"I'm a Business Analyst get me out of here" -

Or,

How Business Analysts fit into an Agile development team

Over the last couple of years I have met a number of Business Analysts (BAs) who have been keen to know more about the BA role on Agile software development teams. There is a genuine desire 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.

This essay will attempt to answer two questions:

What is the role of a BA on an Agile software team? and,

How does the BA role change between traditional (so called "waterfall") development and Agile?

In the process I will also explain why there is more work for a BA to do on an Agile team than on a traditional team.

Naming the role

The first thing to point out is that Business Analysts do not exist in every organization. While they are common in corporate IT departments and external service providers (ESPs) they are usually absent from product development organizations. Instead of BAs companies like Adobe, Oracle and Autodesk have *Product Managers* or *Technical Product Managers*.

The Product Manager role is a first cousin of the BA and like the BA they are concerned with determining the needs the software is attempting to satisfy. Product Managers employ many of the same tools as the BA and at the heart of both roles is: analysis.

However the roles are different in one important aspect: while the BA role is focused inwardly (within the corporation, within the department or within the client company) the Product Manager role is externally focused. Figure 1 illustrates the general position.

For a BA the end product, that is the software, will be used by the people within the organization who have little choice over what they use. Conversely, Product Managers deals with customers who have a choice: they can buy the software or not, they can buy elsewhere.

This distinction is important to Agile working for two reasons. Firstly, as we will show when discussing changing to the BA role; BAs needs need develop the same commercial awareness as Product Managers.

Secondly, the most popular Agile method, Scrum, contains a role called *Product Owner* which is largely based on the Product Manager role. Thus it is easier for Product Managers to understand their role on an Agile team than it is for BAs to.

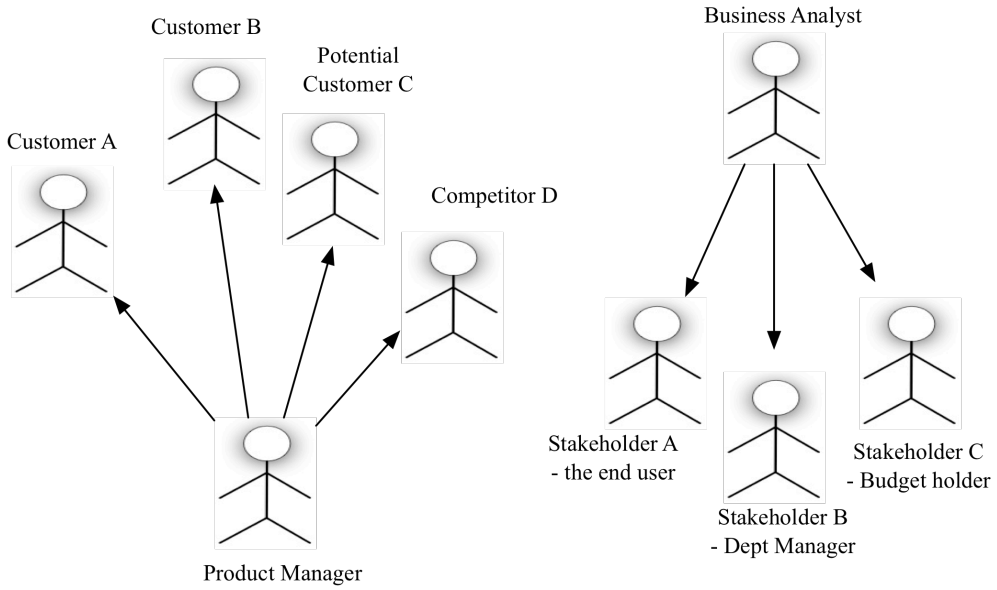

Figure 1 - Product Manager look outward; BAs look inward to stakeholders

Part of the confusion surrounding the BA role on Agile teams stems from the fact that neither Scrum, nor the other widely known Agile method, Extreme Programming (XP) describe a BA role. Both Scrum and XP place great emphasis on development teams working as closely as possible with someone who actually wants the final product. XP calls this person the "customer" and Scrum, as already mentioned, calls it the "product owner."

In fact, the first XP project (the Chrysler C3 development) had a BA filling the customer role. Although the books and descriptions of the project call this role a "customer" the two people who filled the role were BAs.

It is not always practical, or desirable, to have an actual customer work with a development team. Instead a proxy needs to play that role. Indeed, it is wrong to assume there is a single customer. If there is to be a single customer voice someone needs to amalgamate multiple voices.

This is where the BA fits in. The BA is a customer proxy. The BA is the person who listens to multiple "customers" and speaks with a single voice to the team.

One can think of the term Product Owner as an alias used in Agile literature to mean "the person who represents the customers needs." In a software product company there is a Product Manager behind the alias, and in a corporate IT environment there is a BA behind the alias. (Unless otherwise stated from here on I assume the Product Owner role is filled by a BA.)

Agile is not a management free zone

To date most Agile methods have largely been developer centric. As a result

the BA role has been underplayed. Adding to this neglect is the belief in some places that Agile does not require management. And since the BA role is normally a non-coding role it is perceived as a management role by many developers.

The management free message contained in some Agile texts, and espoused by some Agile advocates, comes from a belief that developers know best and self-organizing teams are the most productive way to work. While there is some truth in both arguments there is still a need for management. Self-organizing teams do not come into being spontaneously.

Even on a self-organizing team it helps if someone is focused on the question of what the customer wants, their needs and where the greatest value is to be found. It makes sense for someone with business analyst's skills and background to take on this role. While they may take on some other work within the team (e.g. administration, testing) one should not fall into the trap that other work necessarily includes coding. Not everyone on the team will posses the necessary skills to write Java or Python. Indeed, not having these skills can be a useful barrier to prevent the "all hands to the pump, we need more code!" mentality taking hold. Such a mentality drives out other work in a blaze of naivety.

Historically Agile has underplayed this role. While XP describes what developers do in detail and Scrum describes how the team works together to achieve the project no popular method describes the how the Product Owner fills their role.

To put is another way: think of an Agile team members as actors performing a play. There is a BA playing the role of Product Owner or Customer. The script (Scrum, XP or some other method) describes what to do when on the stage: work with the team, prioritise, etc. But it doesn't describe what the actor does off stage.

It is the offstage work that allows the Product Owner to fulfil their onstage duties. While the onstage role is similar for a Product Manager or a BA playing the Product Owner, the offstage role is very different. To date Agile methods have had little to say about these activities, leaving BA to decide for themselves what needs doing.

The problem with Customers

On the face of it the original XP model seems ideal: find out what you want from an actual customer. However there are a number of reasons why a customer proxy might be better than an actual customer.

Firstly customers may tend to see what is in front of them immediately rather than looking at the longer term or strategic objectives. In an iterative development model this may result in a stream of requests to change the appearance of the software, or add minor features rather than driving towards the ultimate goal. While useful, even valuable, this can lead to small changes, perhaps only cosmetic, which do not justify the cost of the work.

This is particularly true when the development is intended to introduce change into an organization. IT systems are rarely developed and deployed

to support the status quo, instead they are created to improve processes, reduce costs, enter new markets and so on. Using an actual customer, or end user, may focus efforts on the current environment and not on change.

There is a balance to be struck here. The message of Agile methods, incremental change and continual improvement, is as applicable in the wider business environment as it is in the software development one. However there needs to be a change driver.

There is another, more commonly cited, problem with using actual customers to source requirements: *time*. Customers have their own jobs to do, they may not want to leave their job to work with a development team or the company may not be prepared to spare them for the task. This is particularly true in development teams that work with highly valuable, highly paid, individuals like financial traders or doctors. Getting access to such experts is nearly impossible.

All of this assumes there is a single customer who can be identified and who knows what is required. While this is true in some cases it is far from universal. Many internal projects struggle with multiple customers and stakeholders, each with different expectations and needs for the development. At best the stakeholders needs are complementary and combined will produce a better system. At worst these needs and stakeholders are in competition and not compatible.

Given the focus on delivering business need there is a real need to evaluate requests, compare them and turn some down. Where one of these customer stakeholder to be elevated to the position of "the customer" there is a danger that their own priorities may take precedent over other regardless of the business value.

Frequently there is not one single customer but several end-users, and if we further expand the list of those interested in the work to stakeholder the possibility of alternative needs conflicting becomes an almost certainty. Someone needs to decide what is most important, what compromises are possible, and what doesn't get done.

Taken together it becomes clear that the single actual customer model is too simplistic for many environments. The net result is that it is often better to employ a proxy in the customer role than an actual customer.

In short there is more to fulfilling the Product Owner role than knowing what one wants oneself. Someone needs to understand the deeper needs, the motivation and goals of undertaking the work, who he interested parties are and what their interests are.

Just like the original

I was recently involved with a project which had been started with the brief "Make it exactly like the original." Obviously there was no need for a BA here, all the development team had to do was copy a working product.

The original was built using PowerBuilder and while it was operating fine the client didn't want to risk of using being dependent on a dated product. So a project was started to re-write it as a web based Java application.

The users were denied the option of changes that would benefit their work and consequently demanded "exactly the same." The developers were told to copy the original, as a result they produced a web application that looked and felt like a PowerBuilder one.

With no BA involved, and no sense of business value, this potentially win-win turned into a lose-lose. Rather than have the improved application they wanted the users were given an application which cost more to build than it needed to, probably with more features than they needed.

When you consider that as much as 80% of the features in bespoke software are not used then in copying the original software, the development team probably did five times more work than was actually required.

The Problem(s) a BA can address

Many Agile adopters believe the best way to produce software a customer wants is to listen directly to the customer. This can work; this approach should not be dismissed, particularly if the customer is the person paying the bills.

However this approach is not guaranteed to bring success. There are a number of reasons why this approach either will not operate or will not result in the desired outcome. Consider for a moment a corporate situation where one set of "users" will use the software created, but another set of customers, "managers", are commissioning the software with view to changing how the users work, and a third set of customers are paying the bills. Who is the customer for the team?

Such situations were competing "stakeholders" have different needs and desired outcomes for work is the bread-and-butter of business analysis and systems engineering. Of course the development team may nominate one of their own to understand these issues and make sense of the competing demands but in doing so they have created a business analysis role.

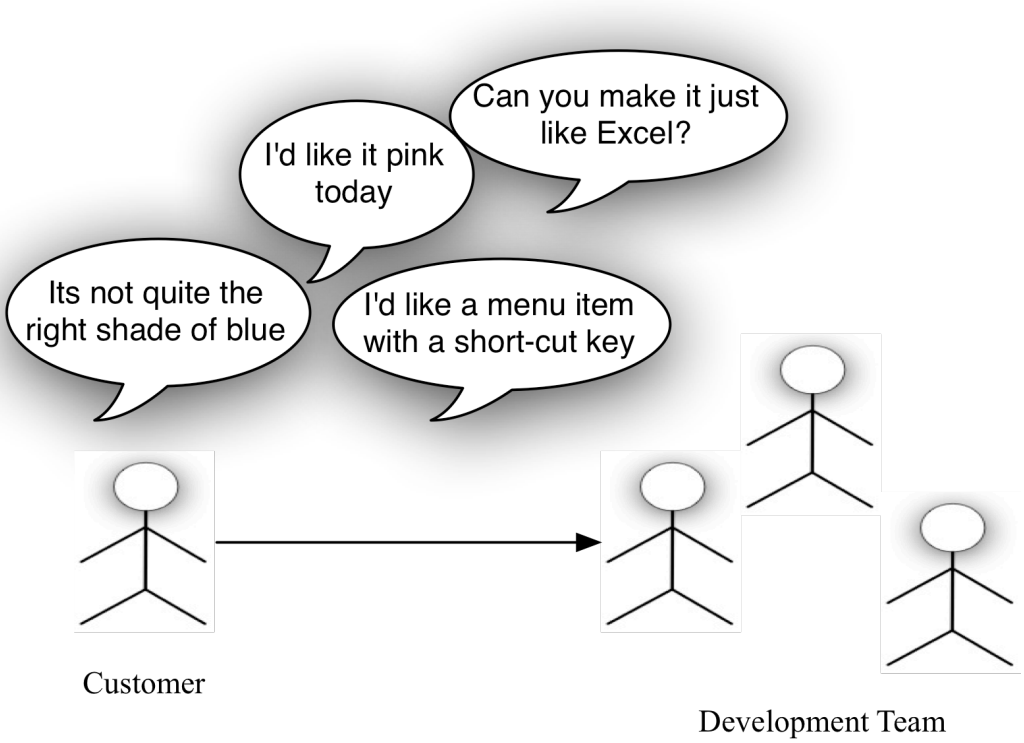

Figure 2 - Customer doesn't have a clear understanding of what they want

When this happens it is fair to ask: *does the person filling the BA role have the right skills and experience?* And: *Is this an effective use of the skills and experience they do have?* It seldom makes sense to stop a brilliant coder from coding and ask them to perform a role they are less effective at.

In order to understand where the BA fits on an Agile team it is worth considering a number of these scenarios that can, and do occur.

Figure 2 illustrates the position in which the customer doesn't know what they want, perhaps they ask for trivial changes or oscillate between different requests. However, there is a more subtle but severe issue here. Namely the customer does not know what will result in the greatest value. Asking a development team to "make it like" is a common way of describing a need but results in zero value to the business. If the customer wants a product "like Excel" then why not have Excel?

Two problems will occur again and again: *what should the team develop?* And *which requests will maximise business value?*

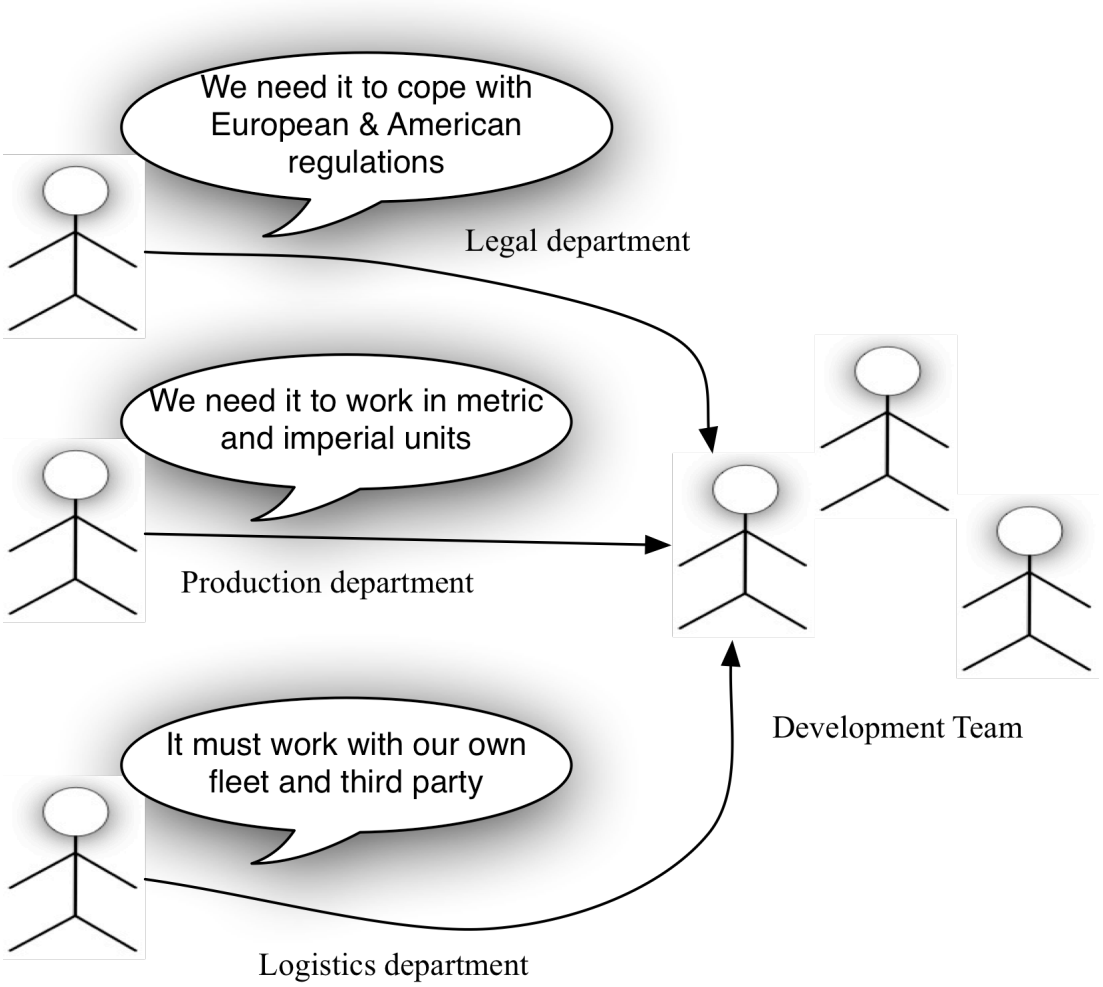

Figure 3 - Multiple customers, multiple needs, multiple expectations

Another common issue is demonstrated in Figure 3, that where there are multiple different "customers" for a system each of whom has different requirements on the system and different expectations. Determining which requests will result in the greatest business value is even more complicated in these cases as each customer group will argue their own side.

Complicating matters further is the small matter of business strategy and company objectives. On occasions it may not be possible to demonstrate highest value for a particular feature but business strategy demands the feature.

For example, take the case illustrated; the software may be under development for a European company with no American operations. On the face of it there is no need to have the software work within US regulations. However, the board of the company may be looking to sell the company within the next two years. If they can interest American investors they may be able to extract a higher price from the buyers even if the final buyer is European.

This example starts to illuminate the difficulty in determining business value for IT developments. Research shows there is commonly a time lag between investment in IT and value returned (Brynjolfsson, 2009). Nor is this the

only complicating factor. Without new processes, training and other changes the potential value delivered by an IT system may remain unrecognised. Consequently while a business unit may be able to demonstrate significant business value from new IT this value might not be recognisable if the unit concerned cannot make other changes.

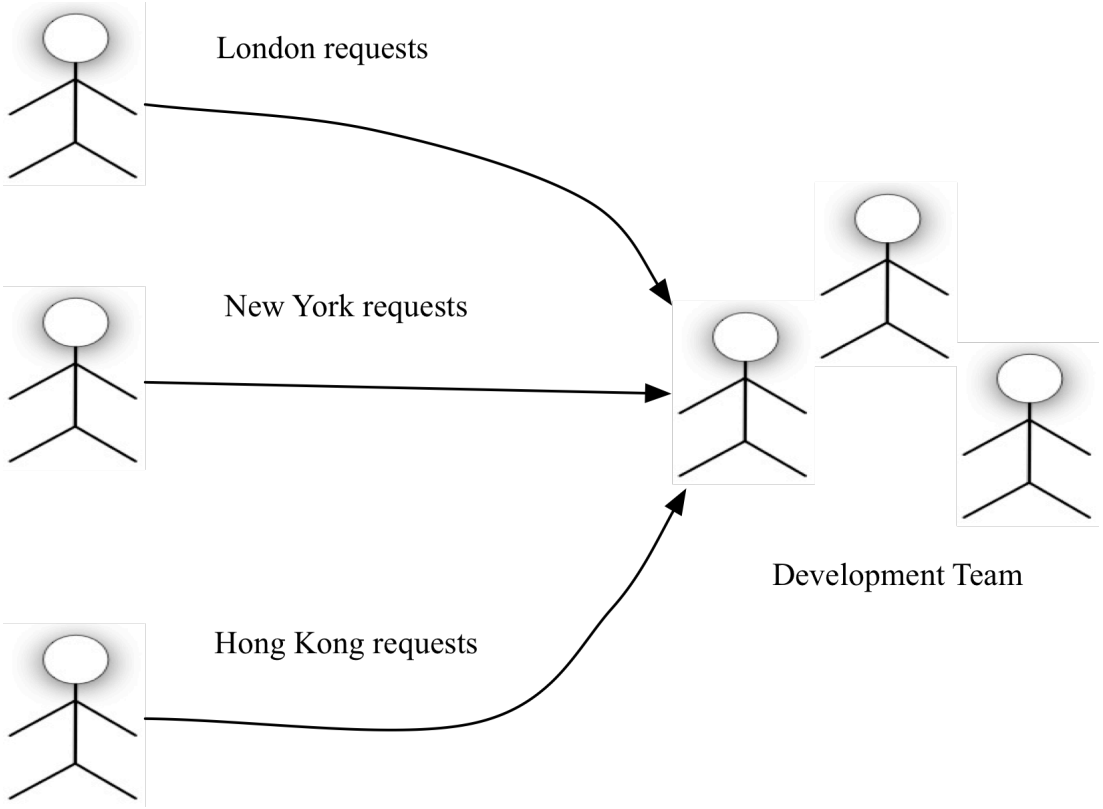

Figure 4 - Geographic differences

Globalisation means that development teams are increasingly faced not only with multiple customers but also with customers spread geographically. Determining which location gets what it wants, and which should be made to wait is no small matter. When a team is based in one location, say, London, requests from that location may win out over location, say, Hong Kong, not through logic but through acquaintance. The team and their customer eat in the same places, drink beer together and share the same space.

Not only do different geographic locations introduce subtle preferences and information asymmetries, they can slow down work. In a perfect world a question asked by the London development team could be answered by New York and Hong Kong over night and the answer be waiting for the London team the when they arrive for work. Yet experience shows the reverse, increased distance between team members and customers results in slower communication.

The need to arrange conference calls, video and online meetings means spontaneous and informal meetings cannot occur. A question sent from London to Hong Kong on Monday afternoon (GMT) that is not answered by Hong Kong during their Tuesday cannot be chased until Tuesday in London, which may mean the answer does not appear until Wednesday.

The answer to all the problems outlined so far is to introduce a new role between the customers and developers - the age old "add another layer of indirection" solution. This role gathers the requests, helps the requester with their logic, examines or creates the business case and value statement, and generally keeps the request pipeline ordered.

Which raises the question: *who should fill this role?* Often this person is some sort of manager. Project Managers are a popular choice for this role because they are professional organizers. But a close look at the skills required, and responsibilities inherent in this role suggests a Business Analyst would be more suitable - shown in Figure 5.

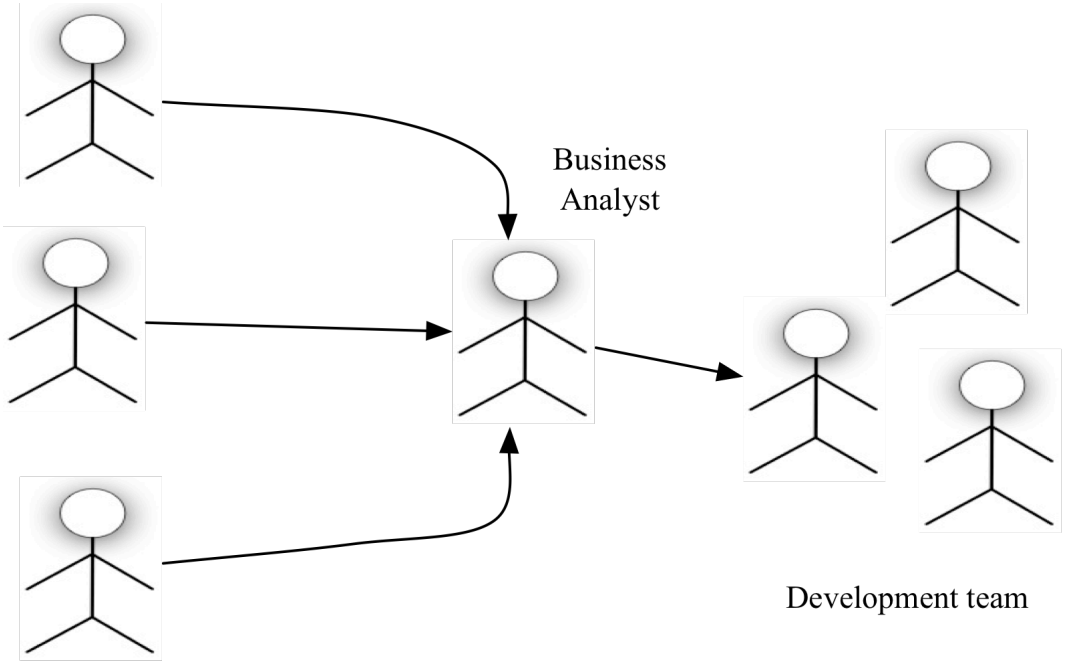

Figure 5 - BA sits between diverse customers and the development team

Project Management training is focused on the "when" of work. They learn about work breakdown structures, contingency planning, risk logs, reporting and perhaps producing Gantt charts. In an Agile project much of this is irrelevant. Agile teams operate without Gantt charts and many of the traditional artefacts of Project Management.

Business Analysts on the other hand are trained in stakeholder identification and liaison, business and process analysis and requirements discovery. This makes them a better candidate for filling this role.

Yet every extra role that is place between the development team and their customers introduces more potential gaps: messages need repeating, some get lost or distorted in the retelling, and more competing views and agendas are brought into to play. The more layers between the development team and the ultimate customer of the company the more the developers are isolated, and insulated from market forces and real customer focus. Each extra layer reduces the real Agility of the team and the business.

Sometimes the right answer is to remove layers. The difficulty lies in

knowing when adding an extra layer will improve things, and when removing a layer is the right thing to do.

BAs and the other type of Product Managers

It is not only software companies that have Product Managers. Banks, telcos, FMCG and other companies have them however their responsibilities are to the customers of the real product: financial products, telephones, washing up liquid and so on. When these ultimate products contain a high degree of technology, and in this case software technology, it is better to have these Product Manager work directly with the developers.

In software companies Product Manager replace BAs because the software is the product. In such places a Product Manager who does not understand software, what it is, how it is created, the value it delivers and how it delivers value will have problems.

Yet when the product is not software the Product Manager is unlikely to understand software and IT in the way they need to work with the business team and it is necessary to introduce a BA.

Consider a traditional travel and holiday company. Products are sold through high street branches or through a call centre. Product Managers focus on customers experience in the shop, on the holiday and stages in between. IT supports the experience but the Product Manager needs to understand holidays and customers, not IT. So they use a BA to make request on IT.

But, imagine the company ditches its high street stores and closes the call centres. It offers the products online through a website. Until customers board the plane the experience is electronic and based on IT. Product Managers still need to understand customers and the travel market, but if they do not understand IT they are handicapped. The ultimate product now has a high IT content so the BA role should be removed and Product Manager work directly with the technology team.

As more companies find their core products are delivered by software systems the role those systems play in the product experience becomes more important. No longer is software a back-office operational issue; it is a part of the front-office environment customers engage with. Someone needs to represent the software by looking both externally to how customers engage and internally at how it drives the business. This role is part BA and part Produce Manager, and I expect it to become more common and important.

The BA role on an Agile team

Most of the skills and experience a BA has can be carried directly from traditional projects to Agile projects. Gathering needs, talking to stakeholders, running workshops, writing business cases and so on are as important on Agile projects as any other. When these old techniques are carried forward they often occur in an accelerated fashion.

What will be new to some BAs is the need to step back from requirements and examine more closely what the business is trying to do and, more importantly, why. Rather than simply document some given need the BA

needs to understand the motivation behind the request and the business value. Only with this information can the BA prioritise the work requests - something else that might be new.

Requirements

What BAs won't be doing any more of is writing long requirements documents or leaving a project when the coding has only just begun. Most BA work still occurs before code is written but the two phases overlap. The BA discovers a bit, the developers code a bit; and while the developers are coding the BA discovers a bit more so when the developers have completed the first bit there is a little bit more to do.

Business needs (i.e. requirements) are taken in bite-sized chunks rather than in thick binders. These small chunks of work have been called *Minimally Marketable Features* (MMF) or *Business Value Increments* (BVI). The emphasis is on just in time requirements that produce near term business value rather than trying knowing everything that can possibly be known before asking for anything.

BAs need to be embedded in the team, in daily contact with developers, answering their questions and reviewing work as it is done. Simultaneously the BA needs to work slightly ahead of the development team, ahead but only just ahead so they are ready for the next question, or to prioritise the next request.

The development team can only move fast if they are fed a constant stream of needs. But those needs are changing as they work - not least because work which is complete changes the view on what is needed next. The further the BA is ahead of the team the greater the possibility that the needs will change while they are waiting to be implemented.

The key BA skill, namely analysis, is to the fore. While developers are concerned with creating a solution, i.e. synthesis, it is the BA's role to work out what needs to be done.

Ensure business value

Research has repeatedly shown that as much as 80% of features or functionality in customer software development is unused (Poppendieck and Poppendieck, 2003). (For commercial products the figure is usually 20% of the features used by 80% of the users.) Stemming the requests at source by understanding what is truly needed and what will actually be used can therefore reduce the team's workload by four fifths.

More importantly, Product Owners need to ensure these requests have business value attached. For a team to demonstrate its worth it needs to be producing valuable software. And if a team is going to turn down 80% of requests then it needs to ensure it does the 20% with the greatest value.

Gatekeeper and prioritiser

Product Owners on Agile teams are the gatekeepers to the development team. They decide what will be developed in the next iteration, what will be held

until later and what will not get done at all. And when work is accepted into an iteration it is the Product Owner who sets the priorities for the team in the next development episode.

In order to make the best possible decisions about what functionality to develop and what priorities to set the Product Owner role needs to know about the value of the work being requested and overall objective. Sometimes they may be told the value but more often than not they will need to determine the value for themselves. This means need understand what is being asked for, and how it aligns with the objective. They need to know about all the options and different requests being made on the system when they make the call.

They also need to know when the best option is to do nothing, and when to cease development. If value falls below cost then the greatest value comes from doing nothing.

While business value is the most obvious criteria for determining priorities there are others. Some may prefer to prioritise by risk, or others by "juicy bits" - those features which will get the most attention.

Prioritisation criteria can, and do, change over time as work progresses. In the first few iterations a few "juicy bits" may be picked off and delivered to demonstrate progress. The next few iterations might attack risk directly while later iterations use straight business value.

Go to

As if this weren't enough the Product Owner is the "go to person" to two groups of people. The first group is those who want the software to fill some need, whether we consider this people customers, users, stakeholders or simply "the business" all requests should travel through the BA (or the BA team.) This is necessary both so requests can be validated and prioritised and in order to reduce disruption to the developers.

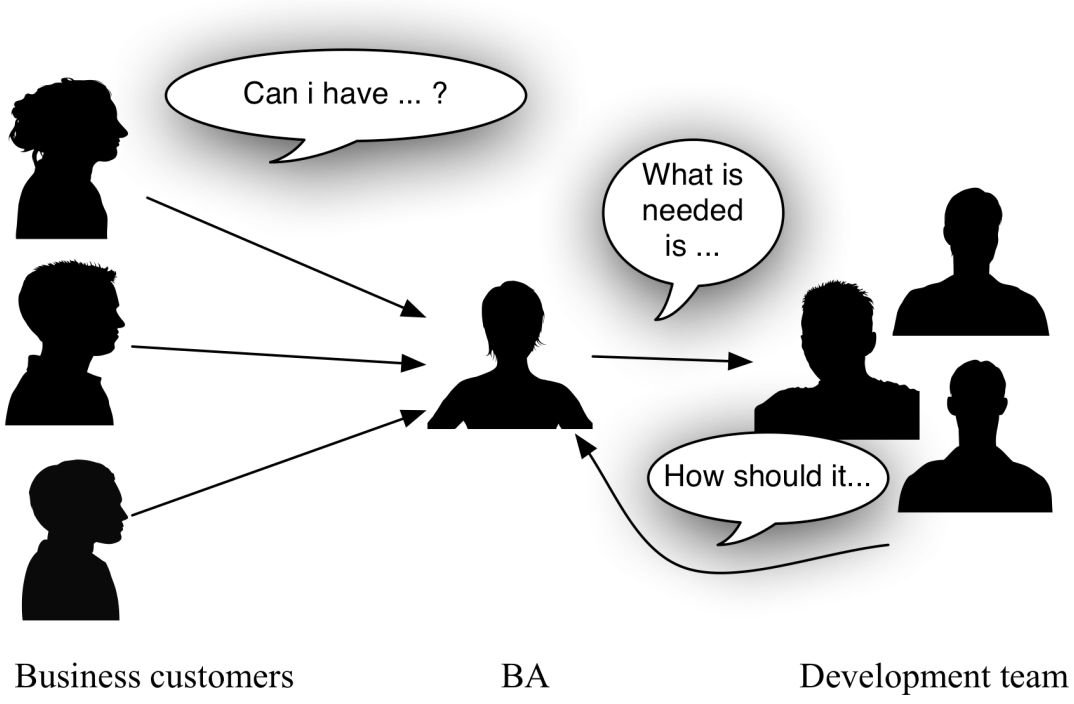

Figure 6 - Business funnels needs through BA

The second group for whom the BA is the "go to person" is the developers. When a developer needs more information about a request or when some unforeseen question arises they need to ask the BA. In some cases developers might be able to go to the final customer or user with such a question but when this is not possible, or not appropriate, then the BA takes on this role.

When the BA is also a subject matter expert (SME, sometimes called a domain expert) this they may be able to answer the question directly, other times the BA will need to know where to go to find an answer, or be prepared to make a judgement call.

Where delay occurs in answering a question development should not proceed. If a developer makes a guess there is every chance the guess will be wrong resulting in rework, delay and disruption. Alternatively a developer may work on another piece of work but this too results in disruption as one piece of work is laid to one side and another starts - and consequently tracking work is more difficult. It may well be better to have a developer do nothing rather than take their focus away from the work in hand.

In fulfilling the roles of gatekeeper, prioritiser and go to person the BA has to continually keep the delivery to the business to the front of their mind. The BA's primary responsibility is always to ensure the work being undertaken will produce business value. Ensuring business value is delivered in a timely fashion is not just a case of determining what needs to be done and asking for it. There are a multitude of options, possible actions and decisions which follow once need is determined. Maintaining a steady flow of deliveries means keeping sight of the overall objectives.

In closing

The lack of a clear BA role in Scrum and XP has created confusion about what, if anything, Business Analysts do on Agile teams. Yet there is an important role for them to play in keeping the corporate arteries clear; ensuring considered and valuable requirements reach Developers, and ensuing that Developers get the information they need, when they need it.

There is more to Agile software development than simply declaring your team "Agile." The *Magic Agile Dust* only works if teams and their management are prepared to make real changes. This includes changing the way needs are presented to the teams. The changes in the BA role are perhaps more subtle than the changes to the Developer role but are just as key in delivering genuine agility.

Publication

Published ACCU Overload 98 (August 2010), http://www.accu.org.

Author

Allan Kelly is the author of Changing Software Development: Learning to be Agile published by John Wiley & Sons (2008).

Allan can be contacted at allan@allankelly.net. More writing from Allan at:

- Allan's website: http://www.allankelly.net
- Allan's blog: http://blog.allankelly.net

Allan offers a range of training courses and consulting services in relation to Agile software development. More information is available on his company website:

- Software Strategy Ltd, http://www.softwarestrategy.co.uk

References

BRYNJOLFSSON, E. 2009. *Wired for Innovation*, MIT Press.

POPPENDIECK, M. & POPPENDIECK, T. 2003. *Lean Software Development*, Addison-Wesley.