



White Paper

Requirement Modeling In Agile Framework

Author: Tabinda Aftab
Research and Development
Application Services

Abstract

Requirements Analysis is the method of understanding the customer needs and expectations from a proposed system or application and is a well-defined stage in the Software Development Life Cycle model.

Given the numerous levels of interaction between users, business processes and devices in global corporations today, there are instantaneous and complex requirements from a single application, from different stages within an organization and outside it as well.

The Process of Software Requirements Analysis covers the complex task of extracting and documenting the requirements of all these users, modeling and examining these requirements and documenting them as a basis for system design.

Requirements Engineering is an emerging field which deals with the systematic handling of requirements that are critical to software project success

Requirements analysis is critical to the success of a development project .Studies shows that insufficient attention to Software Requirements Analysis at the beginning of a project is the most common cause for critically vulnerable projects and the basic task for which they were actually designed, they often do not deliver to that as well. Corporations have spent huge amounts on software projects and the end result eventually does not perform the tasks it was intended for due to lack of concentration in Software Requirement Analysis . Requirements must be actionable, measurable, testable, related to identified business needs or opportunities, and defined to a level of detail sufficient for system design.

Software Requirement analysis is hard for many reasons. Part of it is that software development is a design activity, and thus difficult to plan and cost. Part of it is that the basic materials keep changing quickly. Part of it is that so much depends on which individual people are involved, and individuals are hard to predict and calculate. However, in real world scenarios the users' requirements are not completely known at the initial stages and non functional requirements cannot be easily identifiable at ground zero.

The intangible nature of software also cuts in. It's very tricky to see what value a software feature has until you use it for real. Only when you use an early version of some software do you really begin to realize what features are important and what parts are not.

Agile Development Framework transformed the requirement modeling and analysis like all the other phases of traditional software development. Agile life cycle emphasizes iterative and evolving requirement modeling. These requirements need to be concise, focus and doable in small releases. The following sections describe the Agile requirement analysis concept and processes involved.

Lightweight Requirements Analysis

Agile requirement modeling process is not as document intensive as in the traditional approaches but that doesn't imply that the process is given less importance. In fact the agile requirement modeling process is emphasized and the outputs are improved and better defined with every release.

Agile's approach to requirements gathering is less formal and based around user stories. The goal is to produce a minimal, working version of the software as quickly as possible. Then, based on continual customer feedback the requirements evolve. Emphasis is placed on the features that are really fundamental to the customer. These are then honed and improved in an iterative, refactoring process. This leads to build the system from the core functionality towards the less important features.

Agile Development teams capture requirements at a high level and on a piecemeal basis, just-in-time for each feature to be developed. Agile requirement modeling sometimes also referred to as "Just-In-Time" requirement modeling or "Just Enough" modeling all implying the prioritize, analyze, build and improve process.

Agile requirements are preferably visual and should be hardly adequate, i.e. the absolute minimum required to enable development and testing to proceed with reasonable effectiveness. The basis for this is to minimize the time spent on anything that doesn't actually form part of the final product.

However any requirements captured at the beginning should be captured at a high level and in a visual format, perhaps for instance as a storyboard of the user interface. At this point,

requirements should be understood enough to determine the outline scope of the product and produce high level budgetary approximations and no more.

The timeframe of an Agile Development is fixed, whereas the features are changeable. If during the development of a work release a more important functionality is identified and if the user requests to add that into the current release, the existing scope will be altered to adjust that, unlike the traditional approaches where project teams don't control changes and end up with the dreaded scope creep, one of the most common reasons for software development projects to fail.

Agile teams, by contrast, accept change; in fact they anticipate it. But they overcome changes by fixing the timescales and trading-off features.

It is an analysis process that expects and embraces change and is distinguished from other analysis methodologies in several ways:

- Requirements aren't analyzed or defined until they are needed.
- Only a small initial investment is required at the start.
- Development is allowed to begin with incomplete requirements.
- Analysis and requirements definition is continuous throughout the project.
- Requirements are continuously refined as the project moves forward.
- Change is expected and easy to incorporate into requirements.
- Analysis tasks compliment XP planning.

The Methodology

The agile modeling process is a collaborative process unlike the traditional approaches, where requirement analysis is done independently by business analyst(s) with some involvement of client. The actual development team, the business managers and even the client or stakeholders do not know the actual picture of analysis done by the analysts. The collaborative process in agile brings all the involved from the development team to business/product managers as well as the client to mutually understand the big picture of project/product, current scope, needs and scenarios.

This focus group of technical team(s), business team(s) and stakeholder(s) visualize requirements in interactive whiteboard sessions and create storyboards (sequences of screen shots, visuals, sketches or wireframes) to show roughly how the solution will look and how the user's interaction will flow in the solution. There is no lengthy requirements document or specification unless there is an area of difficulty that really warrants it. Otherwise the storyboards are just annotated and only where necessary.

Requirements are broken down into very small pieces called user stories, similar to use cases. These stories are presented on cards. Thus this is not only very quick, extremely visual and tangible but also saves a lot of time in creating loads of formal documentation. These sessions are very informal and focus on discussing the issues and priorities and the possible limitations or risks. The use of T-card system also allows stories to be moved around easily as and when project priorities are required to be adjusted. These "model storming sessions" are typically spontaneous events; one project team member

will ask another to model with them. Model storming is just in time (JIT) modeling: you identify an issue which you need to resolve, you quickly grab a few team mates who can help you, the group explores the issue, and then everyone continues on as before.

This process can be divided into following steps/phases:

1. Initial Envisioning
 - a. Functionality Analysis
 - b. User Story Analysis
 - c. Architectural Analysis
2. Proof Of Concept Modeling Through TDD
3. Reviews

Requirement Envisioning

The envisioning effort is typically performed during the first week of a project. The purpose of this activity is to broadly define the scope for the work ahead and to specify an initial set of features, functions, and capabilities required for the specified unit of work.

This step involves high-level requirements modeling and high-level architecture modeling. The goal isn't to write detailed specifications, that prove incredibly risky in practice, but instead to explore the requirements and come to an overall strategy for the project. This step assists in identifying the overall strategy for the project.

Feature Set Analysis

Functionality analysis is performed continuously throughout the life cycle of a project. It's purpose is to build User Stories that feed into iteration planning and into individual iteration development. Most of the analysis effort on any project is performed as part of functionality analysis.

User Story Analysis

As part of iteration planning, User Stories are allocated to specific iterations. Story Analysis is then performed as part of each iteration for every Story being developed. The purpose of Story analysis is to finalize the details of each Story in the iteration and to baseline the Story at the completion of the Iteration. User Stories are a easy way of captivating user requirements throughout a project - an alternative to writing lengthy requirements specifications all up-front. A User Story is a simple statement about what a user wants to do with a characteristic of the software; written from a user's viewpoint A User Story should not use technical terminology or state design goals. User Stories should be written in business language that is explicable to all.

Software requirements are a communication problem. There is no perfect solution. User Stories seek to find a balance between written and verbal requirements, relying on collaboration between team members to clarify details near the time of development.

Architectural Modeling

The purpose of the initial architecture modeling effort is to attempt to identify an architecture that has a good chance of functioning. This allows you to set a (hopefully) viable technical direction for your project and to provide ample information to manage your team around your architecture.

Proof of Concept/Prototype Modeling Through TDD

The proof of concept or the prototype generates the executables against the lightweight requirements identified during the model storm sessions.

Agile employs Test-First Designing (TFD) or Test Driven Development (TDD) and refactoring to do the majority of the detailed modeling in the form of executable specifications, often customer tests or development tests. Customer tests, also called agile acceptance tests, can be considered as a form of detailed requirements and developer tests as detailed design. The earlier model storming sessions effort enable to think through larger, cross-entity issues whereas TDD is very focused in modeling and showing issues typically pertinent to a single entity at a time. Refactoring enables to evolve the design via small steps ensuring that the work remains of high quality and the basic functionality intended to implement through that code is not changes or altered.

Reviews

At the completion of each iteration, a review is performed. Scheduled reviews to ensure that the software release is in accordance with the requirements modeled and choose during the session. If the built is successful more requirements can be modeled for next iteration. However, reviews are not always achievable as this session needs to be very focused and requires the audience to fully understand the scope of the built. This allows “lessons learned” from the previous iterations to be included in the analysis process of subsequent iterations. As part of After-Action Analysis, new requirements may be defined. These new requirements may identify new or modified features or they may specify changes to features that have already been implemented. Regardless of the form the changes may take, new requirements are fed back into the process at the appropriate level.

Conclusion

Just-in-Time Requirements analysis significantly reduces project risk and shortens development time. It ensures the most important parts of a system – as defined by the business stakeholders - are being worked on at any given point in time and only defines requirements when they are needed. It supports the evolution of requirements and provides mechanisms for easily incorporating changes into the analysis process. In short, Just-in-Time Requirements Analysis matches the vision and promise of agile model.

References

1. Ambler,S.W.(2008), Agile Model Driven Development (AMDD): The Key to Scaling Agile Software Development.
2. Martin Crisp(2008), Approaches to defining requirements within Agile teams
3. Ambler,S.W.(2008), Just Barely Good Enough" Models and Documents: An Agile Best Practice.
4. Kelly Waters(2007),All About Agile.

