PhUSE 2011

Paper AD02

Adoption of Agile in Medidata

Tony Hewer, Medidata Solutions Worldwide, Uxbridge, UK
Andrew Smith, Medidata Solutions Worldwide, Uxbridge, UK

ABSTRACT

About two years ago, Medidata Solutions committed to adopting an Agile approach to its product development and operations. And, it has evolved its business model to one of SaaS. This paper describes the transition that has taken place within our company of this move from traditional, "waterfall"-based processes - the challenges, obstacles and benefits - both to our company as well as to our clients.

We describe a journey rather than final outcomes - where we have come from, where we are and where we wish to go from here! We explore how this has changed our company in terms of our processes, our use of tools and the changes to the behaviors and mindsets of our personnel. We also describe how we have sustained our fundamental need to offer high-quality, low-defect products and services that conform to regulatory requirements as well as satisfying the scrutiny of our clients.

INTRODUCTION

To quote: *Medidata Solutions is a leading global provider of SaaS clinical development solutions that enhance the efficiency of customers' clinical trials. Medidata's advanced solutions lower the total cost of clinical development by optimizing clinical trials from concept to conclusion.*

This is taken directly from Medidata's website. Some key points that relate to this paper:

- SaaS – software as a service. Our software products are firmly based on this key concept across the range of functions in our customers' clinical development businesses
- Those products – and the service solutions that accompany them – are geared toward facilitating greater efficiencies for our customers in their conduct of *their* businesses
- It is the features within our products that enable those efficiencies and….
- …hence, we strive to build those features in a manner that brings them to market to meet our customer needs as expediently as possible.
- We are supplying solutions to our customers who work in a highly regulated industry. Their expectations of us are that our solutions development, testing etc meet the regulations.

WHERE WE WERE

Our legacy was firmly based on the traditional approach to software development that is probably best visualized by the classic V model as shown in Figure 1. Oftentimes, this approach is also dubbed the "waterfall" model.

Working in our regulated environment, this, largely serial, approach was characterized by the following:

- Each "phase" was highly dependent on the totally satisfactory conclusion of the previous phase
- Each phase produced evidential deliverables that typically took the form of human-readable documents. Examples included:
    - The requirements specification (RS) – resulting from the Requirements Analysis.
    - The design specification (DS) – resulting from the System Design, Architecture Design and Module Design phases.
    - The overall testing approach – oftentimes referred to as the *validation* approach – resulting from the
        - Unit Testing evidence
        - Integration Testing evidence

  - System Testing Evidence
  - Acceptance Testing.
- Documented procedures that described the ways and means for drawing up these deliverables as well as the methods for controlling other aspects of the approach - such as:
  - Change control
  - Version control
  - Code management
  - Defect handling
  - Development project governance
  - Document storage.

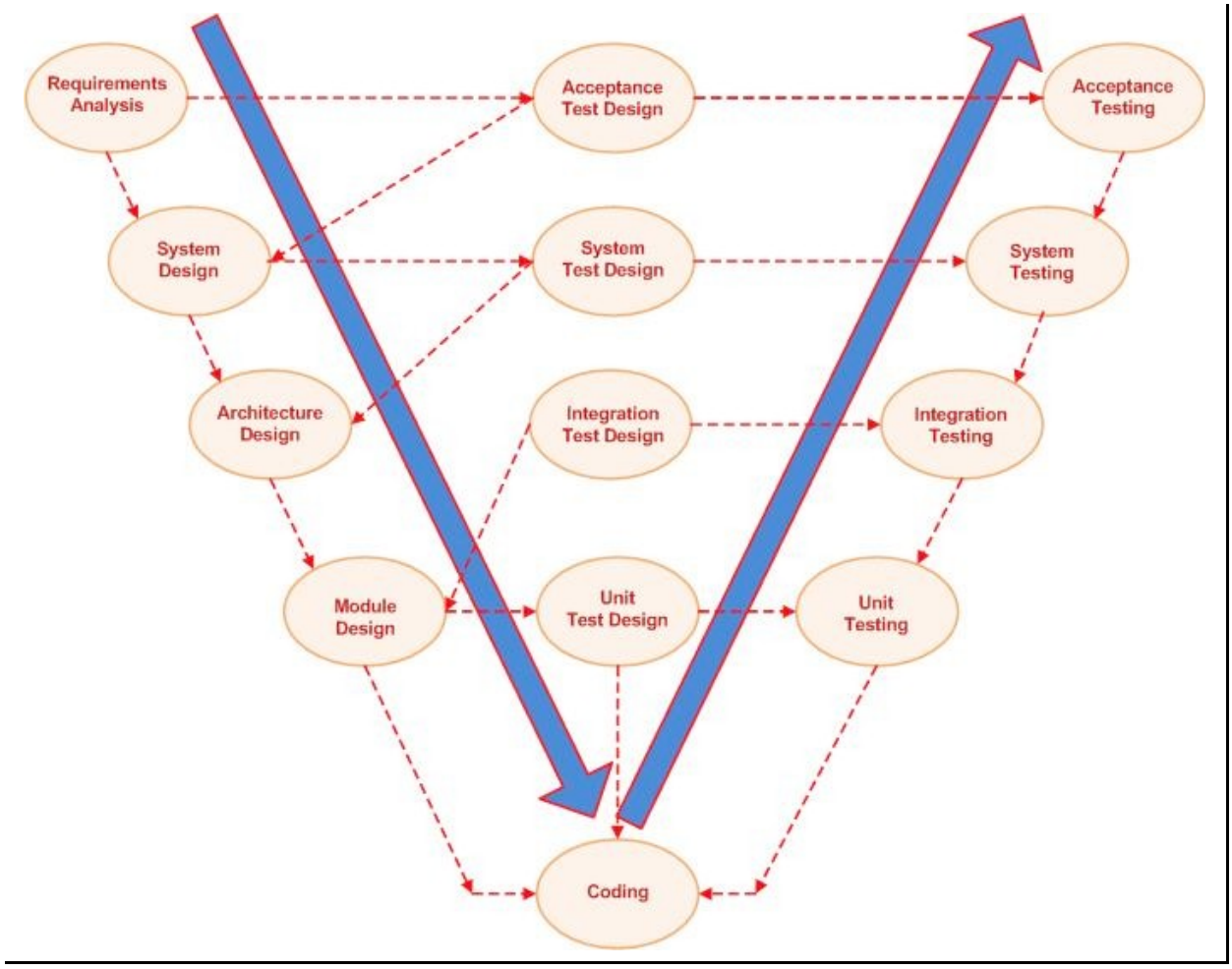

Figure 1: Classic V model

DID THIS WORK?

Well, yes! And, moreover, we developed and brought to market some great products using this approach. But, there are many issues that arose including – and certainly not limited to – the following:

- Clarity of Specification: With this inherently linear approach, the real danger exists that what gets delivered (at the end) is not what was in the mind of the person who thought about the requirements in the first place. We have all seen this picture (Figure 2) – and, moreover, we can all relate to real-life experiences. In Medidata's case, our customers were, indeed, real customers, but also ourselves – seeking to enhance and further develop our product feature set

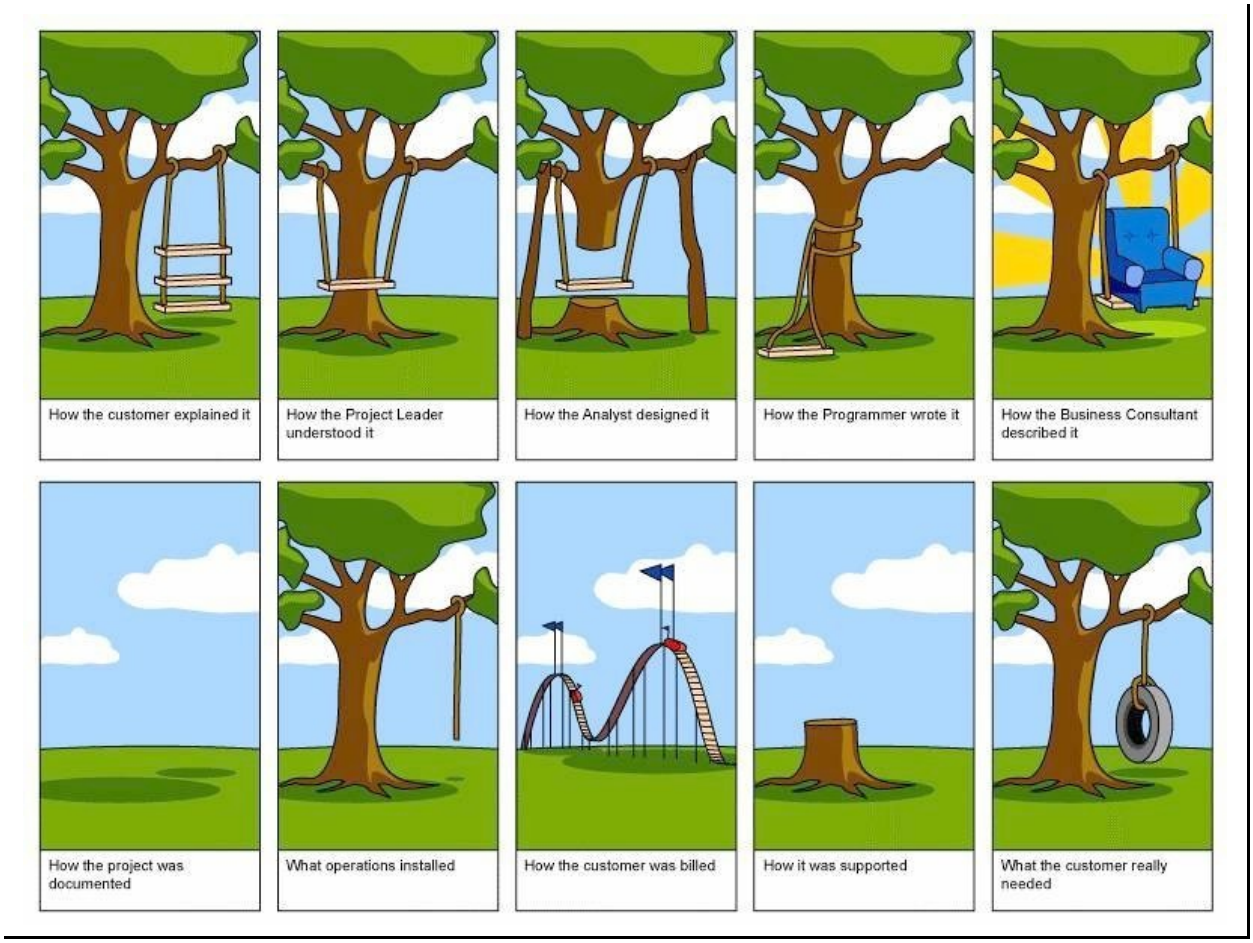

Figure 2: The Reality!

- The Code: The actual coding of the software is a minor component of the overall model and approach. There is little that verifies that the code is good – or that it does everything that is specified. In fact, could the coding be doing something that's superior to what got specified? At that stage, we don't know.
- Time and Resources: Lots of issues here! The end-to-end timeline associated with this model is, typically, quite long by virtue of its linear nature. The different resource skills are, typically, only deployed as and when necessary; in fact, they are often deployed on other activities (away from the project in question) when not required. This, inevitably, results in resources not being in the right place at the right time and, hence, the overall project gets delayed. But, most importantly, the lack of engagement of all the different skill sets across the project is the major contributor to the production of armchairs and useless swings rather than a hanging tire.
- Documentation: As noted above, creating *documentation* in our industry in not optional. Our legacy approach to this was literal – human-readable, often paper-based, files. A large human effort was required to manage this in itself as well as the mechanisms of having such documentation reviewed and approved by several people.

WHY WE NEEDED TO CHANGE

Our company was growing in a number of ways: more products, more services, more clients and more people (within our company) to develop and sustain those – along with the professional services that complement them. Above all, we determined the following guiding principles to our revised approach:

1. Faster time to market: We needed to be responsive in a timely fashion to our clients' requests for new or enhanced product features as well as, naturally, rectifying the defects that our products had.
2. Build the right thing!: Further to above, we needed to have a way of working that checked and checked again that what we built (or what we enhanced or fixed) really did match both our own and our clients' expectations and requirements.

3. Incremental delivery: We realized that we did not need to commit to building and delivering everything at the same time. Indeed, from our own perspective, there were several good reasons not to do so
4. Transparent prioritization: Further to the previous point, we needed to be clear about what we were going to build and deliver. Moreover, we would be clear about what was most important – and what was less important
5. Representation: Our clients – as well as us – were to be plugged into the prioritization process. And, of course, such prioritization could change – due to issues encountered or changes to the priorities themselves.
6. Automation: We recognized that in order for us to adopt this new approach with a greater emphasis on an iterative approach, we would need to do the same thing more often. The most vivid example of this is testing – specifically, regression-type testing. In turn, we recognized that we would need to invest in automating our approaches and techniques to testing. Our only resourcing was, of course, also a factor in this realization – i.e. our growth could not be allowed to cause a scale-up of our testing human resources in a linear way.

What's described here is the *Agile* approach albeit, expressed in terms that relate to the strategy – and success of our own business.

TO CHANGE – AND WHERE WE ARE TODAY

Our approach to adopting Agile was, in itself agile! Or, in other words, it was not big bang! We started with a small number of projects – some of which were totally new – for new products to bring to market, others being incremental developments associated with existing projects. Doing the former was easier than the latter!

Whilst there was an element of diving in at the deep end, we paralleled a training program around Agile – geared especially to the practitioners – our software development team leaders, our testing leads, our product managers, our project managers and our quality assurance representatives. We followed up with training to more senior folks within our company so that they could support, guide and encourage the ongoing proliferation of agile ways of working

Also, at the same time, we formalized our new approaches in new Standard Operating Procedures that operated and alternative to (and not a replacement to) the legacy SOPs that we had followed in the past. We recognized that our new approaches might not be familiar to our clients – especially in a client audit situation. So, we've gone to the trouble of produced an eLearning module for client auditors telling them about Agile, our use of it, the validation evidential artifacts and the SOPs that govern it.

Our "agile SOP" is not too prescriptive – deliberately so. It gives latitude to our teams in how they execute their projects but it does define what deliverables MUST be done so that we can readily demonstrate (via documentary evidence) to the outside world – when we need to – that we're doing what our SOP says we should do.

And, finally – and also at the same time – we heavily invested in automation and structured approaches! Without going into too much detail, this included automation of things like:

- Code review
- Feature Files (which have become, essentially, the modern equivalents of the RS – but, importantly, they get made by many members of the team and, in turn, add value to all members of the team)
- Testing – which directly leverage the Feature Files
- Validation documentation
- Traceability.

We have learnt as we've gone along and scaled up our agile ways of working – both from our successes as well as our mistakes – and there have been many examples of both. We produced what we've termed an *Agility Map* for our company. This changes a lot but it always provides a snapshot of how agile we are right across our product suite, our release planning cycles as well as from the perspective of the functional and organizational structures within our company. We've made a real conscious effort in this continuous improvement space – not just at the end of a project, but during projects. See below for more detail on this.

We have also invested in Improvement Communities that, perhaps, historically can be viewed as Communities of Best Practice or Centers of Excellence. Specifically, we've formed communities focused on

- Best practices for scrum masters
- Best practices for product owners
- How best to handle product defects in the context of the next releases of the product alongside enhancements

- How to streamline (and enhance the agility) of formal, documented processes across our product to market lifecycle

Today, the vast majority of our project teams are using an Agile approach. We have come to characterize these as "those that are using the Agile SOP". The residual small number are also using some of the agile principles (especially in our software development) but continue to use our legacy SOPs

So, here's the detail of how we work today:

- Each of our products has a *backlog* of work that, empirically, contains *stories* that embrace
  - New features – both as defined by our clients and us. These aren't just "functional features"; they include features associated with the way we operate the product, support it, and monitor its usage and so forth. A key aspect of developing and maintaining this backlog is, of course, our client interactions. We have invested in user groups and client forums of various kinds with, often, such bodies becoming focused on very granular detail
  - Defects in the product.
- We plan a product release and this is generally driven by time – not features or defect fixes.
- We develop the items in the backlog into stories that, in turn, get defined into lower-level stories. In other words, we granularize as much as we can into small chunks of work, each of which can be characterized with a tangible deliverable that can be verified as being done and ready. And which, moreover, could be potentially releasable to the market.
- We continually refine and prioritize the backlog
- We iteratively work out the level of effort that is required to undertake the defined stories and then allocate them to *sprints* of work. So, each sprint, at its start has a backlog of work. Each sprint of work is a defined period of time for the members of the team. The overall workload is determined by the resource availability. Each sprint is, typically, just two weeks!
- At the beginning of each sprint, we plan it – in detail, at the task level and give ownership of each task to a named individual.
- Each day, the scrum team meets to report on progress, what's up next and, importantly what, if any impediments there are to fully achieving the objectives of the sprint.
- This daily scrum meeting is short and sweet with little in-depth discussion. The scrum master (who is, if you will, the scrum meeting chairman) is always responsible for driving and/or facilitating impediment removal.
- Active scrum team members are known as *pigs*. The number of pigs can vary as the overall project moves forward. If a pig is absent (for whatever reason), there's an expectation that the team can still get a view of status. Chickens can also sit in on scrum meetings but are not expected to actively nor regularly contribute.
- At the end of each sprint, there is a retrospective meeting that focuses on:
  - What was achieved – and what was not achieved
  - How could things have been done better
  - A *reveal* of what actually got delivered; often, this takes the form of a demonstration of a newly developed feature. Or, it might, more simply, be a document walk-thru, for example.
- The sprint end meeting usually dovetails into the next sprint's planning activity.

Figure 3 shows a simple representation of some of the above points. Above all, the scrum team is a self-governing team that shares responsibility for the success of the sprints and the project as a whole. The scrum master, as mentioned, lightly coordinates that team encouraging and, sometimes, enforcing, the good practices and learnings (from inside and outside the team).

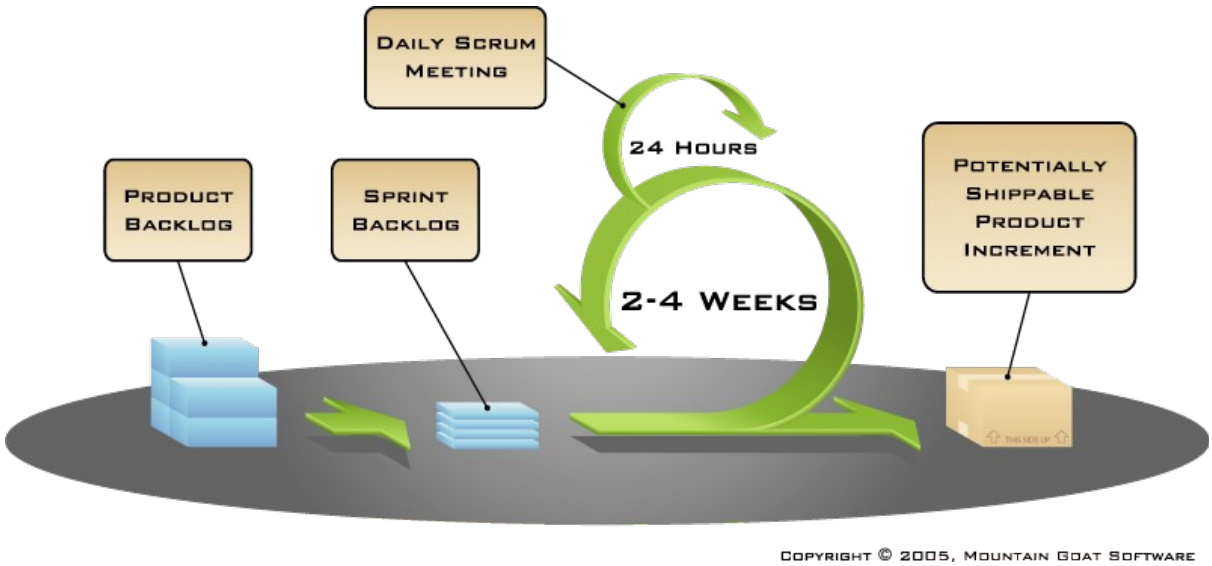

Figure 3: Simple Agile!

We've not yet mentioned the *product owner*. In a sense, this is the person with whom the buck stops! He or she has responsibility for the "big picture" deliverable. In our company, this, in essence, equates to a product manager who is accountable for the product in all its flavors and colors – functionality, quality, supportability etc – as well as its uptake and success in our client marketplace. The product owner is an important member of the scrum team. Oftentimes, the product owner – and, indeed, the scrum team as a whole is supported by a project manager who looks after and supports the higher level aspects of project progression, budget, liaison with and reporting to executive management and resource managers.

As mentioned above, this approach does, in principle, allow the potential for very frequent delivery of a product – i.e. incremental releases of a product very often. With some software products in the world, this might be fine. For us, it's not realistic – largely due to the "overheads" that we have to bear associated with the rigor of testing and documentation that we must do. Hence, our product release cycles are of the order of 6 weeks to 6 months depending on the scale of the product as well as the impact that a new release has upon our clients in terms of their need to validate it within the context of their business.

WHAT'S NEXT?

Whilst we've come a long way, we still have a way to go – even in terms of our existing product base, let alone our overall agility.

Our first priority today is to get all of our product teams to adopt our agile methodology that we've outlined above. This will not only enable us to further enhance our approaches but, importantly, we will be able to formally retire our legacy way of working – along with the SOPs etc associated with that legacy. Hence, whilst we will maintain flexibility, by definition, across our product teams, our staff can have a single, over-arching point of reference.

Second, there is the matter of *technical debt*. We have not mentioned this thus far in this paper and it's not a topic for in-depth analysis here. Suffice it to say that we must attack this debt of long-standing defects and architectural inadequacy but recognize that such debt will always be present – and, indeed, that it's desirable.

Thirdly, we intend to strive for the adoption of agile approaches at the broader level within our company. It's fair to say that the impetus for adopting agile originated within our functions associated with software development. Over time, it broadened out to our testing, product management and other functions. We will extend it further into our operations and support spaces. And, we also intend – and are currently actively pursuing – using agile approaches in the other functions and activities associated with "bringing product releases to market" including, for example, our training (to users and others), our product marketing and even our legal/contractual activities.

With these forward-looking initiatives in mind – and building upon the learning culture that we described earlier – we've formed an *Enterprise Transition Community* (ETC) to drive such change across our company. The remit of the ETC is illustrated in Figure 4.

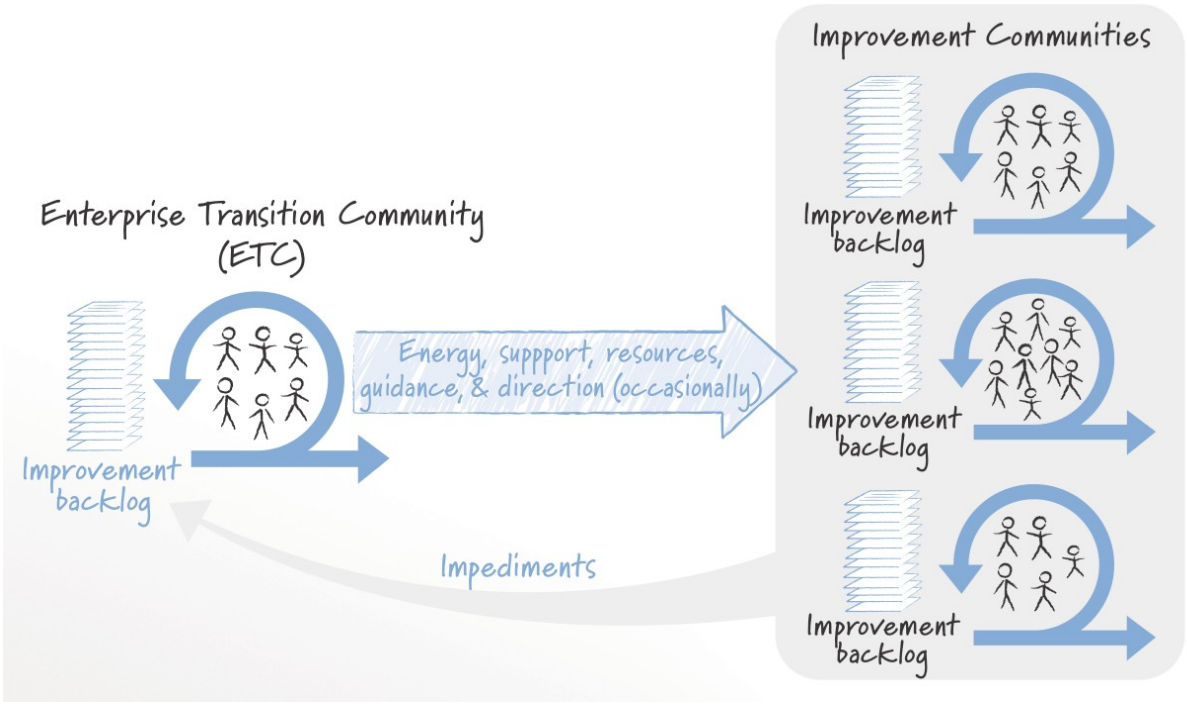

Figure 4: Improvement

CONCLUSION

It's somewhat difficult to draw conclusions per se at this time. Except, perhaps for us to ask ourselves the following types of questions:

1. Are we faster to market?
2. Are we building the right thing?
3. Are they higher quality?
4. Why do we still get defects?

Our clients would almost certainly have answers to these questions. But our own view is, respectively: Yes, Yes and Yes (to the first three). On 4, our response is that it's inherent in the nature of what we do – i.e. we do not – and cannot - build perfect software. But what is certain is that we are far, far better positioned with our agile approaches of detecting (and fixing) defects before the product is release and we're far, far better positioned to be responsive to such defects faster and more effectively after the product is released.

There is no status quo!

RECOMMENDED READING

Succeeding With Agile – Software Development Using Scrum, Mike Cohn, Addison-Wesley, 2010
GAMP 5 – A Risk-Based Approach to Compliant GxP Computerized Systems, ISPE, 2008

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at:

Medidata Solutions Worldwide
Harman House
1 George Street
Uxbridge
UB8 1QQ

United Kingdom
Email: thewer@mdsol.com & asmith@mdsol.com
Web: www.mdsol.com