Chapter 3. **Quality**

What is quality and why is it important?

Before diving into the details of the Quality with Agile through Pictures process, in this chapter I explain what I mean by 'quality'.

The process recognizes an independent Quality Assurance team that is responsible for verifying the quality of the product throughout its complete lifecycle. Outside of the sprint development cycle, the QA team is involved in every step of the product lifecycle, from requirements elicitation through to testing of deployed software.

Quality is not testing. If your quality assurance team is only measuring quality against Acceptance criteria, they are involved too late in the process. If quality was not built in prior to development, once the product is built, it is already missing quality. All quality assurance can do is tell you where quality is missing. Quality should be built into the product from day 1. That is why I recommend that quality assurance be invited to observe and understand every stage in the process.

- **Development is not discussed in this book, but QA could be invited to participate in assuring the quality of the code during a sprint.**

Ultimately, I measure quality by understanding how well the product keeps the customer satisfied (happy). Customer satisfaction may be measured by the cost to the customer of using the product. The following traits of a product could be used to deduce cost the customer:

- Does it crash? – Product downtime costs time and money, is extremely frustrating for the user, may require system administration resources to repair, and could cause loss of customer data.

Product downtime is not only caused by planned system maintenance, but anytime a user is prevented from completing an operation. Keeping the system operational is a result of thorough functional testing and load testing. A well-designed architecture is essential for keeping the system operational.

- How well does it satisfy the customer need? – If a user operation could be reduced by a few seconds, the saving to the customer may appear to be insignificant. If that operation is repeated many times, then the savings become significant. Making the product just a little easier for the user is a significant cost saving over the lifetime of a product.

A good UX design reduces to time it takes a user to complete an operation. A good analysis of the use cases can reduce the number of steps it takes a user to complete an operation.

- How much training is required? – Users of the product come and go. Training is not a cost that is spent up front and only necessary when significant changes are made to the product. Training should be considered a continuous expense over the lifetime of the product, and therefore be as intuitive to the user as possible.

A consistent and familiar UX design can reduce time to learn how to use the system. Well-documented user manuals can significantly reduce training costs.

- How much maintenance is needed? – Cost to the customer includes time spent maintaining the product by system administrators. Unlike user downtime, maintenance costs can be scheduled to occur so disruption is minimized, but they could have significant adverse effects if they do not go according to plan.

Minimizing maintenance is a result of a good architecture design and a deployment process that allows the current system to remain operational during maintenance. Ideally, (for the customer) maintenance and software upgrades can be performed without any disruption to the users. For example, deploying to redundant system architecture allows the hardware to be switched over with only minimal interruption of service.

3.1 **The Impact Of Quality**

How do you know that the customer is happy with the product? They will tell you. Happy customers will spend money on your products. Maximizing income from customers is often the purpose for deploying your product in the first place. Even if it is a non-profit system, customer happiness with the product reduces the cost of supporting the system. Less customer interaction with customer support makes both the customer and the supplier happier.

When you understand customer's reactions to your product, you may ask:

- Is the product making money? Measure the ratio of revenue against cost to supply the product to the customer, for each release.

  ♦ **Even if it is a non-profit product, you should be asking how much the product is costing.**

- Could it be making more money? Estimate the cost of changes to the product that encourage customers to increase their investment.

  ♦ **Even if it is a non-profit product, you should be asking could we provide an improved service for the same cost.**

- What does the product need to do to continue making money in the future? Listen to customers and prioritize their needs.

> ♦ **Even if it is a non-profit product, you should be considering how much costs are going to increase in the future.**

- What are the obstacles that could reduce how much money the product makes? Estimate increases in usage over time and plan for an architecture that can support that increase in users.

> ♦ **Even if it is a non-profit product, you should be asking how current costs could be reduced.**

These questions can be addressed by measuring product quality.

Remember that testing can only give us an indication of how much quality was built into the product. Quality assurance starts from day 1 of the product life.

The following quote comes from a description of the Scaled Agile Framework (SAFe) - Inspection does not improve the quality, nor guarantee quality. Inspection is too late. The quality, good or bad, is already in the product. Quality cannot be inspected into a product or service; it must be built into it." —W. Edwards Deming.

A well-known software author once told me, "You cannot inject quality into software." You build quality into your product. If your product does not have enough quality then you dismantle it and rebuild it with quality. (This is often known as re-factoring.)

3.2 **How To Measure Quality**

How well does the product satisfy customer needs?

As a business analyst, I often write requirements that are intended to test the quality that is built into the product. These 'quality' requirements are often known as non-functional requirements (NFR) or supplementary requirements. NFRs do not specify what the product will do; they specify how well the product will do what the product is going to do.

3.2.1 Specify The 'ilities'

Software quality is defined as a field of study and practice that describes the desirable attributes of software products. I describe these attributes using non-functional requirements (or 'ilities'). A comprehensive list of 'ilities' can be found on Wikipedia at https://en.wikipedia.org/wiki/List_of_system_quality_attributes.

If you are able to verify that all of these attributes of the product will still satisfy the customer needs for each release, you can safely declare that you have a

'quality' product. Unfortunately, there are so many of these quality attributes that no-one could test them all. It is up to the development team to assist with determining which of these quality attributes will provide a positive return of the cost of specifying and implementing them.

Some of the more common of these quality attributes are addressed below.

3.2.1.1 Architecture

Architecture is concerned with ensuring that the system can grow.

- Maintainability – How much effort is required it to maintain changes to existing functionality and add new features?
- Supportability – How much will it cost to maintain system performance as more users, more data and more interfaces are added?

3.2.1.2 Operation

Operation is concerned with ensuring that the system meets the customer operational needs.

- Functionality – How well does the system do what it is intended to do?
- Compatibility – How well does the system integrate with other systems that it shares information with?
- Reliability – How safe is my data, when can it be accessed and how easy is it to recover when something goes wrong?

3.2.1.3 Quality Assurance Testing

Quality assurance testing is concerned with ensuring that users are able to do their job in a timely manner.

- Usability – How quickly can the user achieve their goal?
- Performance – How long does it take me to complete an operation?

3.2.1.4 Deployment

Deployment is concerned with installation and access to the system.

- Portability – How difficult is it install, replace, update and adapt to a new environment?
- Security– How secure is my data and how well is the system protected from hostile access?

3.3 **Quality Activities**

These are some of the activities that can improve the quality of the product.

Introducing the following activities into a development life cycle does not necessarily add functionality. Instead, their purpose is to increase product

quality, hence reducing product rework that is the result of trying to inject quality after the product has already been built. (I.e. improving quality reduces defects and code refactoring.)

3.3.1 Elicit Accurate Requirements.

The business analysts body of knowledge (BABOK) describes 3 types of requirements elicitation; collaborative, research and experimentation. This book is concerned with collaborative requirements elicitation. Collaboration is the most common form of requirements elicitation, and concerns communication between the business analyst and representatives of the business.

When eliciting requirements, a stakeholder is very good at telling you what they want. What they are not so good at, is telling you what they need. The job of a BA is to figure out the requirements that will satisfy the business needs. A common method for achieving this is called root cause analysis (or the 5 'whys'). When a stakeholder specifics what they want, figure out 'why do you want this?'

- **Repeatedly asking 'why?' until the root cause is unveiled is likely to cause a person to stop talking to you. Business Analysts are trained to find the root causes while keeping the stakeholder engaged.**

For example, consider a system that displays a restaurant bill to the restaurant customer. A stakeholder of the system states that they want a 'tip' field added to the customer bill. This is a customer want, not a business need. To understand the business need for this new field, the BA will relate the information in this field to a business process. In this case, the business process is to take payment. The business 'need' is to allow the customer to give additional money to staff members. The requirement is to allow the customer to enter a tip in the payment user interface. The design adds this tip field to the customer bill user interface.

The reason for working the stakeholder request back to the business need is that often you will find that this request is either inaccurate or even unnecessary. This is what is I mean by eliciting 'accurate requirements'. Requirements satisfy a business need, and not necessarily what the stakeholder initially says they want.

The business analyst is responsible for ensuring that user stories accurately capture the customer need.

3.3.2 Review Requirements

Reviewing user stories with stakeholders, helps to determine that the requirements match the customer needs.

Reviewing user stories with developers helps to ensure that the impact to the product (as a result of the requirements) is understood.

Do not wait until the sprint planning meeting. Every user story should be prioritized, 'ready for development' and understood by everyone working on the story (including quality assurance), prior to the sprint planning meeting. Reviewers should not only understand the requirement, but they should also understand the business need driving the requirement.

The business analyst ensures that every person who has an interest in the quality of the product is able to review the user stories.

3.3.3 Prototype The Requirements

Prototyping is often the fastest way to get stakeholder feedback on the actual product, by showing a user what the product may look like.

A prototype can be:

- a series of mockup user interface images accompanied by supporting presentation
- a working user interface using dummy data
- or any demonstration of the system requirements that provides the customer with confidence that the system will satisfy their needs

A prototype is not a release to the customer. Its purpose is to encourage the customer to provide more information about their needs.

The business analyst should verify that a prototype is an accurate representation of the user stories.

3.3.4 Review Acceptance Criteria

Acceptance criteria are derived from requirements. They add information to the requirement that is used to test the implementation of the requirement.

Review acceptance criteria in order to confirm that quality assurance and all members of the development team have the same understanding of the acceptance criteria.

Developers should only declare user stories as done when they satisfies the acceptance criteria. Quality assurance ensures the acceptance criteria are met, in order to verify that that this was true.

The business analyst produces acceptance criteria that are unambiguous and accurately specify system functionality.

3.3.5 Review Code, Design And Architecture

There are a number of reasons why the software may change independent of customer needs.

Architecture change – if there is a change to supporting technology, the code may need to be changed in order to comply with those changes. As a minimum, the software will need to be retested after the change is implemented.

Refactoring – as the code base grows, the original design may need to be revised in order to make the software more efficient. Refactoring user stories identify opportunities to improve reliability, performance and maintenance of the code.

Review Code - to ensure that it is up to the standard expected by the development team and to ensure that it is consistent with coding standards.

Any of these activities may cause user stories to be generated without any changes being made to the requirements. Existing acceptance criteria can be used to validate the changes.

The business analyst does not need to be involved in the generation of these types of user story. However, they should ensure that they do not result in changes to system requirements.

3.3.6 Standardize The User Interface

Define the design of the user interface prior to designing the software.

Companies have standards for branding. These standards include logos, colors, fonts and any widgets that are displayed to their customers. User interface design should be defined to comply with these standards and the user experience is consistent company wide.

The business analyst needs to understand these user experience standards. They should ensure that the user experience is consistent with the user interface design when detailing user stories.

3.3.7 Build A Software Architecture That Will Last

Build a foundation to the product that is scalable and modifiable.

Expect the number of system users to grow between each release of the product. Expect new technologies to be introduced as they become available. The system architecture should be designed so that the changes to existing software are minimized as the system grows and evolves over time.

The business analyst should have a sound understanding of the hardware and software components the make up the system architecture. (In the absence of anyone else maintaining architecture diagrams) I recommend that a business

analyst with modeling skills maintains a model of this architecture so that it may be referenced when writing requirements.

3.3.8 Test Units And Interfaces

Test functionality of the software prior to deploying to a customer. This requires a test environment that can simulate real customer data through user interfaces.

Test interfaces to external systems prior to deploying to a customer. This requires the ability to simulate real customer data at system interfaces.

In my experience, testing should not be placed under time-boxed constraints. Testing takes as long as it takes. It is the measurement of quality that determines whether testing is complete and not the time spent on testing. This is why testing tasks should not be performed within a sprint cycle. Within a sprint, there is pressure on the tester to finish within the allotted time. It is almost impossible to estimate how long it will take to test a user story, if you do not know how many bugs it contains. In this process, testing is removed from the sprint and defined to be a separate activity that is complete when an adequate amount of product quality is assured. (In fact, one may argue that testing is never complete. Testing is continuous and the product is ready when testing proves that quality of the product is adequate.)

The business analyst supports testing activities as required by the development team.

3.3.9 Regression Test The Product

Regression testing is used to ensure that existing functionality was not broken during development of an incremental build.

Just because you have agreed with the customer that the delivered features will satisfy their needs does not mean that the final product will perform as the customer expects. When the new feature is integrated into the existing system (both of which are satisfying the customer), you need to confirm that existing functionality is working as expected. In addition, the performance of the system needs to be verified.

In order to perform regression testing you need to know what it was that the system was doing prior to this build.

The business analyst assists regression testing by maintaining a model of the product requirements. Regression tests are performed against these requirements. The model is modified as those requirements change.

3.3.10 Perform User Acceptance Testing

When a user is involved with confirmation of the product, you are validating the system against its business needs. Make sure that a proxy user for the business agrees that the product can be released prior to putting it into production.

The business analyst provides support to user acceptance testing by keeping a record of the steps that the user performs when using the system. Use cases are an appropriate way to model these steps.

3.3.11 Review Stakeholder Documentation

At least one person representing the customer should review the user documentation that is released with the product. Documentation is part of the release, and it is often the last part of the product to be produced. The documentation should be verified that it accurately represents the released product, even if the product does not accurately meet all of its requirements. This is why documentation is written against the product and not the requirements.

The business analyst should review all product documentation to ensure that it corresponds with their understanding of the system.