Chapter 11. **Summary Of The Process**

What are the benefits and costs associated with adopting the process?

In this chapter, I summarize the process and some of the benefits and costs associated with process activities in section 7.2. For each activity, I assess its impacts on an existing Scrum process.

It is assumed that you have already adopted an agile development process that is producing software builds.

Each of these 8 activities is intended to be a stand-alone item. This means that activities may be adopted into an agile process in any order, and that adoption of a new activity does not need to have any impact on activities already adopted by the process.

- **I say 'does not need' to impact, but there are instances where an existing process activity may be modified to make the flow of information more efficient. For example, quality assurance testing may prefer working from model diagrams as a result of a model being introduced into the Elicit Business Needs activity. In which case the documentation of user acceptance criteria in user stories may no longer be necessary.**

11.1 **Overview Of The Process**

The following is a high-level overview of the steps of the Quality with Agile process. The process starts with eliciting business needs and ends when a solution is delivered in a release to the customer that satisfies those needs.

Preconditions:

- The customer has several business needs that could be improved.

Process Steps

1. The product owner identifies the high priority needs of the customer
2. The product owner creates epics to capture the customer needs in the product backlog
3. The business analyst analyzes the highest priority epics to identify potential use cases
4. The business analyst creates user stories in the product backlog that capture requirements from the use cases
5. The business analyst and product owner prioritize those user stories
6. The business analyst details the highest priority user stories for upcoming sprints
7. The product owner approves the user stories for a future sprint
8. The solution architect designs a system architecture from the user stories

9. The business analyst analyzes the system use cases to determine data and user interfaces
10. The UX designer creates screen mockups for the user stories
11. The business analyst adds references to screens and supporting information just in time for sprint planning
12. The development team plans a sprint from the prioritized user stories
13. Quality assurance writes test cases for the user stories in the sprint
14. The development team, business analyst, quality assurance and product owner attend daily standup meetings
15. The development team develops the user stories in the sprint
16. The sprint ends and an incremental build is delivered to quality assurance
17. Quality assurance validates that the build meets the user stories in the sprint
18. The development team conducts a sprint review to demonstrate the incremental build to the product stakeholders
19. The project team holds a sprint retrospective to discuss process improvements
20. The writer documents user instructions for the validated build
21. The deployment manager deploys the validated build to the customer environment and delivers user instructions to the customer

The use case ends.

Postcondition:

- The customer is able to use product features that satisfy their needs.

Figure 73 shows these steps in a sequence diagram.

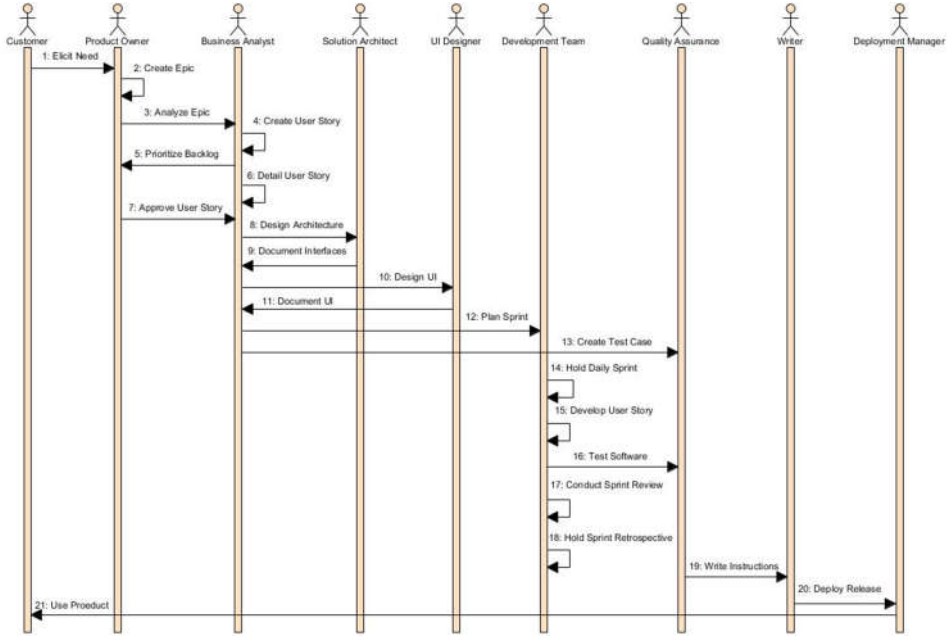

Figure 73 – Quality with Agile Process Flow Sequence Diagram

This is a very simplified list of steps. They describe the happy path through the process activities. In reality, these steps are never processed in sequence. Some steps are repeated multiple times between the pre and post conditions. Some business needs are captured, but never delivered to the customer. (There are too many alternate paths to document. The project team deals with each exception to the process as it arises.)

If the project team decides that a specific role is unnecessary then the steps associated with role may be removed from the process flow. For example, if an independent quality assurance role is removed, the process flow sequence diagram looks like Figure 74.

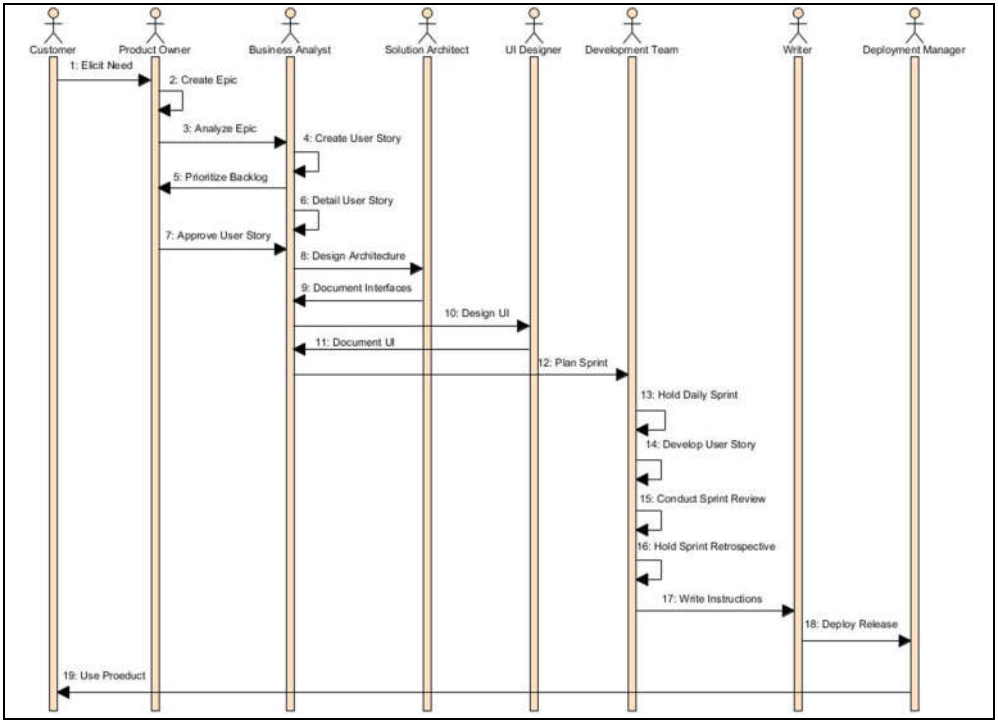

Figure 74 – Process Flow Diagram Without QA

Without a quality assurance role, testing is performed by the development team during development of the user story. User instructions can be written and software deployed once the build is complete.

11.2 **Benefits Of The Process**

The following list contains a summary of some of the benefits that may result from introducing Quality with Agile activities into an existing Scrum process. They are listed in terms of who is the primary recipient of the benefit, how it is provided, what the benefit is and the problem that arises if that benefit is not considered.

Elicit Business Needs

- For the benefit of the customer, business needs are analyzed in order to determine their completeness, which provides a closer fit to the business need, as opposed to simply accepting the customer request as given.
- For the benefit of the business analyst, requirements are structured in such a way that it is possible to trace user stories back to the business need from which they were derived, which provides insight into the extent of a change to the requirements, as opposed to searching for impacted user stories when a change occurs.

- For the benefit of the project team, user stories are created from an analysis of a system feature, which allows a preferred solution to be chosen from multiple options, as opposed to implementing the first proposed option.
- For the benefit of testing, acceptance criteria are derived from well-structured requirements, which allows test cases to accurately test the changes to product, as opposed to writing test cases from the testers best guess of what was supposed to be built.

Groom Backlog

- For the benefit of the project team, items in the product backlog are assessed for their priority, consistency and whether they are still applicable to the product, which means that the product backlog contains an ordered list of essential items leading to less effort expended in searching and less chance of missed priorities, as opposed to searching for the highest priority items when necessary.

Maintain Requirements

- For the benefit of the development team, product dependencies are captured in a model, which determines impacts to the product design due to a change in the requirements, as opposed to implementing the change and using testing to determine what the impacts were to the resulting build.

Design Architecture

- For the benefit of the development team, an independent system architecture is maintained ahead of sprint development, which allows changes to non-functional requirements to be anticipated ahead of time, as opposed to waiting for impacts to architecture change to be assessed from user stories in the next sprint.

Design User Interface

- For the benefit of the project team, user interface design is performed according to standard company guidelines, which means that the user interface that is exposed to the customer looks professional and follows company branding, as opposed to an inconsistent and non-standard user interface being exposed to the customer.

Test Build

- For the benefit of the project team, testing is performed independently from software development, which allows product validation to be performed according to a product release cycle and avoids influence from the development team, as opposed to skipping steps in the testing process due to pressure to get the users stories completed within the sprint deadline.

- For the benefit of the customer, defects are captured prior to release and only make their way into production on approval from someone (such as the product owner), which means that adverse impacts to the customer are minimized, as opposed to exposing defects to the customer without knowing what the impact will be.

Document Release

- For the benefit of all system users, documentation is released with the product, which states what the system actually does, as opposed to allowing the users to figure out how the system has changed by using trial and error.

Deploy Build

- For the benefit of the customer, software is only deployed after it has been validated, which reduces the chance of errors being exposed to the customer, as opposed to deploying an incremental build to the customer that is the result of a sprint.
- For the benefit of the customer, limitations to the product are known and documented prior to release, which means that all customer facing software and documentation is honest and accurate, as oppose to providing functionality to a customer without informing them of its weaknesses.

This list contains a sample of the benefits that could arise from adopting these activities. Whether benefit is realized depends on the project team's unique situation.

11.3 Costs Associated With Adoption

Quality is not free. Additional costs will occur primarily due to people fulfilling the additional roles in the process and the tools they use to support these roles. Also, note that delivery is delayed while the product is validated to ensure it contains sufficient quality for use by a customer. The following costs should be considered when adopting the process:

- Cost of adding an experienced business analyst role to your project team.
- Cost of modeling tool installation and training team members to use the tool.
- Cost of maintaining a model of the system.
- Cost of adding an experienced solution architect role to your project team.
- Cost of adding an independent quality assurance team to your project team.
- Cost of adding a deployment manager role to your project team.
- Cost of installing using and maintaining release management tools.
- Cost of adding a specialized writer role to your project team.
- Cost of installing and configuring document management tools.
- Cost of adding a UI designer role to your team.

- Cost of installing and managing user interface management tools.
- Software is only deployed after it has been validated, which will delay the release of new features to the customer.
- Cost associated with training users to use the process.

♦ **I assume that quality assurance will re-use the same tools that the Scrum development team uses for managing test cases and test results.**

These are the more obvious costs associated with the process. This list is not meant to be exhaustive.

11.4 The Agile Manifesto

One of the most common questions might be 'Is the process agile?' It is not within my expertise to define whether something can be classed as agile or not. In this section I briefly discuss, 'how well the Quality with Agile process complies with the agile manifesto'.

Individuals And Interactions VS Processes And Tools

Several tools are introduced by the Quality with Agile process. The introduction of tools is used to assist with common tasks. For the most part, tools are used to record conversations, not replace them.

Working Software VS Comprehensive Documentation

Working software should be accompanied by quality documentation. User instructions are part of the product. Other documentation should only be encouraged only when it aids a specific role with development of the product.

Customer Collaboration VS Contract Negotiation

Customer collaboration is dependent on accessibility to the subject matter experts. The Quality with Agile process responds to situations where continuous customer interaction is not possible. In my experience, contract negotiation begins prior to the next product release. The customer may change their priorities for the next release up to a user story being taken into a sprint. At which point the functionality captured by that user story is developed for the next release, even if the customer changes their mind.

♦ **After that user story has been developed, a user story is required to remove that functionality from the release.**

Responding To Change VS Following A Plan

The process encourages following a development plan. A plan that is gradually solidified as the product progresses through the process activities. Changes are encouraged during elicitation. Once a user story is taken into a sprint, its lifecycle is defined for the most part. From this point in the process, all features follow a plan for deployment. After the start of a sprint, changes can be

incorporated by implementing user stories that change that plan. During Scrum development, changes do not occur on an ad-hoc basis. Every change is documented in a user story and that user story follows the sprint lifecycle.

Principals

The following list contains the 12 principals for agile development and my response to each:

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

- **It depends on the definition of 'early'. The process delays Scrum software delivery because there are additional steps between the incremental build and a software release. I cannot say if this delay prevents early delivery, but if the product is already delivered on a release cycle then the reason for a delay is because the value of the software is improved.**

2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

- **I don't believe that there is any difference with the way in which changing requirements are handled. Changing requirements are welcome right up to the start of the next sprint.**

3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

- **A release cycle is more likely to be an order of months rather than weeks.**

4. Business people and developers must work together daily throughout the project.

- **The process adapts Scrum to encourage proxies for the business to work closely with developers, when access to the business isn't possible.**

5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

- **The Quality with Agile process identifies many more specialized roles than Scrum. To bring this principal in line with the process, I would reword the statement to say, give motivated and qualified individuals the environment and support they need.**

6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

- **I have no disagreement with this statement. I would add to this statement that every face-to-face conversation should be backed up with a documented understanding of what was agreed to. That documentation may be in the form of textual notes, diagrams or prototypes. If this documentation is not provided, then common understanding from the conversation is lost.**

7. Working software is the primary measure of progress.

- **I heartily agree. ROI is measured against the released software and how well it satisfies the customer. See section Chapter 3 on Quality for more details.**

8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

- **There is no change to the Scrum development process, in this regard. However, developers may find themselves working with proxies for users and sponsors. (I am not sure how a constant pace benefits sponsors and users of the product.)**

9. Continuous attention to technical excellence and good design enhances agility.

- **Agreed, and I might argue that adding specialists to the process encourages technical excellence.**

10. Simplicity–the art of maximizing the amount of work not done–is essential.

- **I would modify this statement to say that maximizing the amount of work not done that provides a negative return on investment – is a goal to aim for.**
- **Any work that provides a positive return on investment is good for the project.**

11. The best architectures, requirements, and designs emerge from self-organizing teams.

- **I am going to disagree to some extent with this statement. Within the development team self-organization sounds like a great idea. However, when it comes to the project as a whole, there are too may egos and personal motivations involved to allow every task to be negotiated. If there are people on the team who specialize in certain technologies, or have specific skills, they should (when available) be assigned those tasks that are appropriate for their experience. For example, assigning a database developer to a task to prioritize business needs, (because they want to pursue a career in management) does not sound like an argument for success. Similarly, I would never expect to see a coding task to be assigned to a product owner. Each role in the Quality with Agile process has a specific expertise that is appropriate for the tasks they are assigned.**

12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

- **This is encouraged by the process.**

Agility is about having the mental attitude to accept changes to your tasks on a daily basis, no matter what role you play.

- As a business analyst - Requirement priorities will change.
- As a solution architect - Technologies will change.
- As a user interface designer - Mockups will be discarded and replaced.

- As a writer - Features will be documented and then removed from the product.
- As a deployment manager - Builds will be invalidated.
- As a tester - Requirements will change.

Everyone on the team needs to be mentally prepared for disappointment and to switch gears to new tasks on a regular basis.

11.5 **Testing**

There are several levels of testing in a traditional lifecycle: unit testing, quality assurance testing and user acceptance testing. Unit testing verifies that the software does what it was designed to. Quality assurance testing validates that the product meets its requirements (or acceptance criteria). User acceptance testing confirms that the business need was met by allowing users to complete a business process.

♦ **Scrum assumes that all testing is performed during the sprint, (or during the sprint review meeting). However, there is no formal testing described during development (or during the incremental build demonstration).**

Why are 3 levels of testing necessary, and why not do user acceptance and validation testing during a sprint as part of development?

Unit Testing

Unit testing is performed during implementation of a user story and this work should be taken into consideration during estimation.

User Acceptance Testing

These are some of the reasons why users may be involved with testing during the sprint:

- Systems end users cannot guarantee their time during the sprint (these people have jobs to do).
- The customer is remote and end users are only available to perform testing on site.
- The system is deployed to a unique customer configuration meaning that user acceptance testing must be performed at the customer premises.
- The complete feature is implemented over many sprints, as such it is not feasible to perform user acceptance testing on an incomplete feature.

♦ **For example, consider the Pay Bill feature has been implemented, but the interface to the payment processing system is not available. Is it sufficient for the end user to test the feature using a dummy payment processing system interface or will the end user be asked to test the software again once the interface is available?**

Validation

Validation not only confirms the implementation of new features, but also that no unexpected functionality is experienced and that non-functional requirements and existing requirements are still met.

To be cost-effective regression and non-functional requirements testing is planned once per release, so ideally validation will only be performed on a build that is a candidate for a release.

In addition, we do not want validation testing to break a sprint if it is not necessary. If a user story cannot be declared done until it has been fully validated and there are dependencies with other user stories, there is no guarantee that errors found as a result of validation can be fixed within the sprint cycle. Rather than removing the user story from the sprint it may be preferable to declare it done, include the user story functionality in the incremental build and raise a defect user story for the next sprint.

Summary

Having quality assurance work outside of the development cycle allows testers to perform their work according to their own agenda within the release cycle.

In the Quality with Agile process unit testing (verification) is performed during the sprint. Quality assurance testing (validation) is performed after the sprint and prior to a release. User acceptance testing is at the discretion of the customer. The customer may want to have their users confirm the acceptance of the product prior to deployment at their site.

11.6 Impacts To The Business Analyst

Finally, I want to list some of the major changes that the business analyst will experience when working on an agile project. The following describes my experience of the differences between working with requirements on a traditional life-cycle versus an agile project.

The skill sets, techniques and deliverables are pretty much the same on any project that delivers a software product to a customer. The main difference is the sequence in which items are delivered.

- **A user story contains the same components and structure as a traditional requirement. I find that user stories are easier to understand because they allow a more flexible format.**

However, work can be much more frustrating for a business analyst on an agile project.

On a traditional waterfall or RUP project, a set of requirements are typically stabilized prior to starting development. The business analyst will typically complete a use case before tackling the next use case in their list of priorities.

This gives the business analyst the satisfaction of seeing a set of requirements for a feature developed through to completion (even if they are never realized in the delivered product).

On an agile project, analysis is performed across several features in parallel. This means that the business analyst may be working on several use cases at the same time. These use cases may be derived from unrelated business needs. The sequence in which requirements are derived is dependent on the priorities that are set by the business (or product owner). These priorities are re-assessed at least every sprint. This means that an epic that the business analyst has detailed and even analyzed into system use case may never be taken in to a sprint as user stories.

I find that up to 50% of the use cases that I create are never delivered. Many epics are shelved never to resurface to the top of the product backlog. User stories that have been created from these epics join them at the bottom of the backlog and any related use cases are archived.

When an epic is prioritized, its child user stories are often delivered over several sprints. The main features of this epic (those that deliver most value) are delivered to the customer first. Subsequent user stories from that epic are often overtaken by higher priority user stories in the next sprint. There are always high priority user stories introduced into every sprint. Low priority user stories that are related to an epic that was started several sprints ago may only be added to sprints as development time permits.

For example, consider a feature that includes a request from a stakeholder that a receipt can be emailed or sent by text message. The higher priority user story is that a receipt is emailed. Sending the receipt by text message is given a lower priority and hence it could be moved to a subsequent sprint. In the meantime, a defect user story comes down the pipeline and is entered into the next sprint, bumping the text message user story out of the sprint. If this continues over several sprints, the text message user story keeps getting pushed down the product backlog. Eventually the business determines that no revenue is being lost because of this missing user story, and it never gets delivered.

The business analyst has already detailed a user story and its acceptance criteria in anticipation that it would be delivered with the original epic. Instead, it remains in the product backlog until removed through backlog grooming.

On a RUP project, this whole feature may have been captured in a single system use case. This system use case would probably have been delivered as a single package and therefore both emailing and texting the receipt are delivered in the same release.

In summary, business analysts may find themselves adjusting their schedule on a daily basis. The daily standup meeting will often introduce new priorities into your schedule. The sprint review and sprint retrospective meetings will also adjust the priority of user stories. The highest priority for the business analyst is to make sure that sprints are fully populated with the highest priority user stories. As time allows the business analyst provides support to other members of the team. The business analyst is the go-to person for information about how to use the product. The business analyst is encouraged to maintain this knowledge by using the system and updating the model to ensure that it captures the actual functionality of the system.