

Metrics of Agility leads to Agility in Testing

Suja N, Shyam Raja and Suja N
Tata Consultancy Services Ltd.,
Gachibowli, Hyderabad,India
m.shyamraja@tcs.com, suja.n@tcs.com

Abstract:

Agility- The term is defined as “rapid movements in different directions”. Now, Agility in Testing is the adaptation to the rapid changes that the software development undergoes and there by the Software Testing cycle. The different metrics leading to agility in the software development- **RTF (Running, Tested Features)** a project which implements these metrics can become agile.

The metric RTF can bring about agility in the following ways:

RTF requires features to be tested by independent tests and the tests to continue to pass. Therefore the team has comprehensive contact with a customer. The testing should be automated for the continuous testing.

RTF requires that the metric curve grow smoothly. Therefore the design will need to be kept clean.

With **RTF**, more features are better. Therefore, the team will learn how to deliver features in a continuous, improving stream. They'll have to invent agility to do it.

RTF Demands Agility

If an organization decides to measure project progress by Running Tested Features, demanding continuous delivery of real features, verified to work using independent acceptance tests, the RTF metric, all by itself, will tell management almost everything it needs to know about what's going on.

It requires that the whole organization, and particularly the software team, commit itself to learning a new way of working. Agility is demanded by the metric. The metric just requires that the team deliver Running Tested Features, as the real point of software development.

Comparison on how this metric might look on agile project (vs.) a waterfall project- the typical way the software development (inclusive of the software testing and maintenance) works in our project.

Case Study

The case study will highlight our experience with respect to our project in terms of existing SDLC methodology that is used and the scope of improvement with agile methodology.

Keywords:

RTF
EVM
Customer Satisfaction
Quality

Iteration
Xp
Scrum Sheet
Scrum Master
Sprint
TDD
FDD
Delivery Team
Refactor
Product backlog

1. Introduction

In olden days when software development was at its novice levels, we had project teams where everybody played every possible role. A developer performed system testing, a tester developed a core module, and a support analyst performed some development/testing. As projects started failing due to late delivery, cost overruns, and cancelled projects, it led to more clearly defined processes of project management, development, test, and support. Then there were customer surprises, production bugs causing “Quality” to take center stage and software testing emerged as separate discipline. So roles like testers, Test leads, and Test managers evolved and we have organizations focusing on independent software testing services as core business. But the woes of customer though reduced but left him asking for more sophistication in delivery and agility.

The phrase “Agile” was coined in February 2001. A group of proponents of what were then called "lightweight methods" gathered to find out what they had in common. The term "lightweight" was not considered very flattering, so “agile” was chosen instead. The group also produced "The Manifesto for Agile Software Development" (www.agilemanifesto.org) which says-

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

Agile Methodologies required project teams to focus on delivering customer demonstrate-able smaller chunks in small iterative cycles, responsiveness and agility to change in requirements. Hence agile team started surfacing –where in close knit project teams worked based on following three principles:-

- Communication –Avoid busy work – focus on optimum documentation and communication
- Simplicity in process and management – iterative development and daily team meeting
- Agility to customer requests and changes – small iterative cycle with each cycle producing something that delivers some immediate business value.

The various Agile Methodologies are Extreme programming (XP), Scrum, Crystal, ASD (adaptive software development) DSDM (Dynamic system development Method) etc,

The Agile Testing Techniques are – Unit Testing, Exploratory Testing, and Acceptance Testing. The practices involved in Agile Testing are – Conversational test Creation, Coaching Tests, Providing Test Interfaces, Exploratory Training.

The metrics that come into play in agility are –

- The count of running, finished, tested features (RTF)
- The time from a business decision to delivering the result
- Code Complexity
- Total lines of code
- Code coverage by unit and acceptance tests
- Velocity - The number of features finished per week

The practice of agile methodologies as it is known today has emerged as light weight process² is fast generating interest and acceptance as sustainable SDLC model and practice.

2. What is agility?

Agile is a term which embraces a number of lifecycle models, characterized by iterative development and cross-disciplinary working. One facet of agile development is the reduction in compartmentalization of testing within the overall software development lifecycle. A prime directive of agile development is to maintain forward progress.

Agile Manifesto lays down the guidelines for the Agile development-

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

The various agile development Methodologies are-

- Extreme Programming (XP)
- Crystal
- Adaptive Software Development (ASD)
- Scrum
- Feature Driven Development (FDD)
- Dynamic Systems Development Method (DSDM)
- XBreed

3. Characteristics of Agility

Close collaboration between the programmer team and business experts;
Face-to-face communication (as more efficient than written documentation)

- Frequent delivery of new deployable business value
- Tight, self-organizing teams
- Ways to craft the code and the team such that the inevitable requirements change does not become a crisis.

4. Need for Agility - Why Agile Methods for S/W Development

Agile Methods of Software Development came into the scenario due to the following drawbacks of the existing SDLC-

- *Requirements are not clear* – not clear to start with
- *Customer feedback* can be given only by seeing some working functionality which is delivered only at a late stage of SDLC
- *Requirements change* too often – the problem is that Traditional models mandatorily need the requirement to freeze at some point.
- *Decrease customer satisfaction* due to budget overruns delayed deliveries etc.
- Problems with *not getting everything* upfront – big solid design to begin with.

- As per some estimates, project teams build about 30% of the *not needed functionality*. This happens due to lack of customer involvement or lack of getting early and continuous feedback.

So there is quest to invent new models or methodologies that produce software in faster in cost effective manner, satisfying the customer in quality and time to deliver.

5. What is Agile Testing?

Testing practice that follows the agile manifesto is said to be agile testing. Ideally agile testing treats development as the customer of testing. In this light the context-driven manifesto provides a set of principles for agile testing. Testing practice for projects using agile methodologies include:

- Conversational Test Creation
- Coaching Tests
- Providing Test Interfaces
- Exploratory Learning

5.1 Conversational Test Creation

Who should write tests? Defining tests is a key activity that should include programmers and customer representatives. Never do it alone. Testers should facilitate the conversation.

5.2 Coaching Tests

Use Acceptance tests to drive development. Tests should provide goals and guidance and measure the progress. Tests should be specified in a format which is clear for anyone to understand and specific to be executed. The tests help analyze the business.

5.3 Providing Test Interfaces

Programmers provide fixtures that automate coaching tests.

5.4 Exploratory Learning

Plan to explore the product with every iteration. Look for bugs, missing features and opportunities for improvement. We don't understand software until we have used it.

Agile testing is about keeping the bugs out than finding the ones that are in there. The (automated) tests that we write are more about showing that the software works the way we expect than about finding where it breaks. Testing now happens closer to the developer in space and in time. Testers design tests to show that the software works as expected. The developers and testers work together to expand the set of cases that works. The bugs should never get introduced in the first place.

6. Agile Methods (vs.) Existing SDLC Methods

For a project using Existing SDLC methods like waterfall, the conversation might have gone something like this:

developer: Everything works.

tester: C is broken

developer : fixed C

tester : E doesn't work with X

developer: fixed E with X

and so on until the tester can't find any more bugs.

The projects that use Agile Methods of Software Development, the conversation would look more like this:

developer : A is done

tester : A works

developer : B is done

tester : B works

and so on until every case works and is functional.

7. Metrics for the Existing SDLC Methods

Testing is the process of running a system with the intention of finding errors. Testing enhances the integrity of a system by detecting deviations in design and errors in the system. Testing aims at detecting error-prone areas. This helps in the prevention of errors in a system. Testing also adds value to the product by conforming to the user requirements.

8. Metrics for the Agile Methods that lead to Agility in Testing

8.1 Running Tested Features

A good way to measure forward progress is to know where the development effort stands in relation to the functional features desired by business stakeholders. Running Tested Features (RTF) is a way to track progress by measuring how many features pass acceptance tests. Running Tested Features is a metric used to measure the delivery of running, tested features to business clients, and collect their feedback. Suggested by *Ron Jeffries* as a way to increase agility and team productivity by continuously monitoring the software development effort from project inception through delivery, RTF is simple to define.

8.1.1 Definition of RTF:

- The desired software is broken down into named features (requirements, stories) which are part of what it means to deliver the desired system.
- For each named feature, there are one or more automated acceptance tests which, when they work, will show that the feature in question is implemented.
- The RTF metric shows, at every moment in the project, how many features are passing all their acceptance tests.

8.1.2 Continuous RTF

RTF is a continuous measurement and, all things being equal, should grow linearly throughout the project. Iterative development teams tend to timebox their iterations, marking each iteration with a milestone focused on delivering a planned set of features to clients every two to four weeks.

Each feature that is released throughout an iteration planning window is a feature that can be measured and contributes to RTF.

8.1.3 RTF tools

Tools in Open source environment like FitNesse and Selenium integrate well with many build tools, and offer a great foundation upon which to build a robust RTF strategy.

Another metric that we can look upon is:

How many customer-defined features are known, through independently-defined testing, to be working?

8.1.4 RTF Curve

The curve that can be drawn for a metric should be clear. Can the metric be –

defects? Then it should be kept low.

Lines of code? Then, presumably, it should be high.

Number of tests? Increased number of tests.

8.1.5 RTF as a metric-What does that mean?

RTF should increase essentially linearly from day one through to the end of the project. From day one, until the project is finished, what we want to see is a smooth, consistent growth in the number of Running, Tested Features.

- **Running** means that the features are shipped in a single integrated product.
- **Tested** means that the features are continuously passing tests provided by the requirements givers -- the customers in XP parlance.
- **Features** We mean real end-user features; pieces of the customer-given requirements, not techno-features like "Install the Database" or "Get Web Server Running".

Running Tested Features should start showing up in the first week or two of the project, and should be produced in a steady stream from that day forward. Wide swings in the number of features produced (typically downward) are reason to look further into the project to see what is wrong.

8.2 How does this metric-RTF lead the teams in becoming agile.?

RTF causes Agility-

- RTF requires feature count to grow from day one continuously. Therefore the team must focus on features, not design or infrastructure. The team must integrate early. And often
- RTF requires features to be tested by independent tests. Therefore the team has comprehensive contact with a customer.
- RTF requires that tests continue to pass. Therefore the tests will be automated.
- RTF requires that the curve grow smoothly. Therefore the design will need to be kept clean.
- With RTF, more features are better. Therefore, the team will learn how to deliver features in a continuous, improving stream. They'll have to invent agility to do it.

The reason that XP and Scrum and Crystal Clear work is that they demand the regular delivery of real software. That's what RTF is about.

8.3 Understanding the RTF curve-When is it good and when is it bad?

The RTF curve isn't right if it starts flat for a few months. RTF metrics are right when features start coming out right away. Projects that focus on infrastructure first need not apply .

RTF isn't right if it starts fast with features and then slows down. Projects that don't refactor need not apply.

RTF isn't right unless it starts at day one of the project and delivers a consistent number of features right along.

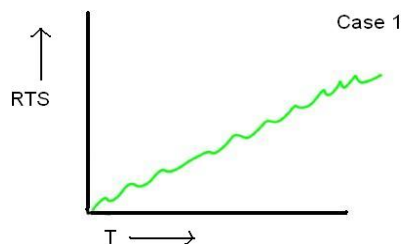


Figure 1: Best Case for a Project using Agile Methodology (Case 1)

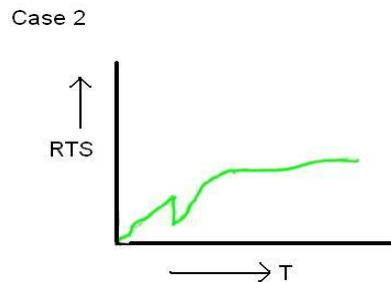


Figure 2: Average Case for a Project using Agile Methodology (Case 2)

9 Agile Testing and Business Benefits

(Source: Nokia) The need to be globally competitive today has made agility a key factor in corporate success. Agile organizations optimize their networks, people, and business processes. Doing so lets them respond to customers more effectively, customise products and services rapidly, exploit opportunities more efficiently, contain threats earlier, and solve problems faster.

Essentially, agility allows firms to combine strength with speed. If agility has become a strategic business goal, mobility provides one means of attaining it.

10 Case Study

Mobile applications nowadays attain a trend of fast growth all over the world. This was a project that we needed to build where in customer had an idea of what they wanted to build and were having problems in visualizing the whole thing. We quickly cobbled a team of experienced designers, developers, project manager and started off what is known as exploration into world of agile. For everybody it was a new experience of building something like this. We all were very committed and excited to make this project a success.

We started off with some high-level design document that outlined core components and suggested architecture. As we started to implement, we started to face design issues as the customer had problems in visualizing the complete scenario. As in agile method we have a customer



representative along with the team, if we too had in one our team, we could have avoided the overhead of the work done which was of no use as there was no architecture support for the same.

Even our project has a high frequency of changes in the requirements. The requirements change on an average on a daily basis.

The content which is shared/streamed using the Mobile Application should be more secured. The strategy and approach taken to testing is critical for the successful delivery of today's complex and integrated systems. The consequences of poor quality are made even more acute as applications become accessible via the Internet and other service-oriented environments, where the presence of defects or poor performance can have serious repercussions for the business. The key to successful software testing is to test what matters. Aligning the testing process tightly to well-defined business requirements of an application assures the most important functionality is being tested. Testers, who are highly qualified and experienced in testing Mobile applications, develop an individual test plan for every project to describe areas and the ways of testing the application on different platforms (Mobile OS), in different networks, in different bandwidths using different devices and tools to make Mobile Testing more Agile. It has been my experience that finding tools for both unit testing and automated acceptance testing is difficult when working on applications for mobile platforms (such as BlackBerry, Nokia, Sony Ericson and windows mobile), written as native apps. To make testing easy we have written tools using .Net framework and MAUI frame work. Test automation tools have significant competitive advantages throughout the development and deployment of mobile applications. It not only provides unrivaled benefits when test scenarios are complex, time-consuming to configure, or frequently repeated, but also becomes crucial when tests must compare devices from different vendors or provide critical 'pass/fail' results. One of the key features of a good test automation system is its ability to provide tests that are fast, repeatable, and simple to customize. In addition, test results should be clearly graphed and consistently presented so they are easy to compare. The discussion below highlights the advantages of using automated test systems and lists the tests you should consider automating when testing the functionality and performance of next-generation mobile applications.

With this kind of a scenario, agile methodology of software development and testing will be highly supportive in achieving goals with respect to all perspectives like business, management, project and individual perspectives too.

11 Citations

- Website's

http://www.agilejournal.com/index2.php?option=com_content&do_pdf=1&id=203
<http://www.xprogramming.com/xpmag/jatRtsMetric.htm>
<http://www.agilealliance.org/show/2047>
<http://agilemanifesto.org/>
http://www.io.com/~wazmo/papers/agile_testing_20021015.pdf
http://www.io.com/~wazmo/papers/agile_testing_20030311.pdf

References

- [1] Test2008: 2008: Conference Proceeding Template.
<http://test2008.in/icsd-template.doc>