Requirements in Agile

by Joy Beatty and Candase Hokanson

1. Introduction

When companies move to an agile Software Development Lifecycle (SDLC), they often remove the processes and analysis of their waterfall SDLC because, as the Agile Manifesto puts it, "They value individual and interactions over processes and tools." Some of the rigor **should** be removed – waterfall processes can get bogged down with gates and sign-offs. However, caution must be exercised to not go too far against processes and analysis and rely just upon backlogs and user stories. Requirements and the analysis that leads to those requirements are just as essential in an agile project as they are in a waterfall project. The difference lies in how much requirements analysis is completed and the timing of it.

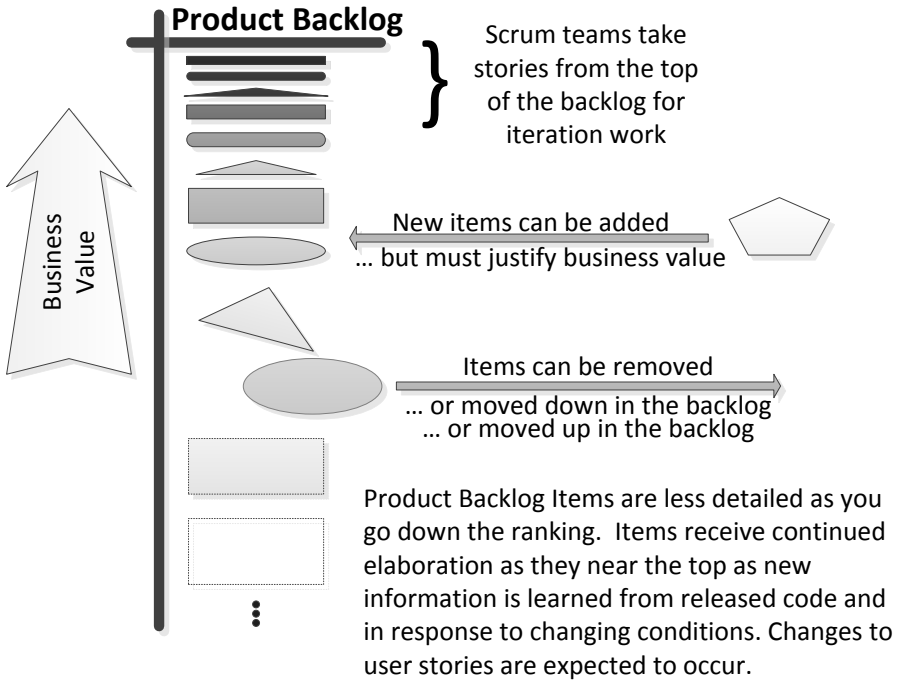
1.1. What is Agile?

Agile is an empirical approach that inspects the facts learned during the course of development and adapts to the facts and other changes. Transparency, honesty, and trust are especially valued in agile, and communications are expected to be open, continuous, and free-flowing.

There are various agile approaches including, Scrum, Kanban, Extreme Programming (XP), Crystal, Dynamic Systems Development Method (DSDM), Feature Driven Development, and Adaptive Software Development. For the purposes of this paper, we will only be talking about Scrum.

1.2. Backlogs

Scrum uses a product backlog, which contains the full list of items that will be worked on to deliver features to the stakeholders. The product backlog is not a complete list on day one of the project, but should encompass the general expected feature set. The Product Owner may add new items, delete items, or change the order of the items as she responds to changing conditions or new information. For example, when the stakeholders receive a released increment of software and begin using it, there may be changes suggested that affect how future features are shaped and developed. The following diagram is an example of how a product backlog changes over time.



Items at the top of the backlog are accepted into an iteration to be created and delivered by the Scrum team – these go into a sprint backlog under the control of the Scrum team. They must be ready to enter a sprint, meaning they have sufficient detail and supporting elements for the Scrum team to immediately take on the work, though further discussion certainly might be required about the items. A general rule of thumb is that the product backlog will have 1 ½ to 2 sprints worth of stories that are fully groomed and ready to be pulled into a sprint backlog.

The product backlog, for an active project, is never static. Items are consistently being groomed to add more detail, such as refinement of the acceptance criteria or updating the effort estimation (in story points). This *elaboration* is an ongoing process.

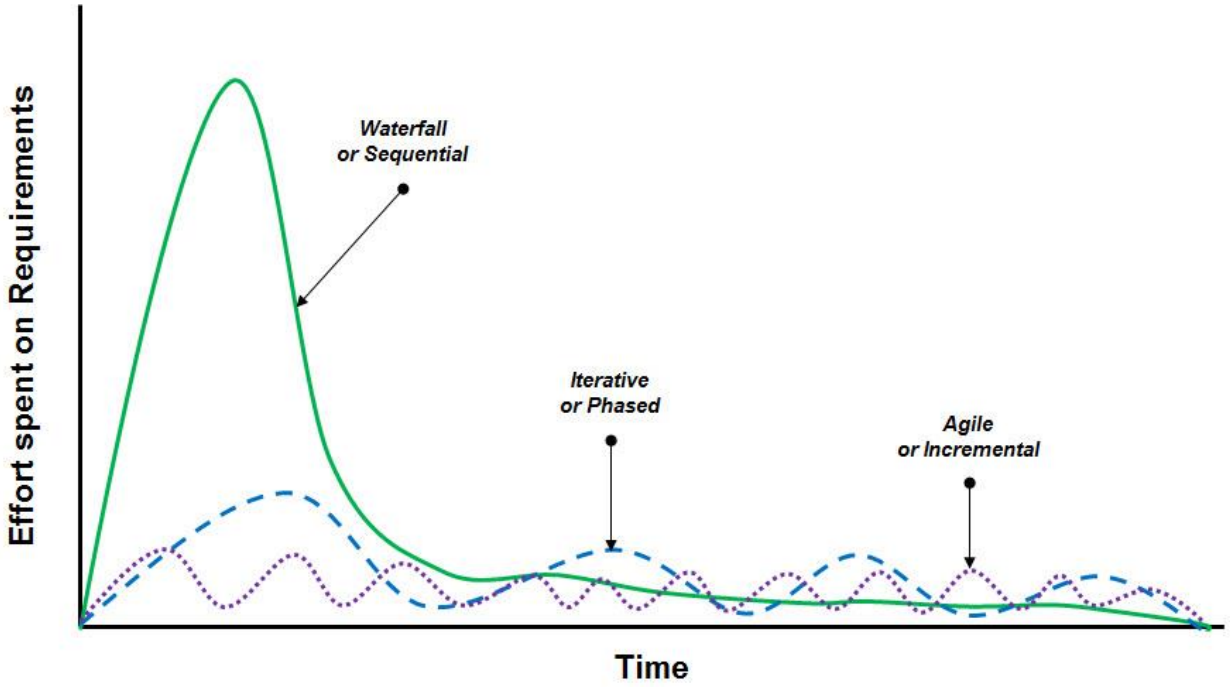
2. Requirements

Requirements tools and techniques do not change much in agile approaches to software development. With that in mind, there are differences in timing and level of detail. Instead of identifying and detailing all of the requirements up front before development begins, the backlog is created with the initial high-level requirements (epics or user stories) representative of the business value to be delivered. The backlog items are further detailed, starting from the top of the backlog and going down.

The result of this is that requirements on agile projects are done just-in-time for a sprint. Also, it is expected that the frequent feedback provided by the business will guide how POs and BAs further prioritize and refine requirements. High risk items can be addressed early in the project, which means the project can react to what is uncovered. If it becomes evident things are just not going to work out, they can “fail fast” and cancel the project early.

However, to identify the most important work or high-risk items to bring into the first sprints, the team must do some initial analysis for the entire product to set the stage of the backlog. This early backlog shows the priority of the main features and lays out the outline for the project work. One idea an organization must get used to is that the backlog can – and **should** – change in response to changing requirements and conditions. The backlog represents the plan at a point in time.

In the classical waterfall approach, the requirements are specified before the first line of code is developed. In an iterative approach, the features are described in less detail at the beginning, but the requirements are constantly being refined as the feature (story) is approaching the sprint in which it will be worked on. The following diagram illustrates when requirements work happens across various development approaches[1].



2.1. User Stories

The most common way requirements on agile projects are captured is through user stories. It isn't the only technique, but it is the foundation for the requirements. A user story has a few common components:

[1] Wiegers, Karl and Joy Beatty, *Software Requirements: Third Edition*, Microsoft Press 2013, p. 47, used with permission

Component	Description
Story	Usually written in “As a …, I want to …, so I can …” form from the perspective of the user of the feature, the story gives the top-level description of what the feature should do.
Acceptance Criteria	These spell out what should be created in order to meet the requirements of the user story. The acceptance criteria form the basis of the “contact” between the Product Owner and the Scrum team as to what will be delivered in an iteration. Testing is based on the acceptance criteria.

User stories might also reference wireframes, architecture documents, process flows, business objective models, ecosystem maps, other visual models and supporting materials.

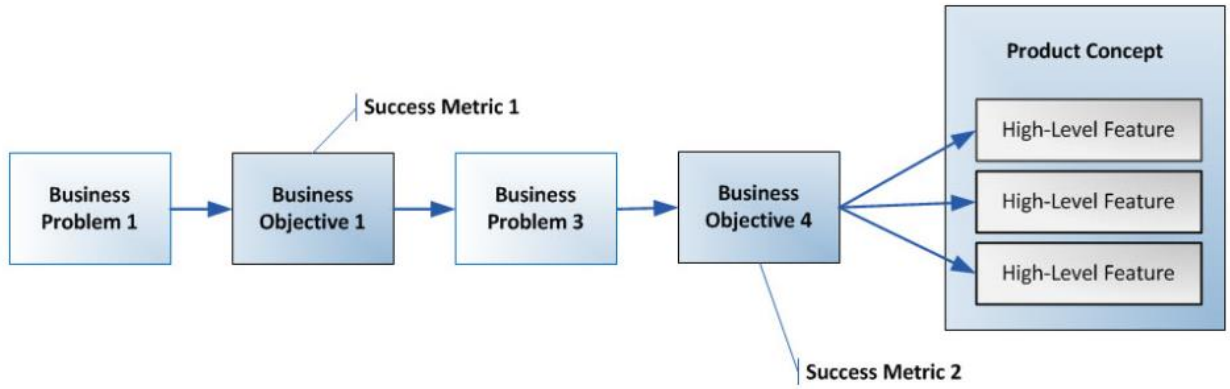
2.2. Visual Models

Requirements Modeling Language (RML®) is a set of techniques for modeling software requirements to organize and communicate large quantities of information, help identify missing requirements, give context to individual details within the overall collection of requirements, and represent different views of requirements details. The models in RML are designed to be easy to create and easy to consume by executives, business stakeholders and technical stakeholders. This article will cover four of the 22 models in RML to show how they apply specifically in an agile setting.

Visual models are especially useful in agile since they convey a lot of information very quickly, and are easier to change than text heavy descriptions.

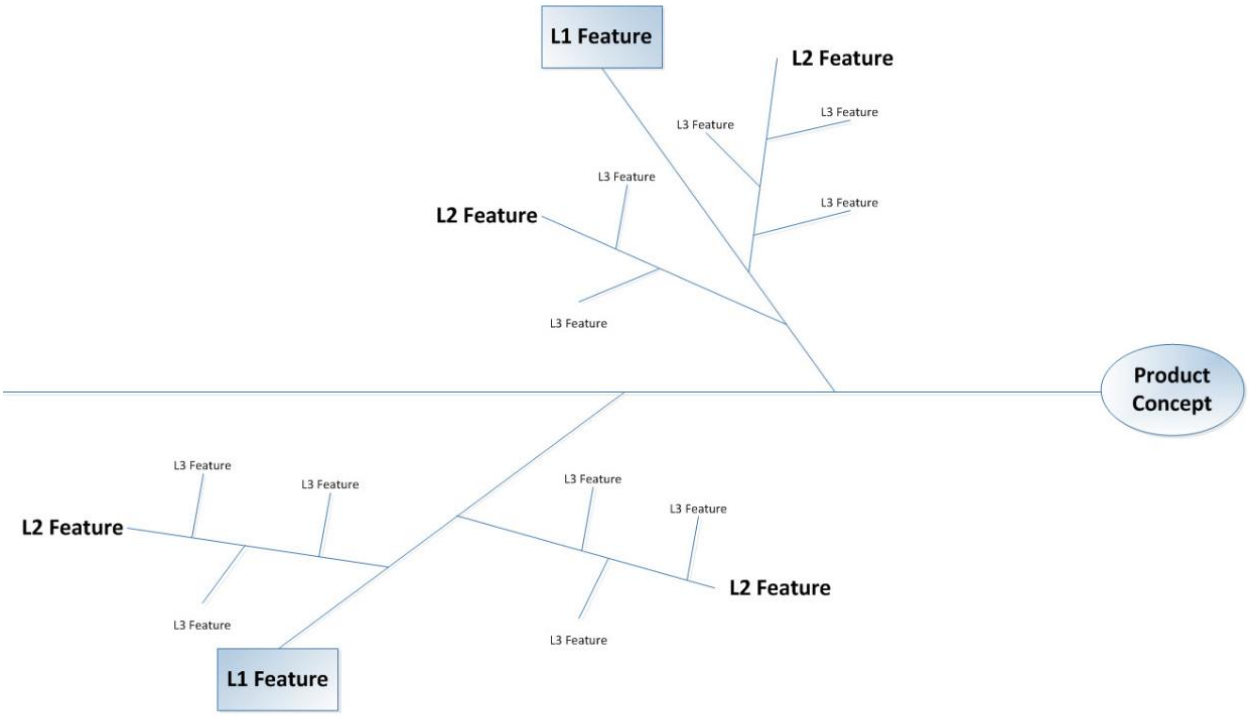
Business Objectives Model

The business objectives model describes the value of a project and is useful in agile to prioritize items in the backlog by focusing on how they contribute to the business objectives.



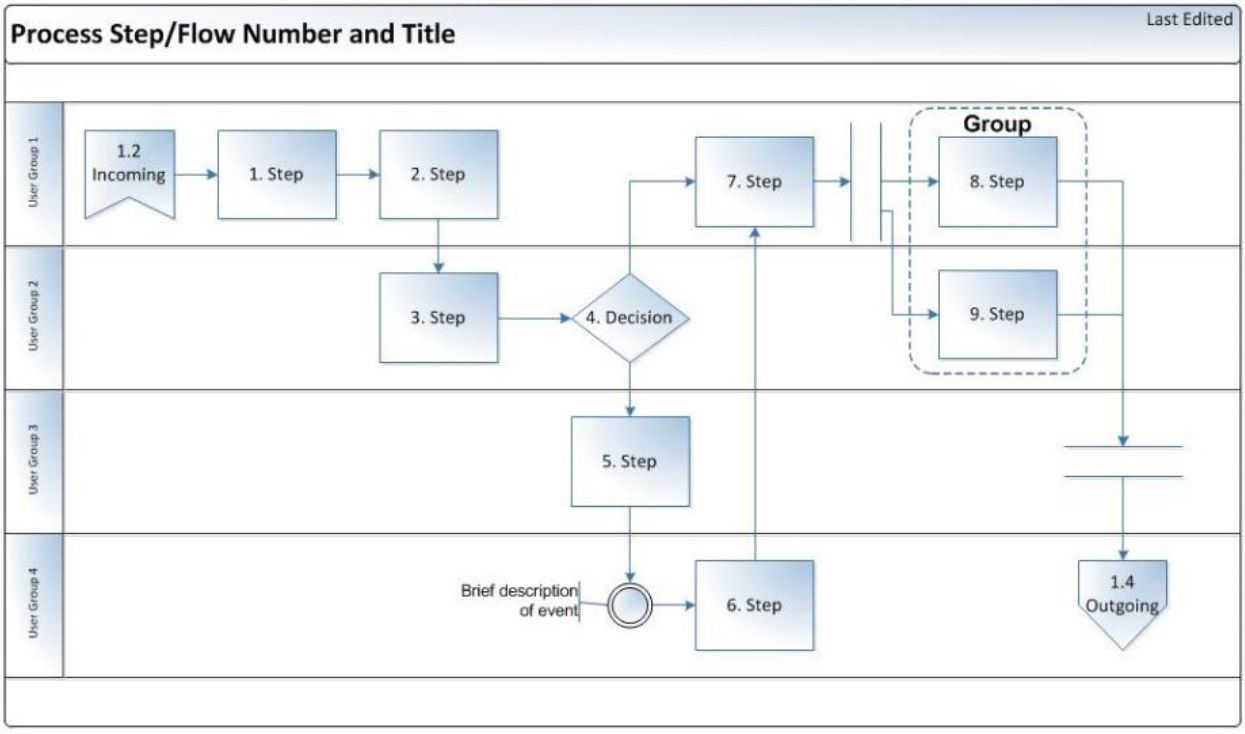
Feature Tree

A feature tree shows all of the features organized into logical groupings. It is used to communicate the full set of features in scope for a project. This model can be quickly updated to reflect the state of the backlog. In fact, sprint-by-sprint functionality can be shown as colors on the feature tree if desired.



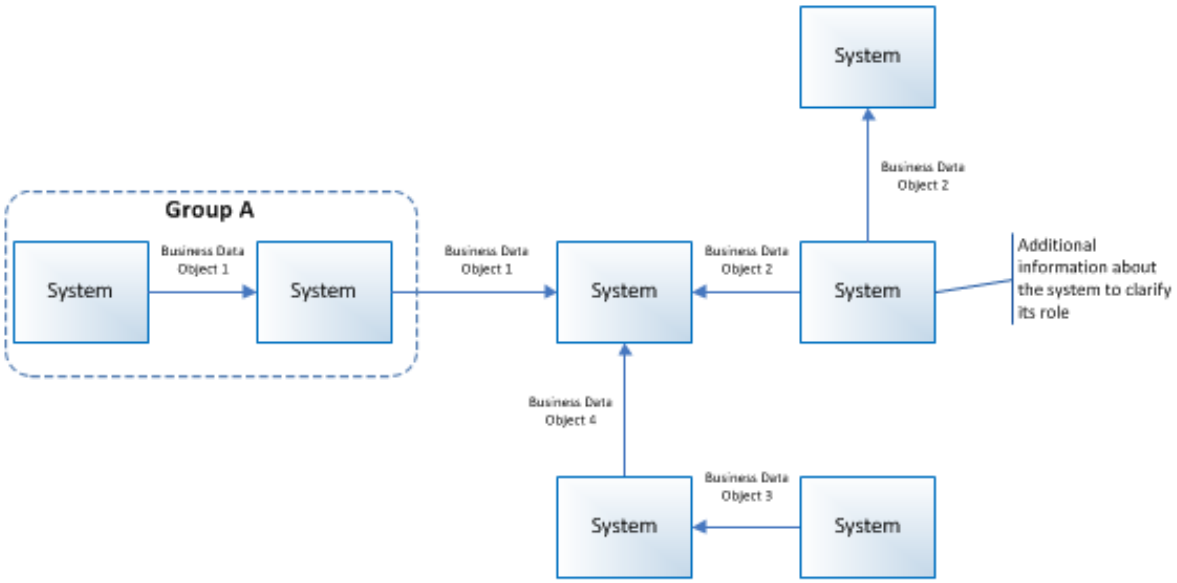
Process Flow

A process flow shows the business process steps people take to perform their job, including the sequence of activities and decisions. These are helpful to look at a full task and then map user stories back to them to ensure no stories are missing.



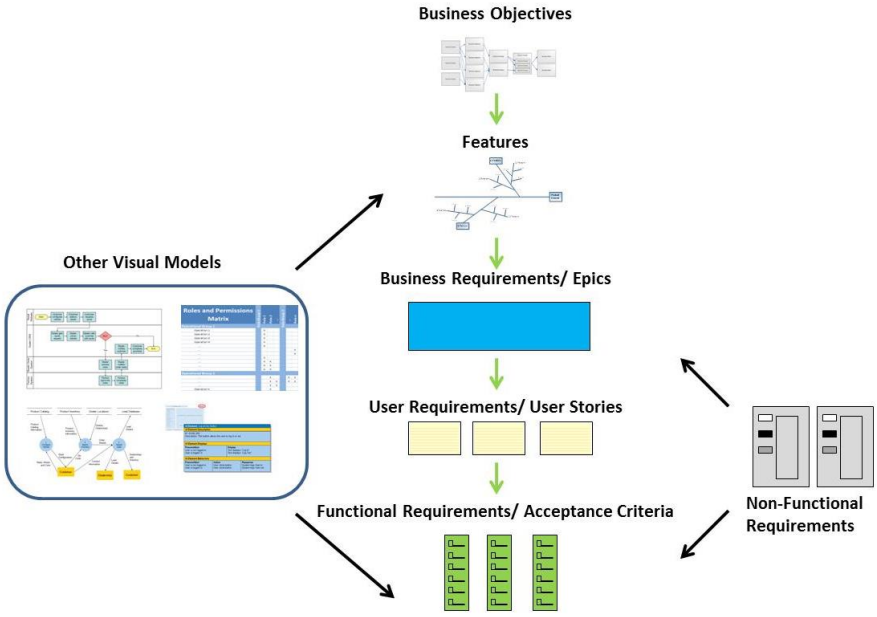
Ecosystem Map

An ecosystem map shows the relationships between all the systems in the solution. It is useful to identify software and hardware relationships as well as high-level data flows. The interfaces in this diagram will help identify things that need to be in the backlog.



2.3. Requirements Architecture

Agile requirements types have a typical hierarchy as part of the overall requirements architecture. Most traditionally is this Epic -> User Story. However, a full requirements hierarchy like the one below can be used on agile projects – which makes the transition between waterfall and agile a little easier.



The hierarchy above shows how some of aforementioned models fit in the hierarchy. Business objectives from your business objectives model form the highest level of requirements. These describe why you are doing the project. Then features from your feature tree (usually L1 or L2 features) tie directly to those business objectives. Epics are the traditional agile epics, and often are the L3 features on your feature tree. Then come user stories and acceptance criteria. We call out acceptance criteria separately from the user story, but it is often a part of the user story. And just like in waterfall projects, non-functional requirements can be sprinkled throughout the hierarchy. Additional supporting models might be required and likely would trace to the functional and non-functional requirements.

2.4. Tools

Requirements management (RM) tools can significantly improve requirements management, but should not impede or hinder elicitation, specification, and development. Given that agile suggests the minimum documentation be created to produce the software, there can be a tension between "just enough" documentation and the tools that manage requirements. The resolution of this tension might be to have an RM tool that integrates with the backlog management tool – such as Rally – and the Business Analyst owns the activities in the RM tool. This allows the Scrum team to primarily focus on using Rally, but still gives the project the overall benefits of having an RM tool.

A few things to think about when using an RM tool for agile requirements:

- **Backlog management:** Does the tool allow the Product Owner to easily change the backlog priorities and manage sprints?
- **Requirements architecture:** Does the tool support the requirements hierarchy for agile that the company has put in place?
- **Integration to backlog management tools:** If the RM tool doesn't do backlog management, does it easily integrate with a tool that does?
- **Models support:** Does the RM tool support modeling requirements and tying them to user stories?
- **Document output:** Especially as teams/companies are moving to agile, does the RM tool support the creation of documents to satisfy the business of ingrained processes that still demand documentation? Often times, a good intermediary step to going agile is to deliver documents that show a "snapshot in time" of the requirements to get sign-off instead of a static document.

One of the most important metrics to monitor is the adoption of the RM tool by all teams, including Scrum teams.

3. Conclusions

In order for agile projects to be successful, they need analysis and requirements to ensure that the agile delivery team is working on the most valuable user stories at any given time. So, with that in mind, a balance must be kept between doing all analysis up-front, bogging down development, having to change a lot too close to a sprint, and leaving everything until the very

last minute. One recommendation is to be working the details of user stories so that there are 1 ½ to 2 sprints worth of user stories ready for sprint entry.

Visual models can help organize information at the product level to give context to the backlog and the user stories in it. They are easily updatable to accommodate the rapid/quick/fast change that is a reality on all projects. They are also vital to help prioritize user stories by tying each to the business value it supports. Additionally, BAs can use a requirements hierarchy and RM tool to manage their analysis work, building on what they did in waterfall.

Requirements and business analysis are necessary on all development projects, regardless of the software development lifecycle they use. In agile projects, requirements are still elicited and specified, but on a different cadence, and sometimes different level of detail, than their waterfall project counterparts.