Global Knowledge®

Expert Reference Series of White Papers

The Agile Business Analyst: How Much Is Enough?

The Agile Business Analyst: How Much Is Enough?

Adam McClellan, Project Management Professional, Six Sigma Green Belt, Certified ScrumMaster

Introduction

By now, you're no doubt familiar with the statements below:

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

-The Agile Manifesto

These words led to a multitude of discussions, leaving business analysts (BAs) often wondering, "What exactly is a BA's role on an Agile team, anyway?" Granted, working software is more important than comprehensive documentation…but just how complete and descriptive does the documentation need to be? And how much detailed analysis are we talking about? A common answer is: "Just enough to get the work done."

"Just enough" is a common refrain in Agile, as in just enough documentation, just enough analysis, just enough oversight. Practically speaking, what does that mean for a BA? The phrase, "just enough" is subjective—how do you flesh it out and make it practicable? Answering that question can be tricky enough within the confines of a team—but how do you pin down what "just enough" means as you scale across an organization?

There's no International Organization for Standardization (ISO) standard at play here, and a healthy dose of independent thinking and individual discretion is advised. But you can start by keeping the following principles in mind:

1. Every project needs some kind of requirements documentation, whether formal or informal.
2. Planning and structure help streamline requirements.
3. "Just enough" applies to communication, too.
4. Look at the other documentation your team is producing.
5. User stories are medium-level requirements. Sometimes that's enough. Other times, it's not.
6. The greater the consequences of failure, the more you need up front analysis and documentation.
7. Greater regulation = more documentation.
8. Higher integration = more documentation.
9. Ask your stakeholders.

Let's dig into these one at a time.

1. Every Project Needs Requirements Documentation

In the first heady days of Agile, there was a lot of talk about no longer needing a specialized BA role. The idea was that everyone on the team would be practicing business analysis—communicating directly with the business, abstracting requirements from stated needs, and working collaboratively to build out a solution that met those requirements beautifully.

In practice, this led to varying degrees of success. Some skilled teams have been able to execute effectively over the long term without having a formal BA on board; others have found that their results improve dramatically when a BA joins the team. But there's also been an unintended side effect: the occasional crowdsourcing of the BA role was in some cases conflated with the idea that requirements documentation itself was unnecessary and obsolete.

If my manifesto-reading skills are as good as I think they are, that was never the point of Agile. Remember, Agile's foundational statement acknowledges the value in good documentation—it just points out the fact that there's less business value in docs than in working software. True enough.

So, requirements do, in fact, have a conceptual home in Agile. Keep in mind that working software is going to be built on some sort of documentation describing an underlying business need. In other words... requirements documentation.

2. Planning and Structure Help Streamline Requirements

Self-organizing teams are not the same thing as an anarcho-syndicalist collective. They have to function within a larger, hierarchical organization that not only sponsors them but gives them the high- and medium-level context that they need in order to deliver solution components successfully.

In Agile, just as in a waterfall project, you want a clear business goal and vision orienting your team. In some cases, Agile teams may be set up and told to "get Agile" in a problem space without being told what exactly that problem space is. In these instances not only is a team likely to flounder, but requirements needs are more likely to increase. Without a clearly defined business direction, your team may start looking for "more requirements" to give it the direction it needs, when requirements aren't the answer at all.

Similarly, you want a clear architectural direction that can govern the design details. The amount of up-front architectural planning may be less in a greenfield development environment than in an established enterprise that's turning to Agile 20 years after the company was founded. But in both cases, the teams need to be guided by a clear articulation of what business purpose the architecture is intended to serve and the principles that will govern it as it evolves.

The upshot is that the structure provided by good upfront planning makes for better requirements as well as better business outcomes

3. "Just Enough" Applies to Communication, Too

The functional purpose of documentation is to facilitate communication between team members, not to give the analysts something to do with their hands, or to eat up storage space on a SharePoint site. In a waterfall project, the active communication load for the BA is reduced somewhat due to the amount of information contained within the requirements document. However, in Agile, requirements documentation is intentionally kept as light as possible.

As a result, the BA has additional responsibilities for communication: either by directly communicating key requirements information to the right stakeholders or ensuring that the right conversations are occurring so that the requirements of the business are clearly understood.

Embrace these responsibilities with effective communication planning at the beginning. Develop a requirements-focused communication plan, including the following:

1. Stakeholder name
2. Stakeholder type
3. Area of responsibility
4. Level of participation in the team
5. Requirements information of interest
6. Preferred communication channels (if known)

Add additional info as you see fit, and then update it as needed. The point here isn't to glory in the creation of a new Excel template, but to be mindful about who's going to care about your requirements information from the start of a project onward (hint: it's not just the people on your team).

Once you've identified the people who care, the natural next step is execution. As the intensity of your team's work picks up, don't lose sight of the information needs you've identified. Communicate actively and mindfully both within and outside the team. Refer to your Excel spreadsheet as needed.

Above all, make sure that you've established the necessary channels for getting your stakeholders the information that they need in a form that they can use. Sometimes that's going to be conversation; other times, it's going to be documentation. However, it's never going to be telepathy.

4. Look at Other Documentation Your Team Produces

Requirements documentation isn't the only material capable of meeting the information needs of your external stakeholders. Design documents, specifications, and test cases may be able to meet their needs as well.

Granted, Agile teams are known to have light documentation as a rule. But each team and each environment is different by definition; and Agile largely embraces that.

So when you're trying to determine what "just enough" is going to mean this time around, start by finding out what other docs your team members are going to be producing. As a rule, assume that hedging terms ("might, maybe, could") are indicators that the document in question won't be produced.

Next, compare your findings to the stakeholders and needs that you've captured in your communication plan. Verify whether any gaps or misalignments exist. If so, take them into your definition of "just enough" and build your requirements model accordingly.

Keep in mind that the variegated forms of documentation that arose over the years in waterfall projects weren't intended as jobs programs for the technically minded. Each document serves a variety of audiences and purposes. Make sure that you're accounting for these needs as you embrace Agile.

5. User Stories Are High-Level Requirements

User stories form the basis of development and work planning in many commonly adopted Agile techniques. A typical user story formulation runs something like, "As a <user role>, I need to <task/feature> so that <goal/business value>."

> *As a customer service representative, I need to be able to view all member transactions for the last 12 months so that I can resolve billing-related inquiries during the interaction with a customer.*
>
> *As a bar manager, I need to view daily and weekly sales by category and specific drink so that I can identify customer patterns and order stock accordingly.*
>
> *As a customer, I need to view unfulfilled orders so that I can track delivery status and determine whether to cancel.*

Broadly speaking, these outline what a given user needs from the system and why they need it. In other words, they cover the classic requirements territory. User stories may not always meet the classic SMART (specific, measurable, achievable, realistic, and time-bound) criteria; however, the same thing can be said of hundreds of thousands of traditional requirements.

The needs in user stories are often further elaborated using acceptance criteria. The practical application of these criteria is to determine whether a given deliverable from the development team will be passed as meeting the business need. However, the criteria also provide further specifics on the needs being voiced by the business user.

The upshot of this is that if you're working on an Agile team that uses user stories, you should treat them as your high-level first pass at requirements. Manage and trace them accordingly. Realize that in some cases, this will be all the requirements documentation you need. Other times that won't be the case.

6. The Greater the Consequences of Failure, the More You Need Upfront Analysis and Documentation

One of the beauties of many Agile approaches is the much quicker, more efficient feedback loop that they establish by releasing working product increments to users on a regular basis.

Inherent in this approach is the idea that the consequences of a poor product are relatively small compared to the efficiency and innovation that Agile sparks. If your loan status tracking page doesn't note the amount of the loan applied for, or your video-chat functionality is counter-intuitive and difficult to use—it may cause your users some frustration and could negatively impact your reputation. But it's not the kind of thing that's likely to sink your enterprise or injure your customers. You just learn from that failure, quickly move to address it, and incorporate the insight.

However, the cost of failure isn't always small. The failure of a transportation product like a car or airplane is literally a life-and death matter. More common to the Agile setting, as an organization's architecture grows and evolves, an ever-increasing number of interdependent pieces come into play, with many of the independencies not immediately apparent. Releasing the wrong system component into this environment can cause consequences far beyond the bounds of the application in question. As a result, the cost of failure increases.

At a certain point, the cost and likelihood of something going wrong outweighs the benefits gained through a leaner, less document-intensive way of working. Give it enough time, and someone, somewhere will come up with a formula to express the idea. But in the meantime, help guide your team and your organization by asking yourself the following questions:

1. If we miss a requirement or our product malfunctions, what is the likelihood of injury or death?
2. If our part of the system degrades or fails, what are the implications for:
    a. Data integrity
    b. Customer or member privacy
    c. Financial reporting
    d. Regulatory compliance
    e. Functioning of other applications within the system
    f. Core business operations

The more dire the answers to these questions, the greater the need for upfront analysis and corresponding documentation. That's not to say that business requirements are necessarily going to be the answer—you may get more bang for your buck from richer design documentation or a more thorough delineation of the system architecture. Just make sure that sufficient analysis and documentation has been done to mitigate the risk to acceptable levels.

7. Greater Regulation = More Documentation

As a rule, regulations love documentation. The Securities and Exchange Commission isn't going to be satisfied to hear that the system is compliant with the Sarbanes-Oxley Act of 2002 (SOX) because the developer had a really productive hallway conversation with your contact in finance. They want to see the written proof. And part of that proof is going to be showing them requirements documentation that demonstrates that the relevant SOX considerations were addressed from the earliest stages of system design.

That isn't to say that Agile can't work in highly regulated environments. Quite the contrary—many organizations in finance, healthcare, government, and other organizations that receive much scrutiny have had considerable success delivering value using an Agile methodology. Meaning, "just enough" requirements documentation in those places is going to be significantly more detailed and thorough than it might be in a startup that's working on functionality to animate digital photos.

Keep in mind, too, that one of the key principles of Agile is encouraging creative thinking. Just because the requirements documentation has to be more thorough doesn't mean that it has to be complete before development begins. Consider documenting the details in parallel to the developers' work (just be sure to communicate throughout and validate when done).

And when in doubt, reach out to your organization's legal and compliance resources. Find out what type of information they need as a way to ensure the appropriate bases are covered.

8. Higher Integration = More Documentation

Unless your team is the only one in your organization, you can't live in your silo. People on other teams working on other products and projects are going to care about what you're building. And the more components that your organization's environment includes—the more other people there are who care.

Targeted conversations with external stakeholders can address your integration needs to some degree. But when you've got dozens of project teams, hundreds of systems, and thousands of impacted internal end users to deal with, conversations don't suffice.

Keep in mind that while a given team may have the need to move forward quickly on their work, stakeholders throughout the organization have their own needs to contend with, such as:

- Other analysts and other teams need to be able to understand requirements and design activities taking place in their primary and secondary problem spaces so they can reconcile and coordinate their work.
- The business needs a sense of what changes are coming their way in a given week, month, and quarter so they can prep their users on how to make the most of them.
- Testers need a clear articulation of what functionality is being developed by the various teams so that they can validate that the component parts work well together as a whole, and so that they can focus their attention appropriately for regression testing.

As the number of moving and conversing parts increases, the need for materials to communicate details more broadly increases as well. That doesn't mean you should document everything. Over documentation can lead people to become overwhelmed with data and begin to look elsewhere for information as a result. So instead, start by focusing on the following:

1. Descriptions of all business capabilities being touched by your team in the current sprint
2. Solution requirements for the following:
    a) The areas of greatest complexity in your own problem space
    b) Any problem space known to be shared with another team
    c) Anything critical to the achievement of the value promised in the business case

Not all of this information will be used every time in every project, but it'll be there for you and your stakeholders when you need it.

9. Ask Your Stakeholders

As you go through sprints with your team, keep in touch with the people who care both inside and outside of your project. Ask them whether they have the information they need to understand what your team is doing and to get their own work done.

Not only are those practices considered good manners, but they will help you target any areas of the requirements model that need to be enhanced.

Be sure to make your questions specific and ask:

1. Do you understand what my team is doing?
2. Is there anything about my team's work that may overlap or conflict with yours?
3. Do you have what you need to test what we're building?
4. Are you confident that what we're building is the right solution for the organization?
5. Is the documentation that we have enough?

Follow up by developing the documents and models needed to resolve any points of ambiguity or incompleteness.

Conclusion

The principles above do not and should not lead to a definitive set of documentation for Agile projects. Not only is that against the whole point of Agile, but that line of thinking doesn't even hold true for waterfall. Documentation needs will always vary across projects and across organizations.

But what these principles should do is to help you introduce greater predictability into your own Agile requirements activities—both individually and across your organization. As you start to apply these ideas and pose these questions from project to project, you'll likely see certain patterns emerge and rhythms form. And as these patterns and rhythms emerge, you'll be able to establish your own set of practices that make Agile work.

We'll close with one of Agile's own principles (one the 12 principles that underlie the Manifesto):

> *Simplicity—the art of maximizing the amount of work not done—is essential.*

A great point—as long as you remember that this describes the work done in aggregate within your organization—not just the work done by your team.

Learn More

Learn more about how you can improve productivity, enhance efficiency, and sharpen your competitive edge through training.

- Business Analysis Essentials
- Requirements Development, Documentation, and Management
- Business Analysis in Agile Projects
- PMI Agile Certified Practitioner (PMI-ACP)® Workshop

Visit **www.globalknowledge.com** or call **1-800-COURSES (1-800-268-7737)** to speak with a Global Knowledge training advisor.

About the Author

Adam McClellan has over a decade of experience as a project professional, including multiple stints as a Business Architect and Business Analyst experience. As a Project Management Professional, Certified ScrumMaster, and Six-Sigma Greenbelt, he brings a combined focus on high-quality solutions and timely delivery to his work and has experienced first-hand the valuable knowledge and enlightening conversations to be had at all levels of an organization.

The philosophy at the heart of Adam's approach to work can be summed as

1. Know the problem you're trying to solve.
2. The solution may be simpler than it looks.
3. If you think every solution is simple, you're fooling yourself.
4. Always keep driving to a solution.

Adam lives with his family in Durham, North Carolina, where he operates his consulting practice, Brightrope.