Agile Delivery at British Telecom

Articles

Posted by:

Posted on : 2007/10/8 3:24:03

It is becoming clear, not least from the pages of this publication, that agile development methods are being adopted or at least considered by a growing number of software development teams & organisations. Whether you are already an active practitioner agile development, or considering its adoption on your project, you will be aware of the business benefits that can be derived through faster and more effective software delivery not to mention the motivational impact it can have on development teams. Alternatively, maybe you work for a large organisation that has yet to make any serious inroads into agile development, and are left wondering how agility could be made to work on a large scale.

Author: Ian Evans, British Telecom, www.bt.com

If you’re in the latter camp, or even if you are not actively considering agile development as such but are struggling to deliver large and / or complex programmes using traditional approaches and wishing there was a better way, then you are probably where British Telecom (BT) found itself in 2004. That was before the arrival at the company of a new CIO who systematically set about replacing the company’s long-standing waterfall-based delivery processes with one that embodied the key principles of agile delivery.

This article presents an overview of the approach taken by BT, illustrating how agile development principles can be applied successfully at the enterprise level. Needless to say, the approach taken by BT is not for the faint hearted – it has included a high degree of risk, and certainly a lot of pain. Now well into its second year however, although the transformation is far from complete, it is already paying dividends.

Background

BT employs some 8,000 IT professionals in a variety of roles including project & delivery management, architecture & design, software engineering, integration & testing, operational support and service management. Much of its internally-focussed development work has traditionally been channelled through a number of business-focussed delivery projects or programmes, ranging from quite small, simple developments to large-scale and complex business solutions, the latter tending to be the norm.

The predominant delivery approach, certainly for the larger delivery programmes, was very much waterfall-based. The use of agile development practice, notably DSDM and Scrum, was limited to a small number of fairly small, self-contained development teams. BT was in fact one of the founding members of the DSDM Consortium and took an active part in shaping the method in its early days.

Despite successfully delivering a number of large, complex solutions into a dynamic, competitive yet highly regulated business environment, many significant transformation programmes were struggling

to deliver any notable results in an acceptable timeframe. As part of a CMMI-inspired improvement strategy, efforts had been made to formalise acknowledged best practice processes into a standard delivery methodology. In 2004, this standard methodology was in the process of being rolled out when the new CIO made it clear that an entirely new agile approach was needed.

Drawbacks of the waterfall

Reinforcement of current waterfall-based practices was not really the answer however. Many of the delivery problems experienced at BT, and no doubt other large organisations, stem from the nature of the waterfall lifecycle itself. Some examples of these problems are given here. For a more complete demolition of waterfall practices, refer to Craig Larman’s excellent work [1].

Poor requirements capture

Capturing requirements certainly isn’t a bad thing. On typical large programmes however,
* Individual business stakeholders are anxious to incorporate all of their known requirements into the first / next release
* "Gold users" generate hundreds, if not thousands of detailed requirements that often bear little relationship to the business problems that needs to be addressed
* Most if not all requirements are given a high priority
* The requirements themselves, at best, represent today’s view, which will certainly have changed by the time the requirements are actually implemented

Disconnected design

Given the sheer number of requirements, the design community finds itself spending most if its time trying to figure out what they mean. Meanwhile,
* The requirements analysts move on to other projects, taking with them important tacit knowledge
* Some stakeholders become concerned that their requirements are not being adequately addressed, and therefore refuse to sign off the designs
* Other stakeholders unearth more requirements or raise change requests, diverting scarce design expertise onto impact analyses

Development squeeze

With the design stage having slipped, development teams find themselves under intense pressure to deliver components into the integration environment by the originally agreed date. In fact, they often take the decision, reluctantly, to start development against an unstable design, rather than do nothing or divert resources to other programmes. Inevitably, system testing is cut short so that original timescales are met and the programme is seen to be on target.

The integration headache

The integration team has a set number of weeks during which it needs to integrate what it expects to be fully functional and relatively bug-free code. Because of the instability of the component code, and the lack of any effective regression test capability, effort is instead diverted to trying to resolve elementary bugs in the delivered code, liaising with a development team that is now engaged in the

next major release. Actual integration therefore runs into months, creating a knock-on effect on other programmes requiring the services of the Integration team, not to mention frustrations within the business community who had been busy preparing themselves for an on-time delivery.

The deployment nightmare

It is now at least 6, or even 12 – 18 months since the business originally identified the need for this particular solution. Compromises and oversights made during the requirements and design phases, followed by de-scoping during development has resulted in a solution that bears little relationship with what was originally envisaged. Besides, the world has actually moved on in the meantime. The business then finds that the solution is not fit-for-purpose and refuses to adopt it. Worse, they adopt it and soon find that it is slow, error-prone and lacks key features, and eventually revert to the old system. The end result – more shelfware!

Early in each delivery cycle, the programme sets out clear targets for what it expects to achieve for the business during that cycle. These targets invariably include a strong emphasis on the end-customer experience, such as overall response times, transaction success rates, and so on. At the end of the cycle, the programme is assessed against these targets, and the outcome of this assessment will influence the timing of bonus payments for the programme team members. Programmes failing to deliver business value over a series of cycles face being closed down altogether.

This of course places a certain amount of pressure on the (internal) customer to be clear about the business priorities and the features that would provide the greatest return on investment. It also requires that the customer is ready and able to deploy the solutions into the business and realise the intended benefits. In practice, programmes often take two or more 90-day cycles to progress a particular solution to a point where it is fit for deployment. Even so, there is an opportunity at the end of each cycle to assess what has been delivered so far, and to provide feedback based on what has already been developed.

Agile Development Although not intended to be a "silver bullet", Agile Development should help resolve these problems. For example,
* Focussing only on what’s really important to the business will help overcome the tendency to want to document all requirements up-front. Techniques such as User Stories [2] often proves helpful here.
* Involving your customer as part of your team certainly helps ensure that the solution is developed in line with expectations, and doesn’t incur too many surprises once completed
* Developing in short iterative cycles again helps to break the problem down into more manageable chunks, and allows feedback to be gleaned early and often. The Scrum [3] method, for instance, advocates 30-day cycles.
* Applying a ‘test-first’ and continuous integration development approach, two of the core practices of XP [4], not only helps to further break down the complexity of the problem but also avoids the ‘integration headache’ described above

The Challenges of Enterprise Agile

This is all very well when you have a development team of some 10 or 20 people, or if you are

implementing agile development on perhaps one of your larger (say, 100 people) projects and have ample mentoring resources at your disposal. Even then, if you’re adopting Agile for the first time, you have the challenge of changing people’s mindsets, aligning the new practices with a supportive customer, while also remembering to successfully deliver the actual project.

For most large organisations, architectural considerations also need to be addressed. Also, no doubt a high proportion of the code base is probably implemented in the form of third-party Commercial Off-The Shelf (COTS) components. On top of this, you then have the added complexities of off-shore development, widely distributed in-house resource, an IT department that is probably not highly regarded within the business, plus a whole pile of legacy code for which tests probably don’t exist in any shape or form let alone automated.

Finally, your programme has significant business commitments to make, and little scope to take on the added risk of adopting new practices. So where do you start?

Scaling to the Enterprise – The BT Approach

At BT, the drive towards agile delivery started with an uncompromising determination to introduce shorter delivery cycles across the entire delivery programme portfolio, establish a ruthless focus on delivering real business and end-customer value, and to nurture a strong collaborative ethos between the IT and Business communities.

90-day cycles

With more or less immediate effect, all delivery programmes were required to adopt a standard 90-day delivery cycle. That is, having picked up one or more distinct business problems on day 1, a working solution is expected to be available, fully-tested, for deployment by the end of the remaining 90 calendar days. For BT, this represented a seismic shift from the 12+ month delivery cycles that were previously commonplace.

Each 90-day cycle is kick-started by an intensive 3-day "hot house" in which cross-functional teams explore one or more business problems in some detail, with the main stakeholder(s) usually being at hand to resolve any queries. Out of each hot house, the intention is that one or more working prototypes are produced which, if accepted by the stakeholder, forms the basis of the development activity for the remainder of the cycle.

For the remainder of each cycle, the intention is that wherever practical, programmes pursue agile development practices such as more fine-grained iterative development (e.g. 2 – 4 weeks),

Step 2 – Focus on Delivering Business Value

Early in each delivery cycle, the programme sets out clear targets for what it expects to achieve for the business during that cycle. These targets invariably include a strong emphasis on the end-customer experience, such as overall response times, transaction success rates, and so on. At the end of the cycle, the programme is assessed against these targets, and the outcome of this assessment will influence the timing of bonus payments for the programme team members. Programmes failing to deliver business value over a series of cycles face being closed down altogether.

This of course places a certain amount of pressure on the (internal) customer to be clear about the business priorities and the features that would provide the greatest return on investment. It also requires that the customer is ready and able to deploy the solutions into the business and realise the intended benefits. In practice, programmes often take two or more 90-day cycles to progress a particular solution to a point where it is fit for deployment. Even so, there is an opportunity at the end of each cycle to assess what has been delivered so far, and to provide feedback based on what has already been developed.

Step 3 – Instil a Collaborative approach

Truly successful programmes require a strong partnership approach between the business customer and the development community. Within BT, close collaboration is established at the outset of each project through attendance at the hot houses. The onus is then on the programme teams to ensure this collaboration continues throughout the delivery cycle through design walkthrough sessions, prototype reviews, and so on.

In practice, the end-user representatives you want to have at your hot house are also the ones who are hardest to release from operational duties. With several programmes running numerous hot houses during the course of a year, this problem can quickly become compounded and often makes it extremely difficult to achieve true collaboration on a day-to-day basis. However, collective ownership of the eventual solution needs to become the accepted norm, and any practical steps to enhance collaboration should be taken.

Early Reflections

Despite some turmoil at the start, and some painful failures among some of the earlier hot houses & delivery cycles, the new practices have now become accepted as the norm across BT. Now well into the second year of its shift from waterfall to agile delivery practices, few people would be willing to revert to pre-Agile practices. In fact, most programmes are now seeking ways of refining their delivery processes further by adopting truly iterative & test-driven development practices within each delivery cycle.

However, some observations would be worth noting.
* Firstly, when you’re embarking on an agile delivery strategy at the enterprise level, it is imperative to quickly establish a ‘critical mass’ of people who not only grasp the ideas behind it but are also comfortable with its application. To establish that critical mass, you will probably need to turn to outside help. A number of consultancies now specialise in the adoption of agile practices within large organisations. BT chose to use two different companies, each of which brought different strengths and perspectives. Further to this, it is also essential to establish a strong central team to provide ad-hoc support, nurture the new techniques, and to actively support the new practices.
* Certain agile practices, such as test-driven development, are harder to adopt when most of your development is based on legacy code and / or externally-sourced components. Similarly, continuous integration becomes extremely complex when some of you main components are shared across multiple programmes. Some of BT’s programmes are now pursuing test-first and continuous integration techniques, but this takes time and investment and is only being done on a selective basis.

* For Agile Development to work at the enterprise level, you still need to pay due attention to your systems architecture. "Big Design Up-Front" (BDUF) may not appeal to the agile purist, but re-factoring of an enterprise architecture simply isn’t practical.
* Not all delivery activity fits neatly into the agile development model. Given a choice however, the natural tendency is to pursue most activities using the traditional approaches – you can always find some excuse why "the new approach" isn’t appropriate on your project. If you go down this road, agile delivery will at best become a niche activity. At BT, a strong mandate ensured that all programmes put the new practices to the test whether this seemed logical or not. This helped to break through the "pain barrier" and to ensure that the new practices were given a real chance of taking hold.
* To be truly effective, the agile approach needs to reach right across the business, not just the IT organisation. You might expect that the business would be excited at the prospect of having regular deliveries of valuable functionality. However, the business also needs to move away from traditional waterfall practices and change how it engages with the IT organisation. It also has to place its trust in the IT organisation (something that certainly takes time) that it will deliver as promised. It then needs to ensure that it is geared up to exploit the deliveries to gain maximum business benefit.
* Finally, remember the old adage – "There’s no gain without pain!" Applying the principles described here on large projects or programmes in typical large organisations requires courage, determination, and no small degree of risk. Also, such a radical strategy requires absolute commitment from the very top.

Conclusion

Re-orienting a large IT organisation from pursuing well-established waterfall-based delivery approach to being a truly agile delivery unit takes patience and time, as well as a lot of commitment. In BT, where the initial steps towards enterprise agile delivery were taken late 2004, there has been a noticeable and decisive shift away from waterfall-based thinking. It has also transformed, quite radically, the traditional function of the IT department as a supplier of IT services to one where IT is now seen as integral to all major business initiatives. Above all else, it has created an attitude, bordering on obsession, of delivering real value to the business through IT.

Despite the early successes however, it is clear within BT that there is still a long way to go before it can consider itself to be truly agile. For any large organisation, the journey from waterfall to agile can be very long and challenging. As with other proponents of Agile Development however, few at BT would want to turn back to the old ways.

References

* "Agile & Iterative Development" by Craig Larman, Addison-Wesley (2004) ISBN: 0-13-111155-8
* "Agile Software Development with Scrum" by Ken Schwaber & Mike Beedle, Prentice Hall (2002) ISBN: 0-13-067634-9
* "User Stories Applied for Agile Software Development" by Mike Cohn, Addison Wesley (2004) ISBN: 0321205685
* "Extreme Programming Explained" by Kent Beck & Cynthia Andres, Addison Wesley (2004) ISBN: 0321278658
* "Lean Software Development – An Agile Toolkit" by Mary Poppendieck & Tom Poppendieck, Addison Wesley (2003) ISBN: 0321150783
* Agile Manifesto – http://www.agilemanifesto.org Originally published in the Summer 2006 issue of Methods & Tools