



Defining and managing requirements within large and distributed Agile projects

By Martin Crisp, CTO, Blueprint
www.blueprintsys.com
martin.crisp@blueprintsys.com

Abstract:

Agile methods have gained traction in large enterprise organizations as adoption success stories keep growing. Several metrics seem to suggest that Agile projects are completed more quickly and create software with fewer defects than projects built with traditional waterfall methods¹. Yet some studies show that when offshore or distributed teams are involved, the defect rate almost triples².

This paper reviews processes and technologies that can dramatically improve how effectively requirements are managed and defined within large and distributed Agile projects. It is assumed you are already familiar with Agile methodologies.

¹ [Michael Mah \(QSM\)](#) – Agile project are done 25% to 50% faster with roughly half as many defects.

² [Michael Mah \(QSM\)](#) – 2.8 times as many defects with Agile offshore teams than onshore teams.

Introduction

It is challenging to optimize requirements definition and management so that customers get what they need as quickly and cheaply as possible. An emerging category of solutions collectively referred to as “Requirements Lifecycle Acceleration™, or RLA™, is designed to help reduce rework and increase accuracy of requirements.

RLA™ leverages rich requirements definition, simulation, and validation to dramatically accelerate key areas of the software development lifecycle; one of its main goals is providing more accurate requirements sooner in the SDLC.

The challenge: Getting requirements right sooner

We expect software to change because it can³, and because we know that it is very difficult to get requirements right the first time. This change creates complexity, missed deadlines and software that often fails to meet customer needs. “In the last year, 70% of projects failed to meet deadlines, and 50% of projects fail to meet the needs of the business. Eighty percent of the issues stem from poor requirements.”⁴

As a byproduct of software change, requirements go through a lifecycle of maturity and accuracy, regardless of what methodology is used. But this lifecycle often takes much too long before the true requirements are understood.

Requirements specification documents: Not the Answer:

The traditional approach to defining and managing requirements typically results in the creation of very large, complex and hard-to-maintain requirements specification documents. But this approach has led to considerable misunderstandings and misinterpretations of requirements between businesses and their IT departments or Product

³ “Software is called upon to bear the brunt of change because, being that it is made of bits, it can change”, B. Foote and J. Yoder

⁴ According to the Standish Group Chaos Report, 2007

Managers and R&D. As a result, projects must go through a lot of rework and cost/schedule overruns as requirements eventually become mature (i.e. precise, understood and stable) much later in the development process. The root cause is that traditional requirements approaches make it difficult for a customer to interpret these complex, static requirements documents, envision the final software product, and thus validate the requirements.

Moreover, once these requirements specifications are approved, the pace of requirements maturity typically slows down even more because change management processes use a very cumbersome and manual process to maintain large and complex documents.

Agile Methods: help address this issue:

Agile methods are a proven success in cutting down the time needed to understand true customer requirements. They do this by building software more quickly so that customers can see what they are getting and provide feedback. That lets customers refine the end product more quickly and accurately.

Agile prefers working code over documentation, because documentation can create a lot of rework and refactoring if it is managed incorrectly or when the project scope and scale increase; that's especially true if team members spend little to no time defining the requirements to "enough" level of detail.

Agile methods emphasize shrinking documentation and boosting communication, so they are very well suited for small teams that work in the same physical space. When Agile projects become large and teams start to get distributed (e.g. for offshore development) producing "just enough" requirements specifications has a new meaning. No longer can stories simply fit on 3x5 cards to be stuck on the war room wall.

Simulations: help address this issue:

Simulation is another way to understand true customer requirements more quickly, but often it is not precise enough to hand over to developers and testers, so it tends to be augmented with "document"-based requirements. For example, simulations tend not to capture non-functional requirements, core data elements and their relationships, etc.

The difficulty of ensuring the simulation is consistent with these "document"-based requirements leads to misunderstood requirements. It can also be difficult to handle change quickly and accurately in these documents and simulations -- especially with large distributed teams.

Creating test cases from these requirements and then updating them as their documents and simulations change also is a huge undertaking that wastes time, promotes accuracy and undermines requirements validation.

Enhancing Agile to Define and Manage Requirements in the Enterprise:

By iteratively defining and managing requirements using the techniques and technologies outlined below, we can help decrease the amount of time required to identify accurate requirements (consistent with Agile software development within each sprint or iteration).

Add lightweight iterative techniques

Defining requirements happens through an iterative process that continually defines and "elaborates" understanding. To support this process, the solution must let the team initially capture requirements at a simple level of detail and fidelity and then let the teams obtain much more comprehensive detail and fidelity in subsequent versions of the requirement. This must be the case for all aspects of requirements definition, including functional, non-functional, GUI screens, business processes, business rules, data elements etc.

Add Integrated & Collaborative Requirements Repository

Large and/or distributed Agile projects require an integrated and collaborative requirements repository in order to improve communications the accuracy of requirements. But these repositories must do much more than store documents. They need to cohesively "link" or "weave" together requirements information.

For example, use cases requirements should not be separate from simulations or test cases. All of these requirements should be inter-related within the repository. This is much more than just a traceability requirement for impact analysis. Weaving together

these requirements helps ensure accuracy and consistency, and thus reduces errors.

Some projects don't get much use out of traceability reporting for impact analysis⁵. This may or may not be the case for your projects, but here is the point. If requirements information such as uses cases, business rules, GUI Screen, simulations and test cases are automatically linked together through and never treated as separate entities, then this dramatically reduces inconsistencies and thus ambiguities in requirements. For example, if you make a change in a use case this automatically updates the test cases, simulations and any references to the GUI screens, instead of just telling you in a traceability report what would be impacted by a change in the use case.

At Blueprint, we use a Scrum-based Agile methodology to develop our product, which we call "Requirements Center". We do our product development with a distributed onshore and offshore team and we use "Requirements Center" with the Blueprint Definition Server as our integrated requirements definition repository to elicit, elaborate, simulate, accept, document and communicate requirements within each sprint. So yes, we use "Requirements Center" to build "Requirements Center".

Our high-level product development lifecycle shown in figure 1 below has a sprint 0 in which we create the first iteration of our requirements within "Requirements Center" at varying degrees of detail. In subsequent sprints we refine those requirements and with develop the working software. We have found that two-week sprints produce the best results as they keep the team very focused. We don't add any new stories in the last two sprints to the release and we only focus on final refinement and integration testing of the product features.

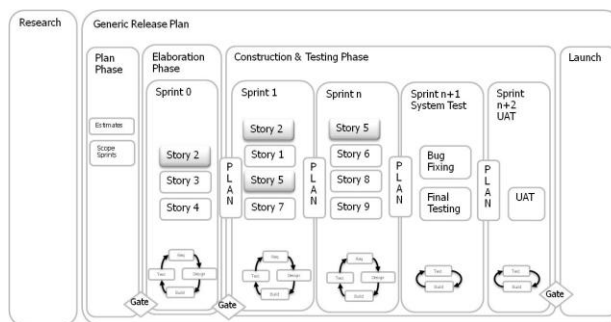


Figure 1

Add Variable Fidelity Simulation

One of the key aspects of validation that leads to acceptance of the requirements is the ability to simulate the functional requirements that the end users will experience in the production system or final product.

Consistent with the rapid and Iterative foundation of Agile, GUI simulations must support different degrees of fidelity from simple hand-drawn wireframes to full blown realistic GUI experiences. Developing just the high fidelity GUI simulations often takes more time than it is worth to communicate what is required⁶. Neither should these simulations not be restricted to GUI screens; in fact, these must include system-to-system interaction as well as the manual steps within the business process.

Add Structured Use Cases

Agile methods typically capture requirements via "stories". Stories are great at providing a quick overview of the key function points and/or a non-structured use case description of a given feature.

However, if you also capture use cases in a lightweight structured manner, then they can be used to drive simulations, generate test cases and permit some level of automatic requirements validation by ensuring no orphan flows, completed branching logic, etc., that non-structured use cases cannot provide. This is an example of "integrated" requirements

⁵ Jim Azar, CTO Orasi (Original developers of Borland® CaliberRM™) "Requirements traceability is best used for validation and verification, it is not very useful for impact analysis."

⁶ Jim Azar, CTO Orasi (Original developers of Borland® CaliberRM™) "I have also found that high fidelity GUI designs lead people to focus on GUI elements rather than functionality too early in the process. The most important requirements that need to be stabilized early in the process have more to do with data and information flow rather than UI details"

mentioned in the “Integrated & Collaborative Requirements Repository” section.

Driving simulations from use cases helps ensure the use cases are validated in a visual manner that a customer can understand. In addition, this automatically removes any manual synchronization between use case definitions and simulations, eliminating inconsistencies and increasing the overall quality of captured requirements.

Add Automatic Test Case Generation

Generating test cases automatically from the defined requirements can add significant value to Agile projects. This results in a significant cost savings by avoiding the need to manually generate test cases from requirements documentation. Equally important, because these test cases are automatically generated they are 100% consistent and in sync with the structure use case and the simulations. That makes the requirements more accurate.

This is especially critical for Agile methods, since some have a test driven development approach that relies on the test cases, in fact, being the requirements. *“Agile methods rely fundamentally on the use of validation (testing), not only as a way of achieving quality by getting rid of errors, but also as a way of identifying requirements.”⁷*

Add Automatic Requirements Specification and Impact analysis Generation

We have seen how manually creating requirement specification documents can be very time consuming, error prone and difficult to validate -- and how they can make it hard to manage change. However, certain requirements -- such as non-functional requirements, key data elements and their relationships -- are difficult to capture via simulations and use cases alone. So we still have a need for requirements documents that go beyond use cases and simulations.

But if your requirements are all defined within an integrated, central, version-controlled repository that has flexible report-generation capabilities, then simply

generating the reports will fulfill the need for requirements specification

And, if this repository keeps track of how requirements elements (e.g., use cases, test cases, GUI designs, etc.) are interdependent, then impact analysis reports that more accurately estimate the size of changes can be automatically generated.

Summary and Conclusion:

Agile methods are showing that they can dramatically improve the time to market and overall quality of software. Yet introducing distributed and larger teams diminishes one of the key success factors of Agile teams: their constant focus on communication that involves the entire team -- including the “customer” being in the same open concept room, daily scrums etc.

Introducing offshore developers to an Agile project for example, does not allow the customer, testers and developers to all be in the same room. The combination of distance, culture and language differences interferes with concise, accurate and timely communication. That results in erroneous requirements and more defective software.

To gain back a significant portion of that lost communications efficiency and accuracy, Agile teams must go beyond video conferencing, conference calls etc. By combining Agile methods with variable-fidelity simulation, structured use cases and automatic generation of test-cases and requirements documentation -- all of it based in an integrated and collaborative requirements repository that automatically links these requirements details -- Agile teams can regain requirements communication efficiencies and accuracy.

These techniques and technologies fall under an emerging category of solutions collectively referred to as “Requirements Lifecycle Acceleration™, or RLA™, designed to help reduce rework and increase the accuracy of requirements.

⁷ [Agile Requirements Definition: A View from Requirements Engineering](#). Armin Eberlein, University of Calgary