Testing In Iterative Product Development Environment

Shivageeta S. Choodi
cshivageetha@novell.com
Software Consultant
Novell Bangalore

6th Annual International Software Testing Conference 2006

Novell Software Development India Pvt. Ltd.
No.49/1 and 49/3, Garvebhavipalya,
7th mile, Hosur Road
Bangalore – 560 068

Abstract

Most of the software product companies are adopting iterative model for product development because of the following reasons,

- At any given point of time there is a **working model** of the product.
- **Increased control** over project progress.
- Customers are more confident as companies can demonstrate **proof of concept** in the earlier stages of product development life cycle.

Iterative model demands a new process for product **development and testing. This study is very critical to Novell,** as many of Novell products are adopting iterative model for the development. This paper deals with the **testing processes** followed in the **industry and Novell** for iterative model of product development and **challenges faced** by the test teams in adapting to this new model and **changes required** from the testing point of view.

Contents

Testing In Iterative Product Development Environment 0
Abstract 1
1 Background 3
2 Introduction to the Iterative Model 3
3. Testing in Iterative Model – Industry View 4
    3.1 Unit Test Process 4
    3.2 Component Test Process 4
    3.3 Integration Test Process 4
    3.4 System Integration Test Process 5
    3.5 Acceptance Test Process 5
    3.6 Test Automation 5
    3.7 Defect Fixing and Verification 5
4 Testing Approaches Adopted for Iterative Model at Novell 6
    4.1 Approach A 6
    4.2 Approach B 7
    4.3 Approach C 9
    4.4 Summary of the Three Approaches 11
5 General Observations 12
6 Best Practices 12
7 Challenges and Suggestions 13
8 Testing Benefits in the Iterative Model 14
9 Conclusions 14
References 15
About the Author 16
Appendix I - Abbreviations 17
Appendix II – Definitions 18

1 Background

Iterative model for product development is becoming more popular and widely used. The conventional testing processes are not suitable for the iterative model. Therefore, test engineers need to adopt new test strategies for products developed on iterative model and evolve tailor-made testing processes.

To harness the potential of product development in an iterative environment a systematic study was carried out to understand the development model and testing process followed in the industry and at Novell

Different product stakeholders like developers, Integration test engineers, and system integration test engineers were interviewed, to understand the testing process and its implications on products adopting iterative model.

2 Introduction to the Iterative Model

Iterative model is an agile methodology. It advocates delivering working software in short, consistent, and frequent intervals. It addresses most of the complex and high-risk items early in the development cycle.

In iterative development, the project plan arranges the product development into small releases, and mandates continuing integration of all coded components, incremental builds, and periodic validation of refined requirements and design. Each of the iteration follows a model derived from a V-model / Waterfall model, that is, each iteration has requirements analysis, design, coding, and testing phases.

The following figure depicts a typical iterative model for software development:

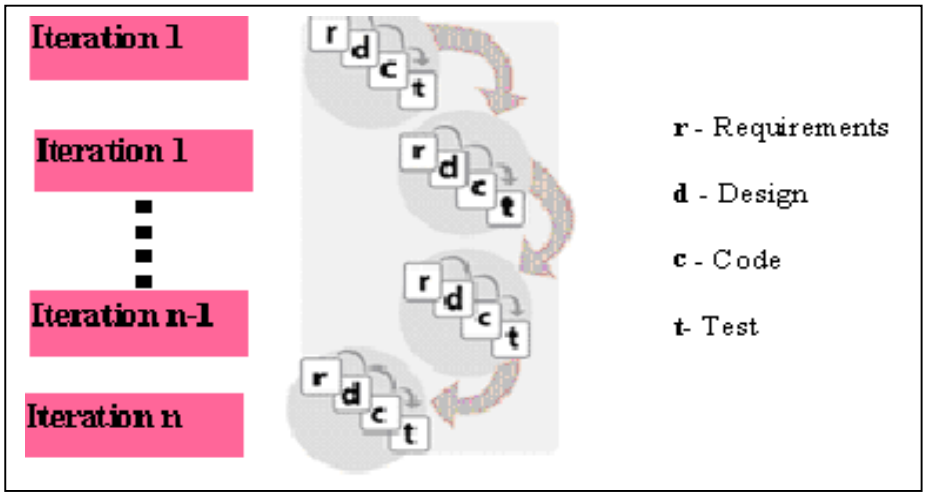

Figure -1

The salient features of iterative model are as follows,

- Quick feedback loop from business stakeholders to engineering and then back to business stakeholders
- Ability to refine requirements and design
- Early defect detection
- Focus on implementing the high priority and the high risk features at an early stage of development
- Short delivery mechanism to improve the quality of output.

3. Testing in Iterative Model – Industry View

The key to successful use of an iterative software development lifecycle is rigorous validation of requirements and testing of each version of the software against those requirements within each cycle of the model.

The software developed at each of the iterations will go through the following test phases:

- Unit testing (UT)
- Component testing (CT)
- Integration testing (IT)

Once the entire product requirements are met, that is, during the final iterations, the product goes through the following test phases:

- System integration testing (SIT)
- Acceptance testing

How ever some practitioners suggest performing system integration test at of the each iterations.

Note: Please refer to Appendix-II to understand the different test phases.

3.1 Unit Test Process

The main focus of unit testing is to uncover the code level defects. If *N* is the current iteration, unit tests for iteration *N* are designed in the same iteration by the developers and peer developers review them. Tests are executed in the same iterations, and the defects encountered during the testing are fixed during the same iteration.

3.2 Component Test Process

There are no guidelines on the time frame for carrying out the component tests, the following process can be followed - if *N* is the current iteration; the tests are designed, reviewed, and base-lined in the N^{th} iteration based on the scope of the current iteration and the tests are executed in the *N+1* iteration.

3.3 Integration Test Process

Testing process is similar to that of component testing. A regression test suite or an accumulated test suite is run to test the interfaces of the components of the current iteration with the components of the previous iterations.

3.4 System Integration Test Process

Agile development projects face significant risk of insufficient testing of the application, particularly outside of a pure code coverage perspective. To decrease the risk of insufficient testing, the following tests are applied to the overall system when the product is completely ready:

- Stress tests
- Load tests
- Performance tests
- Reliability tests
- Security tests at system level

How ever some practitioners suggest running relevant system integration tests optionally with reduced test scope even in the iterations.

Note: Please refer to Appendix-III for the definitions of the different testing types.

3.5 Acceptance Test Process

A test performed by an end user or system owner to verify that the delivered software functions correctly and meets requirements. Usually these tests are derived from use cases of the particular iteration to be tested. The testing process is similar to that of component testing.

3.6 Test Automation

The iterative model emphasizes on the test automation due to the following test requirements:

- Running integration and regression tests for every iterations
- Running more tests in a short span of time (within the time span of an iteration)

Even though there is need for automation, there is no emphasis on the time frame for development of the automation tool.

3.7 Defect Fixing and Verification

Managing defects within an iterative process is one of the most complex elements of a project. Its always recommended to factor in the defect fixing and regression activities in the development and test plans. A common approach followed to manage defects is as follows - if N is the current cycle; unit testing and defect filing, fixing, and regression happens in the N^{th} iteration. The CT, IT, and SIT happen in *N+1* iteration and the defects are filed. Defect resolution (fixation + regression) happens in *N+1* or *N+2* iteration depending on the defect priority.

4 Testing Approaches Adopted for Iterative Model

At Novell, Bangalore many product teams follow the iterative model of product development. The product teams have tailored the model for the development and testing activities based on the project needs and characteristics, such as:

- Product size: small, medium, or large scale
- Project duration
- Team size
- Release type: new, enhancement, or maintenance release
- Geographical distribution
- Project phase at which the model has been adopted

This section focuses on the different testing approaches followed at Novell Bangalore for the iterative model.

4.1 Approach A

This is a large-scale project, where the product is built from scratch. The main reason for adopting this model is to track and control the product development activities as it involves a lot of interdependent components and cross functional teams

- Project Characteristics
- Product size: Large scale
- Project duration: 24 months
- Team size: 42 people
- Iteration size: 2 to 6 weeks
- Release type: New product release
- Geographical distribution: Cross functional teams spread across different geographical locations.

Test Team's Participation

Only unit test, component test, and integration test teams participate in each of the iterations.

ST follows the classical waterfall or V model for test design, automation and execution.

Test Design

Unit test cases for the iteration *N* are designed and base-lined in iteration *N-1* or N^{th} iteration.

Component test cases and integration test cases for the iteration *N* are designed and base-lined in iteration *N.*

System integration testing followed the traditional V-Model, that is,. system integration test cases design and baseline process are not inline with the iterations. The test design starts once the base-lined product requirements documentation and HLD are made available.

Test Automation

Test frameworks for UT, CT, and IT are made available before the iteration process stared.

Unit test automation for iteration *N* was starts in the iteration *N-1* or *N.*

Component and unit test automation for iteration *N* starts in iterations *N* and *N+1*

System integration test automation tool development starts as soon as the system integration test cases are base-lined, and are independent of the iteration model.

Test Execution and Defect Tracking

Unit testing for the software in iteration *N* is done in iteration *N*. The closure of the defects found by unit testing is done in iteration *N* itself.

Component and integration test execution for software in iteration *N* are done in iteration *N+1* along with defect regression for the defects filed on the software of *N-1* iteration.

Along with component and integration test suite of the given iteration, the accumulated test suites are executed.

Execution efforts increase for every iteration, as the tests get accumulated with every iteration.

System integration test cases get executed during the last iterations when the product is almost feature complete as in the classical waterfall or V-model

4.2 Approach B

This is medium scale project, where product is in the maintenance state. The team works on multiple related products, which also follow the iterative model and all the products get bundled and released at the same time. Development of all the products of the software bundle happen concurrently using the common iteration process. The main objective of adopting this model is to be in sync with other project schedules and to handle multiple products in the same iteration cycles.

Project Characteristics

- Product Size: medium scale
- Project duration: 9 months
- Team size: 10 people
- Iteration size: 1-3 weeks
- Release type: Maintenance release
- Geographical distribution: Product team at one geographical location
- Project phase at which the model has been adopted: During coding phase.

Test Team's Participation

The entire test team; unit test team, component test team, integration test team, and system integration test teams participate in the iterations.

Test Design

Unit test cases iteration *N* are designed and baselined in *N-1* or *N*th iteration.

Component and integration tests for each of the iteration are designed as and when the product architecture and the design documents for components are baselined.

This approach is followed as the product architecture and designs and the iteration scope for each of the iterations are finalized during the initial iterations of the project.

System integration test cases for each of the iterations are designed as soon as the product requirements, architecture documents, and iteration scopes are made available.

System integration test cases are more focused on system integration testing, security testing, functionality testing, and a small scale testing for product performance and scalability.

Another set of system integration test cases are also designed that are executed on the complete system (during last iteration when the complete system is ready), to test the following system parameters

- Stress
- Reliability
- Scalability
- Performance
- Interoperability

These tests ensure that system integration testing don't lose sight of the *forest for the trees*.

The regression suite for iteration *N* is identified in the same iterations and gets executed in the next iteration.

Test automation

For UT, CT, IT and SIT - the test automation frameworks that are developed for the previous release are reused.

Unit test automation for iteration *N* starts in the iteration *N-1 or N*.

The test automation tool required for iteration *N* (for CT, IT, and SIT) is developed and integrated with the respective frameworks in *N*th iteration it self

The development of the system integration test automation tool, needed to execute the final system level test cases on the complete product is not inline with any of the iterations.

Test Execution and Defect Tracking

Unit testing for the software in iteration *N* is done in iteration *N*. The closure of the defects found by unit testing is done in iteration N itself.

Component, integration, and system integration test execution for software in iteration *N* is done in iteration N along with the execution of the regression test suite.

During the final iterations, the set of system integration test cases such as stress, reliability, and scalability designed to run on the complete system are executed.

4.3 Approach C

This is a small-scale project, where a major and independent component of a large product is developed using the iterative model. Major requirements for the component are the basic design changes and new features implementation. The main objective of selecting this model are meeting these requirements with less time and less defects, as the component is part of a critical product, which needs to be delivered in a short span of time

Project Characteristics

- Product size: Small scale
- Project duration: 6 months
- Team size: 5 people
- Iteration size: 2-8 weeks
- Release type: Enhancement release
- Geographical distribution: Component team at one geographical location
- Project phase at which the model has been adopted: During architecture and design phase.

Test Teams Participation

Entire test team - unit test team, component test team, integration test team, and system integration test teams participate in the iterations

Test Design

Unit test cases for the iteration *N* are designed and baselined in iteration *N-1* or *N*.

Component test cases, integration test cases, and system integration test cases for the iteration N are designed and baselined in iteration *N* itself

Another set of system integration test cases are also designed that are executed on the complete system (during the last iteration when the complete system is ready), and tests the system parameters such as:

- Stress
- Reliability
- Scalability
- Performance
- Interoperability

The regression suite for iteration *N* is identified in the same iterations and gets executed in the next iteration.

Test Automation

Most of the test execution is manual and no automation frame works are used. Simple and independent scripts are written where automation is a must, that is, to execute stress, reliability, scalability, and performance tests.

Unit test automation for iteration *N* is completed in the iteration *N-1 or N*.

The test automation scripts required for iteration *N* (for CT, IT, and SIT) are designed and developed in the N^{th} iteration itself, by the respective test teams.

For executing set system integration test cases designed to execute on the complete system, the automation tools of previous release are used with required modifications and enhancements.

Test Execution and Defect Tracking

Unit testing for the software of iteration *N* is done in iteration *N*. The defects found by unit testing are resolved in the iteration *N* itself.

Component, integration, and system integration test execution for software of iteration *N* is done in iteration *N+1* along with the execution of the regression test suite.

In the last iteration, a set of system integration test cases designed to run on the complete system such as stress, reliability, and scalability are executed.

4.4 Summary of the Three Approaches

Approach and Project Characteristics	Test Design	Test Automation	Test Execution and Defect Tracking
Approach A Product size: Large scale Project duration: 24 months Team size: 42 people Iteration size: 8-2 weeks Release type: New release Model Adoption: During requirement phase	UT one iteration ahead CT and IT - Inline with iterations SIT-Follows conventional V model	Use of test automation frameworks, which are developed before starting with iterations. UT one iteration ahead CT and IT -Inline with iterations SIT-Follows conventional V model	UT-Inline with the iterations CT, IT lags behind by one iteration For each cycle accumulated test suite is run along with test cases for the current iteration Complete SIT execution during the final iterations
Approach B Product size: Medium scale Project duration: 9 months Team size: 10 people Iteration size: 3-1 weeks Release type: Maintenance Release Model Adoption: During coding phase	UT test design is inline with or an iteration ahead of the current iteration CT, IT, and SIT - before the iteration-0 all the test cases are base-lined A separate set of SIT test cases to run on a complete system	Test frameworks of the previous release are used UT one iteration ahead CT, IT and SIT – Required test scripts are developed in the same iteration. SIT-Follows conventional V model for the test tool needed to run the tests on the complete system	UT, CT, IT, and SIT Inline with the iterations. For each cycle, regression test suite is run along with test cases for the current iteration A set of SIT test cases is run on the complete system during the last iteration.
Approach C Product size: Small scale Project duration: 6 months Team size: 5 people Iteration size: 8-2 weeks Release type: Enhancement release Model Adoption: During design phase	UT test design is inline with or a iteration ahead of the current iteration CT, IT and SIT - Inline with iterations A separate set of SIT test cases to run on a complete system	No automation frameworks More of manual testing. Test script development is inline with the iterations. Simple scripts are written for the tests where automation is a must	UT-Inline with the iterations SIT, CT, and IT lags behind by one iteration For each cycle regression test suite is run along with test cases for the current iteration A set of SIT test cases is run on the complete system during the last iteration.

5 General Observations

After understanding the test process in the above three approaches, the following common observations are made. These observation can be considered while planning for the iterative projects:

1. For products developed from scratch (new product developments), accumulated set of test cases are run for each integration, to reduce the risk of dependency breaks and integration issues.
2. For maintenance or enhancement releases, a regression suite is run instead of accumulated set of test cases as the chances of dependency breaks and integration issues are less.
3. Test automation is critical and a must for new product developments as accumulated sets of tests are repeated in each of the iterations.
4. Iteration size gradually reduces towards the release time to cater to the following:
    a. Need for frequent defect fixation/regression
    b. Release pressures.
    c. Scope changes.
5. Additional iterations are introduced or existing iterations extended to cater to major design changes or arrival of new requirements.
6. For the maintenance or enhancement projects, test efforts are almost same as classical waterfall or V model (as only regression tests are run for every iteration), but product will get tested along with the coding phase, that is, the code is almost tested at any given point of time.
7. Strict adherence to schedule is expected, as all the planned activities need to be completed with in the given iteration, which is a very short duration.

6 Best Practices

The teams followed some of the best practices, which helped a quicker release yet with quality. The practices can be generic in nature and can be easily adopted during the product development process.

1. A practice called “Test Based Development” was followed to reduce the defects found during UT.
2. Defect Prevention was achieved by
    a. Peer reviews
    b. Test team involvement in design discussions and design reviews
    c. Rigorous test execution
3. Use of freely available test automation frame works like Jtest®, which can be used during iterations to plug-in required tests.
4. Regular process health checks through post iteration analysis meetings to identify areas of improvement and improvement plans.

5. Planned buffer iterations to cater to new requirements arrival and clearing of long lasting bugs.
6. Use of "Adaptive test planning" methodology. In the cases where the software is not ready for a given iteration, instead of waiting for the development team to release the code, you can execute the tests that can be executed on the current software but were planned for the later release due to the test criticality or time availability. This will create buffer for test activities of future iterations.

7 Challenges and Suggestions

Challenge 1:

Paradigm shift: Difficulty in adapting to a new model.

Suggestions:

- Training on the iterative model and its applicability
- Incremental migration towards the new model and realizing the benefits of the new model.
- Buy-in and support from the management team

Challenge 2:

Test repetition is more, especially regression tests and integration tests

Suggestions:

- Automate the tests that need to be run repeatedly.
- Have a dedicated iteration after every 3-4 iterations, which focuses on regression tests and extends integration tests, which can test the integration of the components from the past iterations.

Challenge 3:

Overcoming the paradox that SIT can't participate in the iterative model, as the software is released in pieces, and system integration testing has to be done on the partially built software.

Suggestion:

- Need to redefine system integration testing.
- SIT can test the system parameters like performance, scalability, and reliability from the initial stages. Its always better to test early and get defects early and fix them.
- If it is really not worth for SIT to test the software produced during some of the iterations, follow the adaptive test planning approach (which is detailed in the section 6), or regress the defects, or use the time for test automation.

8 Testing Benefits in the Iterative Model

- Shift from the paradox that “Test team always work under pressure as testing is the last phase of a product release”.
- Product testing does not get compromised as testing will happen along with product development as compared to conventional models, where usually testing gets compromised as it is needed to absorb the delays from the product development phase, that is, trade time (ship now but buggy) against testing (ship late).
- Early defect detection and fixation ensures better product design or architecture.
- Detailed and exhaustive testing; as the product undergoes rigorous testing at each of the iterations
- Low burden on testing during intermittent release (Alpha release, beta release etc.) because at any point of time, the product is usable, testable and tested with all the critical features/requirements implemented.

9 Conclusions

The nature of the tester's role changes in iterative projects. We are no longer the high-profile victims, we are no longer the lonely advocates of quality; we are merely (!) competent service providers collaborating with a group that wants to achieve high quality.

Quality and efficiency are best served by testing software as early as possible in the life cycle. Its up to us to tailor the test process to the best suited way and participate as early as possible in the product development life cycle.

Iterative model concepts can be applied to any project independent of size, type, and duration, to avail the benefits of agile methodologies.

References

Software Testing and Software Development Lifecycles
- *IPL Information Processing Ltd*

XP, Iterative Development, and the Testing Community - Stickyminds
- *Cem Kaner*

Iterative development Testing approaches
- *Bryan Campbell*

Iterative Software Lifecycle Process
- *objective software*

Jtest® - Tool for Java unit testing and code complience.
http://www.parasoft.com/jsp/products/home.jsp?product=Jtest&itemId=14

About the Author

Shivageeta is working as a Software Consultant at Novell®, India. She started her career as test engineer and is passionate about testing technologies. She has more than 5 years of experience in test planning, strategizing, designing, automating and executing tests.

Participation in testing forums/conferences:

- Has conducted a half a day tutorial on "**Test Methodology for Non-Functional Testing** " - QAI conference 2004 held at Pune.
- Has conducted a half a day tutorial on "**Best Practices for system Test**" - QAI conference 2005 held at Bangalore.

Qualification:

BE Computer Science from Karnataka University, Dharwad.

Appendix I - Abbreviations

UT – Unit Test(ing)
CT – Component Test(ing)
IT – Integration Test(ing)
SIT – System Integration Test(ing)
PRD – Product requirements Document.
HLD: High Level Design
TC – Test Case
S/W – Software
ICO: Iteration Change Order;

Appendix II – Definitions

Acceptance testing:
The objective of this testing is to verify that the delivered software functions correctly and meets requirements.

Accumulated Test Suite:
In iterative testing, the test cases of each of the iteration are accumulated in to the existing suite of test cases to form cumulated set of test cases. This suite is run at each of the iteration and the test suite size increases with each of the iteration.

Component testing:
The objective of component testing is to exercise individual components through their exported interfaces.

Development testing:
Component Testing + Integration Testing

Integration testing:
The objective is to test the interactions between the set of components

Interoperability testing:
The testing is done to ensure the two or more products can exchange the information and use the information that have been exchanged.

Iteration Change Orders (ICO)s:
ICOs are raised when:
The features planned for current iterations are pushed to future iterations
New features are pushed in to the current iteration.
Design for a component is getting changed during the final stage of the particular iteration

Performance Testing:
The testing conducted to evaluate the time taken or response time of the product to perform its required functions under stated conditions in comparison with different versions of same product and competitive products.

Product Development Process:
Refers to all the activities that are involved in releasing a software product such as:
Product requirements gathering
Product architecture/design

Test Designing
Coding
Test Automation
Artifacts reviews
Different testing phases
Defect fixation/regression.
Product Validation

Regression Test Suite:
Test suite designed with the test cases to test the basic production functionality along with the impact of code changes like defect fixes and design changes.

Regression Testing:
Selective retesting of the product to verify the modifications have not caused unintended effects and that the system or component still complies with its specified requirements.

Reliability Testing:
The testing conducted to evaluate the ability of the product to perform its required functions under stated conditions for a specified period of time or number of iterations

Scalability Testing:
A testing, which requires enormous amount of resources to find out the maximum capability of the product parameters.

Security Testing:
The testing conducted to examine and analyze the safeguards required to protect an Information System (IS), as they have been applied in an operational environment, to determine the security posture of that system

Stress Testing:
Testing conducted to evaluate a system beyond the limits of the specified requirements or product resources (such as disk space, memory, processor utilization, and network congestion) to ensure the product does not break unexpectedly.

System Integration Testing:
The objective of this testing is to test the set of components work as expected with the other system parameters like operating systems, routers, and third party applications.
The focus is also on to test the non-functional aspects like (performance, stress, load, reliability, and security)

Unit Testing:
The objective of unit testing is to exercise the product in small testable increments with focus at the code level.