Agile Service Transition

PATRICK BOLGER
HORNBILL SERVICE MANAGEMENT

MATT HOEY
GRANT THORNTON UK LLP

March 2014

The need for speed

Technology, and how we use it, constantly evolves. In recent years, Cloud, mobility, and social media have accelerated the pace of evolution, changing consumer behaviour and forcing businesses to get serious about their service experience. A start-up, with a disruptive business model or technology, can now dislodge established players and dominate the market in a matter of years. Organisations that cling to a traditional business model, or are slow to react to a change in market forces, may at best struggle to remain relevant, or at worst, struggle to survive. Winning business is no longer a battle between the big and the small; it's a competition between the slow and the fast. To remain competitive, businesses of all sizes must take a more 'agile' approach in delivering value to customers.

What is Agile?

In an IT context, the term 'agile' simply means using a different approach to managing IT and software development teams, with the aim of delivering the outcomes promised to the customer, in a more tangible and timely manner.

In 2001, 17 individuals representing several development methodologies (Extreme Programming, SCRUM, DSDM) convened to help their profession think about software development in new, more agile, ways. The traditional waterfall approach to software development, with its extensive documentation, heavyweight processes and slow responsiveness to change, resulted in high failure rates and caused frustration within the development community. This 'Agile Alliance' promoted organisational models centred on people, collaboration, and communities, based on a set of values listed in their 'Agile Manifesto', which says:

> *"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."*

These values and the use of methods that support them, e.g. SCRUM, gained real traction within the software development community. The potential benefits of 'agile' are capturing attention across wider IT functions and use of these methods are being explored by ITSM and project delivery teams.

Why does IT need agile?

Few would question the need for increased agility in modern business. As a critical link in the business supply chain, IT must also be able respond in a more 'agile' fashion to cope with this demand. However, agile doesn't simply mean doing things faster. It also relates to minimising wasted effort when delivering projects, introducing new (or changing existing) services, and developing software applications.

The traditional approach to delivery involves capturing and documenting the customer requirements up-front, then mobilising and managing resources through build, test, deployment and acceptance phases, meeting business needs and supporting improvements through operations. The theory may be sound, but the practice is rarely as smooth as the theory suggests

Significant time, effort, and resources are invested from initiation to delivery, and stakeholder engagement typically occurs at the end of a critical milestone phase. If requirements have not been accurately reflected at the outset, deliverables are unlikely to meet customer's needs and more time and effort must be invested to bring things back on track. Failure to meet requirements might be blamed on the accuracy of requirements gathering, inadequate documentation, or moving the goal posts during delivery, but in truth, we cannot expect customers to know exactly what they need until they have had sufficient exposure to, and experience of, a deliverable.

How does an agile approach address these issues?

Agile methods can be applied to software engineering, project management, or development of IT services, but the principles used are common across all methods.

Requirements are captured at high-level and are expected to evolve during delivery. The approach is collaborative, engaging stakeholders throughout, with frequent incremental releases. The timing of each release is fixed, and once the release has started, requirements included in that release are not changed (note this 'fixing' of requirements does not apply to requirements outside the scope of that particular release). Although perfection is not expected, each feature must be completed before starting on the next. Testing occurs early on, and frequently, within the release cycle. Whiteboards and stand-up meetings are used to simplify communication, maintain visibility of progress, and managers facilitate the flow of work by removing obstacles and empowering the team to make decisions.

This collaborative and iterative approach actively involves users and provides regular checkpoints to ensure that deliverables are in-tune with customer expectations. Agile projects done well often benefit morale, as collaboration engages staff, raises perception of the value they contribute, and boosts overall productivity.

Can we take an agile approach to all IT deliverables?

There are no hard and fast rules, so in theory, agile methods could be used to approach any IT deliverable. However, some projects suit an agile approach more than others. Highly complex and enterprise-wide projects are typically less suitable, but elements of the project could benefit from agile methods.

The real trick is in knowing where agile methods can be used to engage stakeholders, provide better visibility and reduce waste in the delivery cycle; then to combine agile with more traditional methods to manage delivery of complex projects. Agile operates according to a different set of values than more conventional approaches; if a mixed approach is taken, the greatest challenge is managing the cultural balance of different teams involved in the delivery.

Agile ITSM… really?

Agile, at first glance seems very much the "chalk" to ITSM's "cheese". We've discussed so far the approach to agile being one of breaking down work into chunks and performing frequent releases of these 'time-boxed' chunks of work. This seems to be in direct conflict with an ITSM framework such as the structured, process driven ITIL and no doubt the phrases encountered in our discussion so far will send shivers down the spine of many a hardened ITSM practitioner.

But despite the chalk and cheese appearance of the two frameworks, agile and ITSM can work together well with considerable benefits in the right situation. When bringing the two together, it's not a case of bringing radical changes

to your existing Service Transition processes, it's a matter of adapting what you have. Even agile developments or projects need to have a defined scope or target, budget and an end date otherwise they would run out of control sooner rather than later. Let's take an example of developing and implementing a new corporate Intranet with a budget of £X within three months. In the waterfall world, we would have one release and that would be at the end of the three months – a clear milestone date for change and release management. Of course, this project might run over because the customer isn't going to see the product until it is fully developed and by this time they may of done one or more of the following:

1) forgotten what these asked for

2) changed their requirements

3) change their mind on elements of the design now they've seen it and it wasn't quite how they envisaged it in their minds.

So change management may not get their definite date after all.

Using an agile approach, we can break down the development of the Intranet site into chunks. Let's saying we are applying the SCRUM methodology and we are performing sprints of three weeks. Table 1 suggests how this may look.

Table 1

Sprint #	Description	Date	Release
1	Home page and menu development	End of week 3	No
2	Back end content administration, plus amendments to home page and menu based on customer feedback	End of week 6	Yes
3	Sub sites development	End of week 9	Yes
4	File repository development	End of week 12	Yes

Here we see the customer getting sight of the application after three weeks. Whilst it is not finished at this stage, the customer can get a feel for what the Intranet looks like, feedback and influence the rest of the development. At this stage, should the customer have changed their minds or found it was not what was expected, this early viewing means only three weeks work is impacted rather than three months. A better outcome for all involved.

You will note that sprint one is not released, at this stage it is not usable, however, is substantial enough to be presented to the customer at the end of the sprint and following that subsequent sprints are released in the example are released into the production environment.

Developing using an agile approach gives us a couple of key factors that means we can make minor amendments to the way the development and the service transition processes interact:

- a 'change log' of what is changing in each sprint via the product backlog

- continued customer involvement and testing throughout the development of the application.

With the above in mind, the point at which the Request For Change (RFC) needs to be raised comes into question. Does it still need to be fully in place and authorised at the start of the development? Why not accept agile developments into the change management process at the point where the work has been done? After all it will have been controlled via a project, built and tested away from the production environment, had customer involvement, received continuous testing throughout the process and also comes equipped with its own change log. Not wanting to take away the benefits of change management, raise your RFCs for each release when the release dates are defined in order to get them on the change schedule and then beef them up with the product backlog and further details as that information becomes available.

Ask yourself what benefit the questions you are asking of a waterfall change give you. Now ask yourself how to get the same value from an agile approach. It may lead to you asking the same question at a different point, or a slightly different question. You want to know things like when the implementation will be made, what components are affected, what level of testing has been done, what else might be impacted, what the implementation plan is – some of these are very readily provided by the structure of an agile approach.

Agile processes

You may struggle to work with agile changes if your change and release processes themselves aren't agile too. In the above example, an existing change management process may dictate that each release must go through the Change Advisory Board. It should be considered whether this is absolutely necessary. It may be right for the initial release of the application to pay a visit to the CAB, but the incremental nature and number and frequency of subsequent releases would make this a chore and for unproductive change reviews. Simply put, the CAB would not add value scrutinising each release. After all, the processes agreed for test and release within the project (which you can cover in the first RFC) should be sufficient to give you the assurance you require. Instead, following the initial CAB review, standard changes could be utilised or classing the releases as minor and having them receive authorisation from an appropriate technical manager or service owner and Change Management.

The focus so far has been on Change and Release Management, but what about the other Service Transition processes? Again we don't need to make any radical changes to the other main processes such as Configuration Management and Knowledge Management. An agile approach will supply incremental updates to these processes making updates to the deliverables of these processes more frequent but more manageable. Consider the task of updating a knowledge base with information on the amount of changes over a three or four week period compared with six months. An agile approach is more likely to result in a more up to date CMDB and knowledge base.

Agile process managers

It's one thing having an agile process, but what about the person who operates the process on a day-to-day basis? If they are more interested in adhering exactly to the process then delivering valuable outcomes, are not open to being more flexible or any of the other values in the agile manifesto, then the benefits of agile ITSM are not going to be realised. To be agile the process owner must take less rigid stances. This could be simple changes such as being happy to accept minor changes with a shorter lead time for the simple reason that it would be beneficial or valuable to the customer to have it soon. Hitting the button to set the emergency change alert light flashing in every case isn't the right mind-set for a process owner to have in an agile world!

Benefits elsewhere

Incident Management and Problem Management like agile. An agile approach means the cause of incident and problems records are easier to identify – a sprint release contains a lot less change than the end product of a waterfall approach. It's less of a needle in a haystack to find and more fresh in the mind of the developer. Moving from the identification of incidents and problems to the other end, the resolution, agile allows things to be fixed quicker. Sprint releases come around quicker than waterfall releases, and with the ability to add to the product backlog and decide what goes in a release at the beginning of a sprint, means you can target incidents and problems quicker.

What's stopping you?

We've already discussed where agile is recommended to be used and where it isn't. With all this flexibility and changing requirements, one would be forgiven for thinking that an agile operation may bring chaos and disorganisation to your projects and processes if adopted. Table 2 takes the most common perceptions about agile and explains why an agile approach can actually bring more structure and greater productivity to the table.

Table 2

Perception of agile	The reality
No planning	In agile planning takes place continuously rather than in one large time consuming chunk at the start. There is: - the planning meeting at the start of the sprint - the daily meeting (I've done this, I'm doing this next, these are the barriers) - release planning
No design	Design is a continuous process throughout, rather than a large upfront chunk of time at the start of a project which becomes immediately invalidated and wasted as soon as requirement to deviate becomes apparent.
Lack of governance/control	Looking at agile from a software perspective, when you get into it, it's actually a very disciplined process of delivering software. There is no reason why this can't be a disciplined process of delivering and improving services or processes: - testing **must** take place - there are fixed deadlines - the use of agile tools such as burn down charts actually empowers the scrum master/project manager in estimating how much work can be included in each release, predicting when work will be completed highlighting where there is extra capacity to do more or, conversely, where project resource performance is lagging behind others.

Where can you start?

If you want to get a feel for how agile might work for your organisation, you can apply agile methods to your existing processes. Scrum is relatively easy to understand and may be a good method to get you started. Your Change Manager could assume the role of Scrum Master and work with a Scrum Team to plan a number of sprints to reduce your change backlog. This will allow the team to get used to sizing the effort required to complete tasks within a sprint, and experience how daily stand-up meetings and collaboration helps drive a release. The sprint retrospective enable the team to review what worked, what didn't, and what needs to be done differently the next time round. As the team becomes more effective at running sprints, velocity will increase. Once you've become accustomed to using agile, you can apply these methods to other ITSM disciplines. When agile works well, it brings a real sense of achievement. Teams realise the benefits of collaboration, and individuals see their efforts quickly translated into results, which boosts staff morale.

Conclusion

We know from experience, and this may well be the case for you as you think about what you or your organisation could get from using agile, that customers rarely know what they want until they get their hands on something and it is important to reiterate that agile is not just for software development. Customers also expect change more often. Just take the example of 'app' updates on smart phones – the vendors are constantly producing new versions prompting the smartphone user to update; companies like Facebook are introducing changes to their products on a daily basis.

With agile Service Transition processes in place operated by process owners of an agile mind set, the common complaints of change management and release management being too bureaucratic, full of red tape and inflexible can be greatly reduced and value delivered to your customers quicker and more tailored to their evolving requirements. You never know, it could lead on to extending your agile approach and exploring the world of continuous delivery and the concept of devops, but that's another whitepaper…

Recommended reading

Agile manifesto
http://www.agilemanifesto.org/

Are you a pig or a chicken - Kevin Holland ITSMF UK Service Talk Summer 2013
http://www.itsmf.co.uk/Members_Area/ServiceTalk/ServiceTalk_Interactive_Editions.aspx

The Phoenix Project - Gene Kim
Published by IT Revolution Press, 10 January 2013