

# Mobile Platform Feature Testing

A SIPTECH CASE STUDY

Version: 1.03

Created Date: 13.Dec.2004

---

## Inside this document ...

- 01. Background
- 02. Problem / Challenges
- 03. Method / Intervention / Solution
- 04. Results Achieved
- 05. Lessons Learned / Conclusion

**SIPTECH**  
VALUE... Perfectly Engineered.

© 2005 All rights reserved. SIP Technologies and Exports Ltd. The information contained in this document is CONFIDENTIAL and PROPRIETARY in nature, and subject to the rights and ownership of SIPTECH. Any and all unauthorized copying or use of the contents hereof is prohibited.

## 1. Background

Mobile devices based on the J2ME Connected Device Configuration are coming to market very quickly bringing with it new capabilities and challenges. This market is being addressed by a leading mobile and wireless industry expert by developing a framework that will allow developers to create applications as simple, interoperable, sharable components that can be dynamically deployed. In order to minimize fragmentation, accelerate adoption, and reduce time to market, this framework references existing standards and implementations wherever possible.

The goals of the new architecture include independence, reusability, scalability, security, interoperability, simplicity and good performance. To ensure that these goals are met by the software components sourced within the mobile expert organization and from outside parties, the project adopted an iterative development and testing model.

## 2. Problem / Challenges

Since the framework was being built on Java and open standards, the requirements and various components that comprised the framework came from different geographic locations of the mobile expert group. In addition there were third party contributors like OSGi, Monta Vista, Sun etc. A lot of interaction was thus involved bringing with it a need for cross-cultural sensitization.

The platform is being built on iterative model. Iterations were aggressive with a duration of 4 weeks and often overlapped. Delays caused by the high inter-dependency between the various components and frequent requirements creep would further compress the time available for testing.

The components of the framework are a combination of active elements with no user interaction (services), active elements with user interfaces (applications), and shared libraries (both native and Java). Testing these components therefore involved interaction with the Java and Native layers.

A strategy to cover the entire gamut of testing had to be evolved. In addition to API and functional testing, there was also a need to develop end-to-end and inter-operability tests. Nightly sanity checks, regular acceptance tests and complete regression cycles had to be executed. Conformance tests had to be executed for the various industry specifications that were being implemented.

As the test suite grew in size, regression became a great challenge. In order to ensure that the tests were being executed on an environment that was close to the actual target environment, the tests were run on desktop systems, mobile simulators and prototypes.

As the target phone was a dual processor phone, the hardware brought with it the many challenges of flashing the base processor and the application processor on to the phone and getting both to communicate effectively. Moreover these were prototypes and therefore limited in numbers. Resource availability continued to be a great challenge.

### 3. Method / Intervention / Solution used

SIPTECH is a global testing centre for testing the platform. Separate projects were spawned off under the testing program umbrella, with each project team interacting with client teams in various geographic locations. Regular program level meetings were held to facilitate formulation of common testing procedures and strategies and sharing of knowledge and resources.

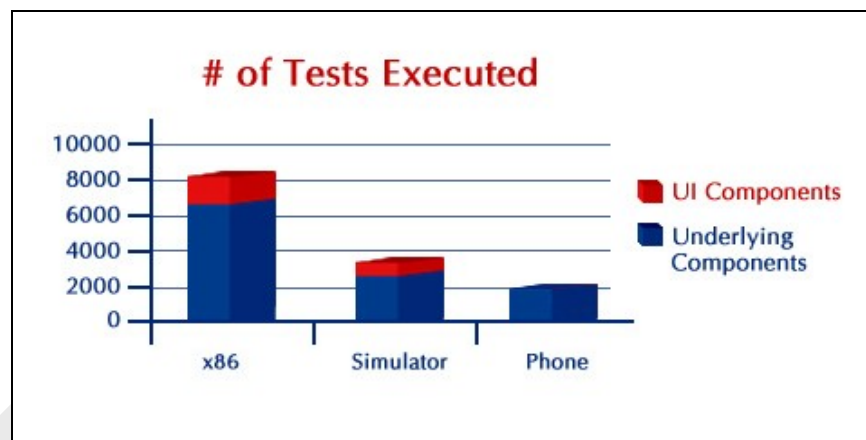
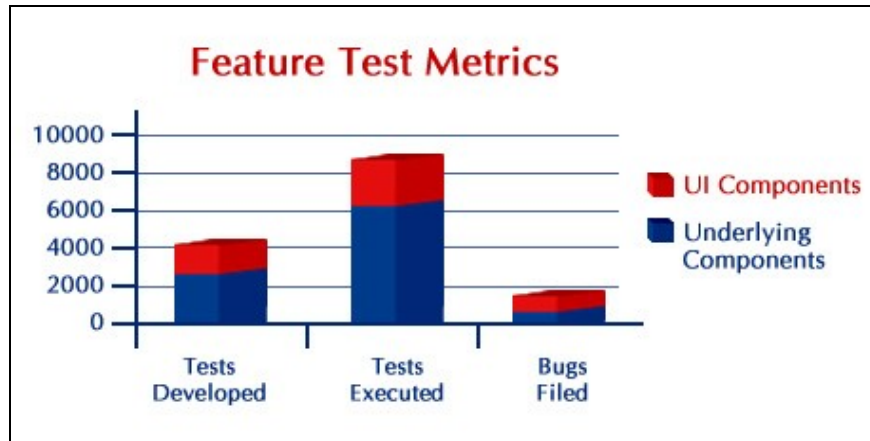
Different types of testing like API, UI, and functional testing were done to cover all the features of the product. At the end of each milestone, Java and Native code coverage analysis tools were used to identify areas where testing needed to be improved. Various testing techniques like exploratory testing and category partitioning were introduced to maximize coverage. Detailed test performance metrics were submitted to the client.

Interoperability tests were developed to ensure compatibility with servers from different vendors. End- to-End testing was done to completely validate the features of each component. As the components were integrated, feature interaction testing was done.

Once the platform became reasonably stable, the tests were automated using the client's proprietary test frameworks and other tools like Automate5. This allowed SIPTECH to focus on increasing coverage and improve performance.

### 4. Results Achieved

Total no. of tests developed by SIPTECH	4103
Total no. of tests executed by SIPTECH	9167
Total no. of bugs filed by SIPTECH	1325
Current code coverage %	82%
Average test development time	~2 hrs.

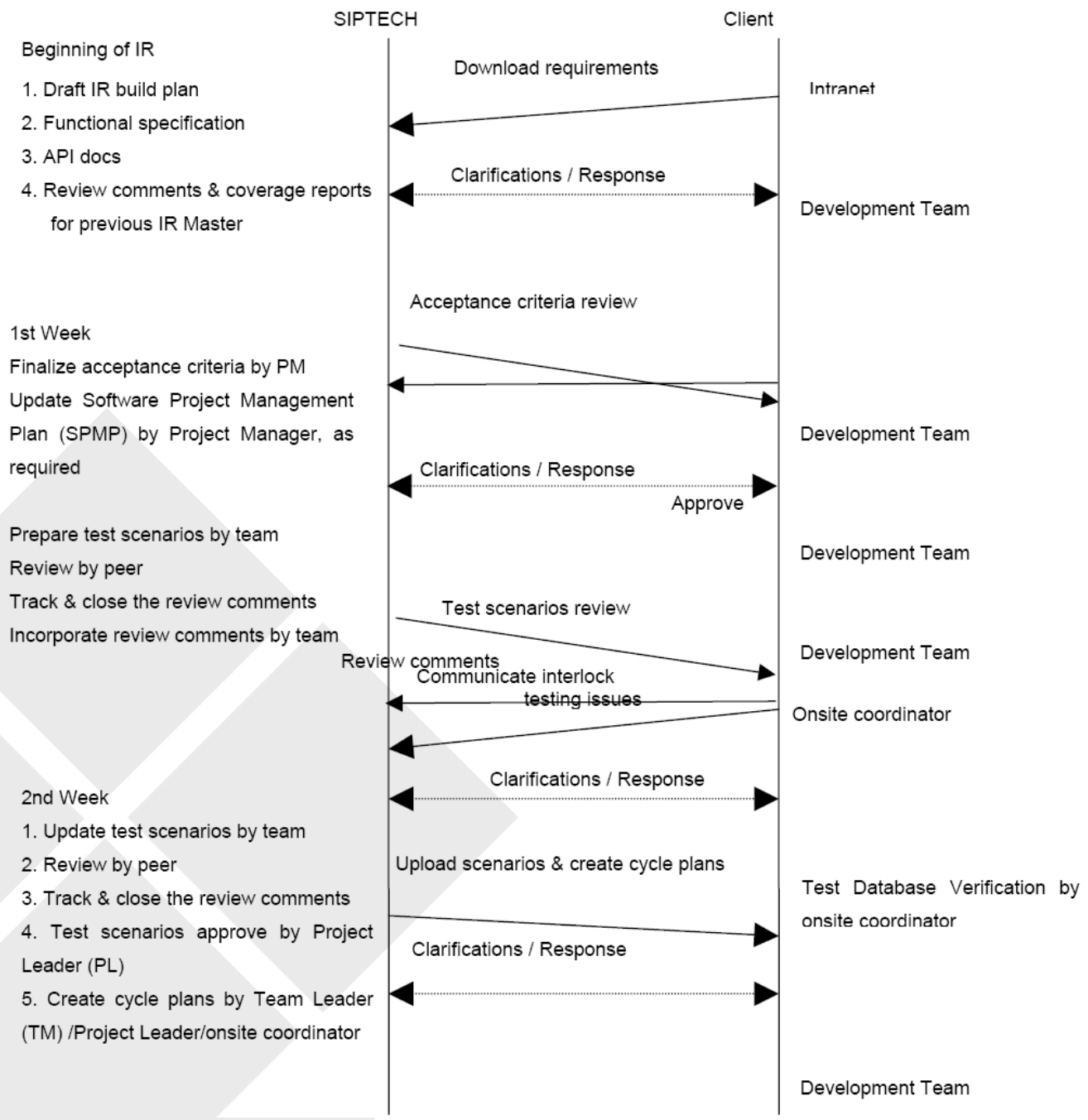


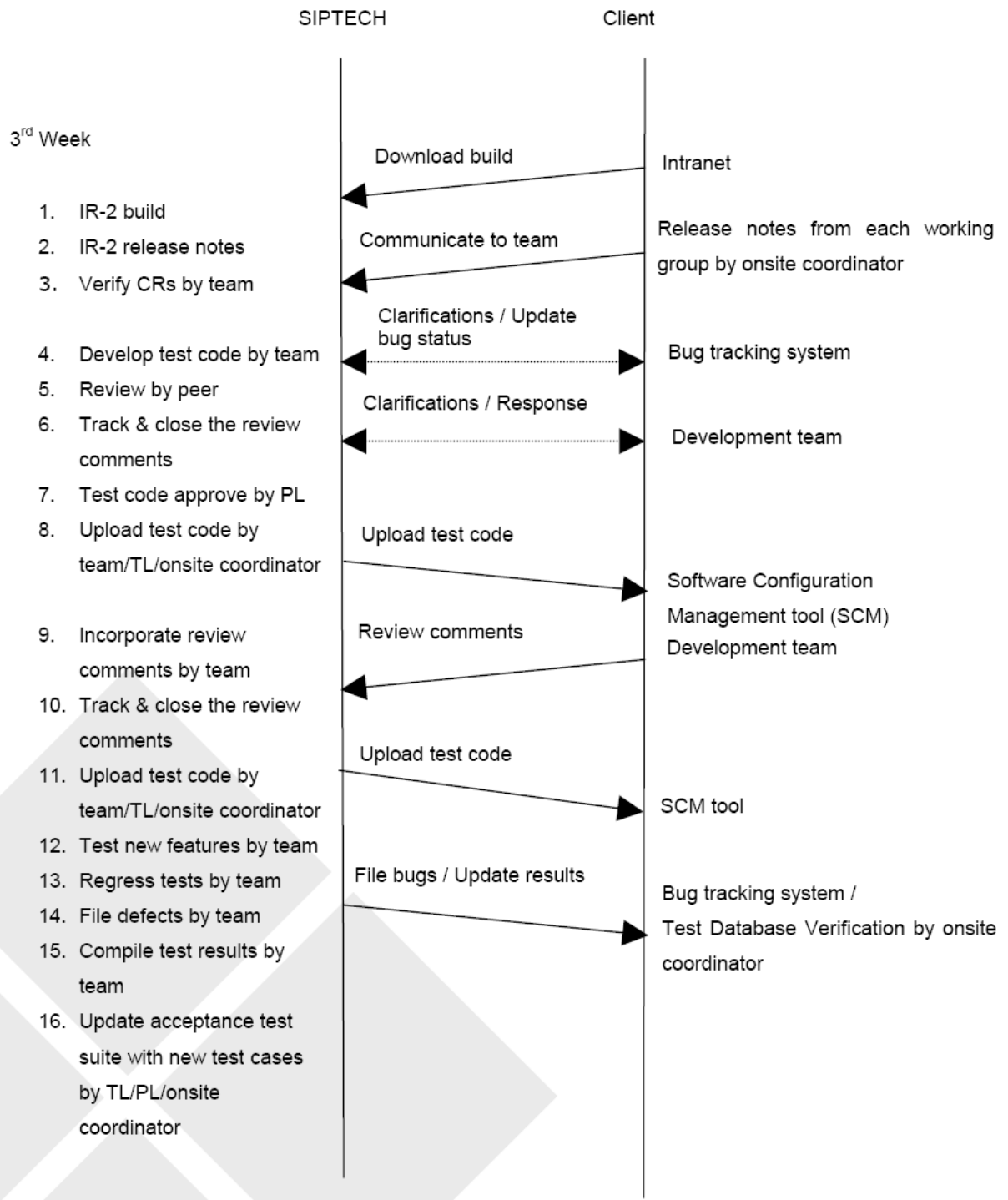
## 5. Lessons Learned / Conclusion

1. Automating test execution reduces the time spent on execution and increases the turn around time for capturing bugs.
2. Measuring code coverage in frequent intervals helps to cover the uncovered areas in testing.
3. Documentation is a major role for a success in a global project. Documents need to be created having the following criteria in mind:
  - a. Issue tracking to closure.
  - b. Collecting, analyzing and reporting metrics using custom templates
  - c. Risk Identification and mitigation for the same.

**APPENDIX A**

Incremental Release (IR) Process





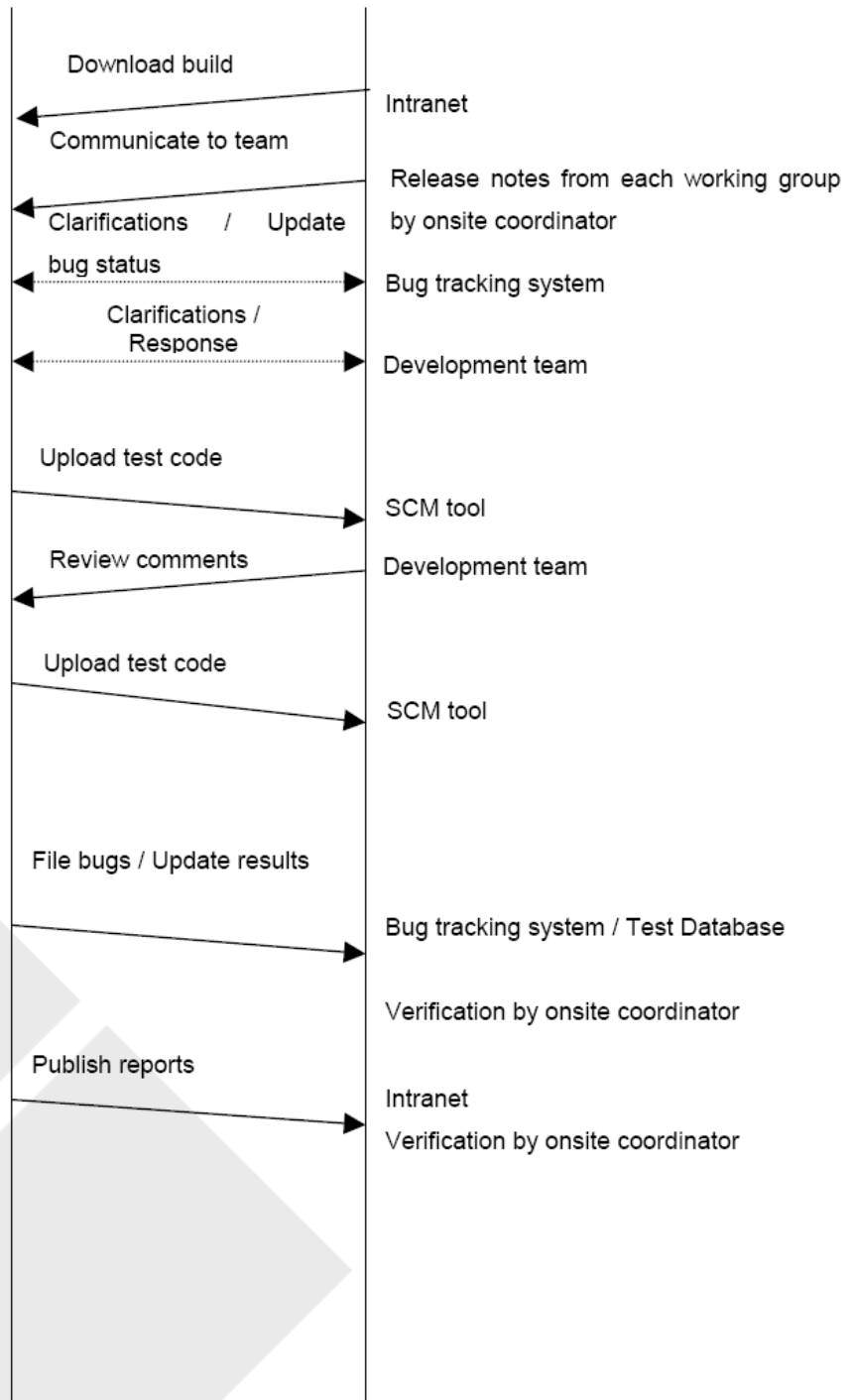
4<sup>th</sup> Week

1. IR-1 build
2. IR-1 release notes
3. Verify CRs by team

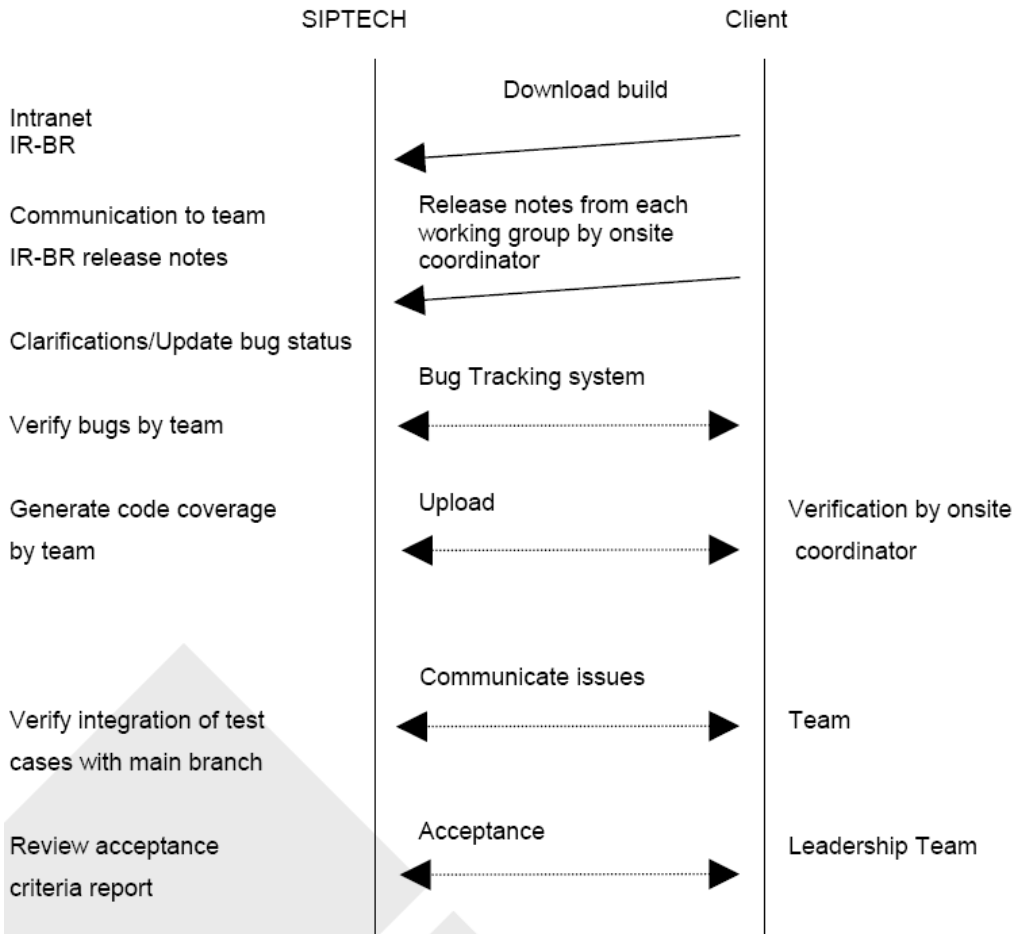
4. Develop test code by team
5. Review by peer
6. Track & close the review comments
7. Test code approve by PL
8. Upload test code by team/TL/onsite coordinator
9. Incorporate review comments by team

10. Track & close the review comments
11. Upload test code by team/TL/onsite coordinator
12. Test new features by team
13. Regress tests by team
14. File defects by team
15. Compile test results by team

16. Update acceptance test suite with new test cases by TL/PL/onsite coordinator
17. Consolidated metrics & generate acceptance criteria report by PL/QC
18. Approve by PM



Beginning of next IR



- API - Application Programming Interface
- BR - Build Release
- IR - Milestone Release
- PL - Project Leader
- PM - Project Manager
- QC - Quality Coordinator
- SCM - Software Configuration Management
- SPMP - Software Project Management Plan
- TL - Team Leader