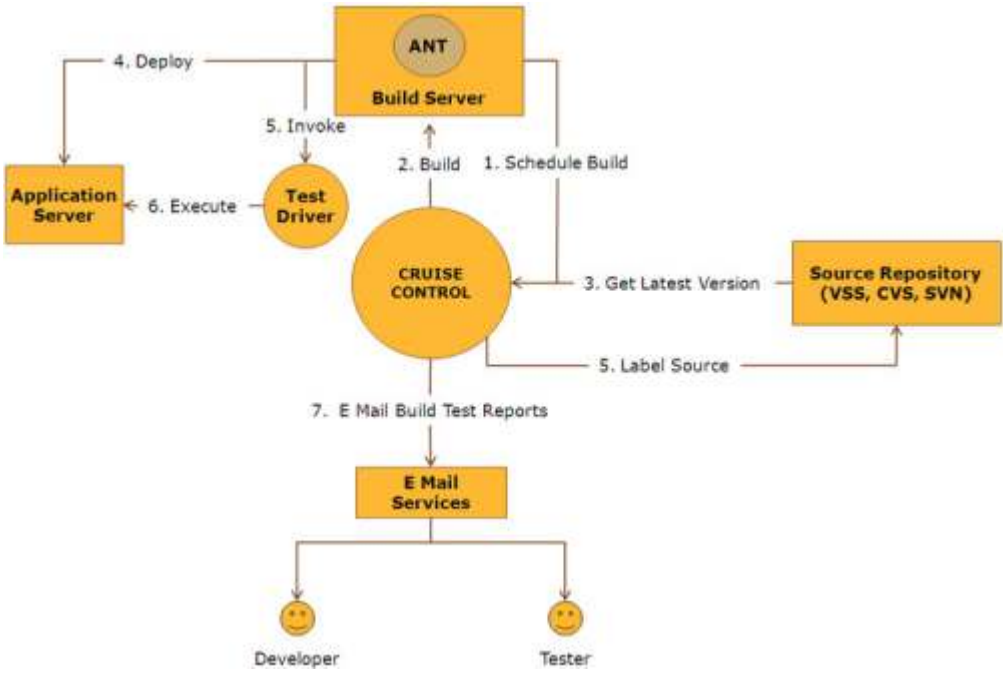
Trigent uses these tools in two ways:

- Integration with the build process. In this case, these tools are run against the code during the build process and helps in early detection of the code problems.
- Self-review. In this case, each programmer runs these tools against his/her code to find and fix the problems earlier on before putting his code into the source repository.

Automatic Build & Test

Trigent follows an automatic build & test process. The whole process executes without manual intervention. This enables the projects to have ready to ship code that is reviewed & tested at any given point of time. This process helps in continuous code integration resulting in reliable builds of the application.

This diagram depicts Trigent's automatic build process:



Trigent Software, Inc. All trademarks, marked and unmarked, are the property of their respective owners.

Tools Used

Below are the tools used in the automatic build process:

- CruiseControl is a framework for continuous build process.
- Apache ANT as a build tool
- Trigent's proprietary Test Driver to execute the test cases

Build Process

The build process is explained below:

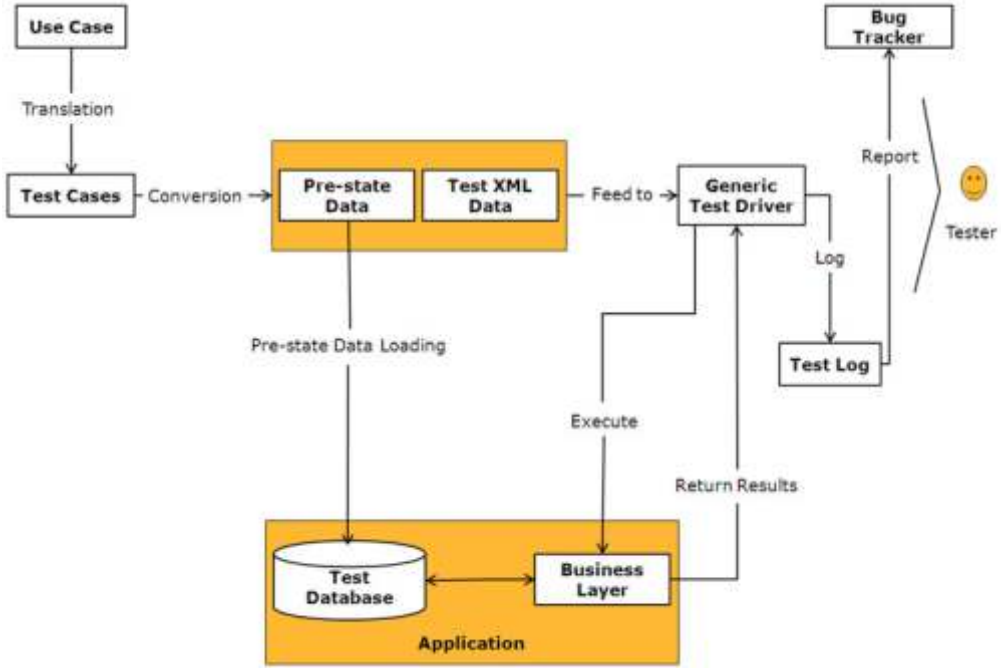
- Build engineer schedules the application build using CruiseControl. Usually the builds will be done during evening times.
- At the scheduled time, the build process starts.
- Latest copy of the source code will be obtained from the source repositories like CVS or Visual source safe.
- Automatic code reviews will be done on the source code.
- If any errors, the results will be emailed to the development team for further actions and the build process will fail.
- On no errors, the application will be built.
- The test database will be created using the pre-state data scripts.
- The application gets deployed on the test application server.
- The latest copy of the regression test suite xml files will be obtained from the source repository.
- The test driver will be invoked and the test xml will be executed against the application deployed.
- The test results will be logged to a log file.
- The test result log file will be emailed to the testing & development teams for further actions.
- The source code gets labeled appropriately in the source repository.

Test Process

Trigent uses a homegrown test driver. This driver is generic and can be used test the business and data access layer code. The test driver is implemented on top of jTestCase from Sourceforge

This diagram depicts Trigent's test process:

Trigent Software, Inc. All trademarks, marked and unmarked, are the property of their respective owners.



The test process is explained below:

- Process starts with the translation of business use cases into test cases. Tester translates the use cases to appropriate test cases
- Tester converts these test cases to:
  - Pre-state data required for the test database creation
  - Test data in terms of test xml
- Test xml and pre-state data will be put into the source repository
- Tests will be executed by:
  - Programmers to test their modules
  - Testing group to system test the application
  - Automatic build process to validate the build
- Tester will verify the test logs for errors
- Bugs will be logged into the bug tracker
- Development team will take further actions on the bugs found

Conclusion

With "Continuous Build, Deploy & Test" practice, Trigent has seen overall reduction in risk with respect to project delivery. Release process is streamlined. It has brought complete transparency into the build and testing process.

Programmers are disciplined, as the entire team will know if the build fails and the culprit behind that

Trigent Software, Inc. All trademarks, marked and unmarked, are the property of their respective owners.

failure. The team is careful and makes sure that bad code does not get into source repository. Code is integrated continuously and bugs are discovered earlier on instead of waiting till the last minute. Regression test suites are enhanced leading to reliable application resulting in timely delivery making the customer happy.

About Trigent Software Inc.

Trigent is a privately held, professional IT services company and a Microsoft Gold Partner with its U.S. headquarters in the greater Boston area and its Indian headquarters in Bangalore. We provide consulting services in various technologies including Microsoft Solutions. Our operating model is to conduct sales, customer relationships and front-end consulting (e.g., business case, requirements, architecture) onsite with our clients and perform the detail design, development, integration, testing and quality assurance offshore at our world class development and support center in Bangalore. We are a SEI CMM Level 4 company and is ISO 9001:2000 TickIT certified organization.

For sales contact sales@trigent.com or call 508-490-6000.

Microsoft Partner
Gold Application Development
Gold Collaboration and Content
Silver Mobility

Trigent Software, Inc. All trademarks, marked and unmarked, are the property of their respective owners.