The Business Analyst role on Agile teams

Over the last couple of years I have met a number of Business Analysts who have been keen to know more about the BA role on Agile software development teams. In fact, I would say there is a genuine hunger in the BA community to know more about how Business Analysis, and the BA role fits into Agile software development.

This is quite natural, Agile software development is changing the face of software development and BAs want to be part of the change while fulfilling their responsibilities to the best of their ability. While there is a lot of literature on the role of Software Developers on Agile teams the same is not true of the BA role. To date the BA role has been more than a little neglected.

The is a draft

This essay is a late draft of a piece intended for publication and will be updated following publication.

May 2010

The name of the role

The first thing that strikes a BA reading about the two best know Agile methods is: "where is the BA?" Extreme Programming (XP) has a Customer role while the even more popular Scrum has a Product Owner role. In fact both roles may be filled by a BA.

Many Agile teams like to work directly with a customer as described in XP. While this certainly can work it is far from guaranteed. For a start, in many corporate environments there is not one *customer* but several stakeholders who may have different, even conflicting, demands on a system

Nor do stakeholder necessarily know what they want. Sometimes a description of needs is actually a description of a solution which may, or may not, meet the underlying need. Sometimes needs are over stated - perhaps innocently or perhaps bargaining tool. And sometimes the end-users who know what the system needs to do don't want the system at all, someone elsewhere in the company - who doesn't necessarily understand what the system needs to do - has decided a new system is necessary, perhaps to create process change.

Experienced BAs can certainly add to this list but there is one more issue that needs highlighting: time. A customer working with an Agile team needs to have the time to take part in planning meetings, to answer questions, assist with test criteria, even with testing itself, and generally be available for the team to reach peak performance. Doing this while continuing with their usual role is a lot to ask.

BAs can help play the role of proxy-customer, representing the customer(s) to the team and helping the team know what is next. In fact, on the first XP team - the Chrysler C3 project - the customer role was filled by two BAs in succession.

While XP appeared in corporate IT, Scrum's roots are in product development. Consequently the Product Owner role has its origins in the Product Manager role found in most successful software product companies. The Product Manager role is a close cousin of the BA role, utilises similar skills of analysis but is also very different.

BAs typically exist within corporate IT departments, or companies supplying corporates with services; Product Managers exist within companies which produce software products, e.g. Microsoft, Oracle, etc. BAs look inside their organization (or their client's organization) to determine need. Conversely Product Managers look outside their organization to existing and potential customers, competitors and the wider market to determine need. These different lead to very different approaches to requirements gathering.

When Scrum is transplanted into the corporate IT setting it is natural for the Product Owner role to be filled by BA. As a rule of thumb, the Product Owner title can be considered as an alias. The alias is used when discussing the composition of a Scrum, or other Agile team. When that team exists within a software product company the alias resolves to a Product Manager, and when in a corporate environment it resolves to a BA.

From big requirements to just in time

It is a mistake to say that Agile development does not need requirements, and it is also a mistake to say that Agile teams forego documentation. However it is true that both differ from what might be the case on more traditional projects.

Every Agile development should have a business case and a goal which will result in the creation of business value. Certainly these should be created before a project starts, and if they don't exist for work in progress it is worth creating them now.

But what Agile projects don't have (or at least don't need) is a big up front shopping-list of requirements. Directed by the goal, requirements and needs emerge over time as the software is created, shown to users and feedback incorporated. Requirements analysis is not a project before the project but part of the same project.

Starting a project with a shopping list of requirements assumes accurate analysis, plentiful up front knowledge of the domain, a stable commercial environment and an unchanging business strategy. Few businesses in the twenty first century meet these criteria.

Instead Agile projects are goal directed with requirements for meeting that goal emerging as work progresses. Over time a shopping list of possible features will emerge - which Scrum calls the product backlog. Many, if not most, of these items will never be implemented. Regular reprioritisation, usually based on business value or risk, aims to deliver the 20% of work which results in 80% of the value.

Teams in transition from traditional to Agile often start off with a shopping-list requirements document because Developers are usually the first to adopt Agile practices. In such situations Product Owners should regard the requirements document as an authoritative source, but not the only source and a source that constantly needs to be revalidated. Maximise the business value delivered requires continual review of what needs doing.

Embedded BAs

One reason Agile team succeed is because they shorten and accelerate the feedback cycle. Automatic tests are run many times a day; demonstrations are shown to users/customers every couple of weeks and retrospectives held regularly. BAs can help shorten the cycles by being part of the development team, breaking down organization and physical barriers so they can communicate directly with developers. Think one team, one goal, one project.

True, the BA may spend a lot of time visiting users, analysing needs and partaking in governance meetings but they also need to be available to the people who are cutting the code. When a developer has a question about how the system should work the BA is the natural go-to person. Delay in getting an answer slows development; worse still, guessing an answer may result in rework.

Nor is it just the Developers the BA works with. Testers too need to be embedded and in many cases the BA will spend more time with Testers discussing acceptance criteria and test scenarios then with Developers.

The days when a Tester would ask a Developer "how should it work?" are at an end. The person the Tester needs to ask is the BA.

Business value

To date Agile methods, and the associated attention, has focused on how it makes Developers more effective. Following closely behind Developer effectiveness have been the changes to project management required to manage the new ways of working.

This is understandable for two reasons. First Agile methods started at the code face where Developers found better ways of working. Second, our industry is desperate for both effective development, and reliable development. Anything that promises such is bound to get attention.

Consequently there has been less attention paid to ensuring business value is delivered. Now we can deliver software reliably attention is turning from the *how* to the *what*. Further more there is a need for this rebalancing to happen now.

Consider for a moment a development team which adopts Agile and consequently doubles productivity. Yes it produces more but actually the marginal value of what it is producing is likely to fall. It is reasonable to assume that when very little was delivered infrequently that which was delivered was very high value. Double the output and it is likely that the extra items delivered, while still valuable, will have a lower value then the first ones delivered.

BAs need to respond to this challenge in three ways. Firstly by ensuring that only work which can demonstrate business value is done. (This is not a license to stop Test Driven Development and Refactoring, these are essential Developer practises that keep the source code malleable and enhance productivity.)

Secondly by ensuring that complementary changes also happen. Research has shown again and again that maximising the value of IT requires complementary changes outside of IT systems: processes need to be changed, staff trained and empowered, organizations flattened, reward structures changed and more.

Third, BAs are the best people to follow up and close the loop on business value: Was the value recognised? Were changes made?

Ultimately measuring business value delivered feeds into another decision the BA needs to be involved in: to continue or not. Agile project don't start with requirements shopping-lists so knowing when they are "complete" isn't as simple as ticking off what has been done. Governance is not based on stage gates or percentage done but on value delivered and the potential to deliver more.

Agile projects should stop when either they can no longer demonstrate a positive return on investment, i.e. the cost of running the team is greater than the business value it can deliver. Or, when other work can demonstrate a higher return on investment and resources can be redeployed.

Its the maturity model, stupid

From the above description it should be obvious that the BA role on an Agile project will change. Rather than being a requirements gatherer who produces a thick document prior to development then moves to the next project the BA is fully engaged for the lifetime of the project, embedded as part of one team, seeking to deliver the highest business value and ensuring the work is guided by the business goal.

If that starts to sound familiar it is because similar ideas lie behind the Business Analysis Maturity Model. As authority increases so does the scope of the work, from point solutions and system improvement at the low end to business improvement and process change at the high end. In so doing the BA role changes from requirements gatherer to internal consultant seeking ways to improve the business.

Summary

The BA role on an Agile team is not always obvious but it does exist, hiding behind an alias like *Product Owner* or *Customer*. The role is essential in allowing the team to meet business goals and maximise business value.

Paradoxically perhaps, on Agile teams the BA can expect to be more involved in the nitty-gritty work with the Developers and Testers, and, to have greater responsibility in representing the business need and deciding how to meet that need.

About the author

Allan Kelly has held just about every job in the software world: system admin, tester, developer, product manager and development manager.

Today Allan is an independent consult helping companies improve the way they develop software and ensuring customers get what they need. He provides training and coaching in Agile and Lean techniques to help teams reach this goal. His focus is on the management of software development work, including business analysis; product, project and change management, and business strategy alignment.

He is the author of "Changing Software Development: Learning to become Agile" (John Wiley & Sons, 2008), numerous journal articles and is currently working on a book of Business Strategy Patterns.

More about Allan at his website: http://www.allankelly.net where a full archive of his writing and presentations can be found. Details of his training and consulting services can be found on: http://www.softwarestrategy.co.uk.