The Agile Approach

By Kevan Lyons
Business Systems Analyst
Conoco Phillips Australia Pty Ltd.

Table of Contents

Introduction

A Question of Balance

People, Community, Teams, Management, Processes, Products … the list goes on, but the underlying foundation must be balance. Without it, we will topple.

The same can be said of system, or software, development projects. We cannot seek to succeed in a project without taking everything into consideration and to balance everything in consideration.

Coaching a sporting team is an analogy that one could easily compare to the Agile methods and subsequent discussions by Jim Highsmith in his book '*Agile Software Development Ecosystems*' (2002). In fact, the term 'coach' is one that is being more commonly used in regards to the management of a team outside of the sporting world.

To successfully coach a team, the coach must first set down the goals they wish to achieve by the end of the project, or, in the case of the sporting team, the playing season. A coach must then select a team in order to reach those goals. The coach will not have access to unlimited resources here, and must choose the best players from those they have available.

This is true of a software development project, as the best 'players' may be on someone else's team. But a team is not simply a collection of individuals; it is the utilisation of the abilities of those individuals that brings out the best in the team as a whole. The often quoted notion that 'the whole is greater than the sum of its parts' applies to the Agile methodologies emerging today.

Then comes the plan. Planning takes in many considerations. The team must be balanced, with the right number of attackers and defenders. However, the opposition must be considered in the plan.

The coach must ask, 'What team are we facing this week?' and 'How will we adapt our game plan to match, and ultimately, beat theirs?' These are questions that should be asked, and should, at least, attempt to be answered.

We can see immediately from just these few questions, that the plan cannot simply be set at the beginning of the season and retained throughout. There are too many outside factors. Player availability, injuries and facing a different opposition every game are the most obvious.

The coach must, therefore, be Agile. They must be able to adapt to a constantly changing playing environment as the season progresses. Some factors are known or can be anticipated, but the underlying factor is his/her 'responsiveness' to the changes that present themselves in order to be successful.

> "Just as winning is the primary measure of success for [the coach], delivering customer value (however the customer defines it) measures success for the Agile project manager."
>
> - Highsmith, J. 2002 (http://www.stsc.hill.af.mil/crosstalk/2002/10/highsmith.html)

This paper will discuss the emergence of the Agile systems development methodologies that have gained popularity since the late 1990's. As Tom de Marco writes (cited in Highsmith, 2002, p.xv), "[t]he Internet changed everything, and only those who were ready to change quickly with it would prosper."

This work will discuss the movement away from the traditional process focus and towards a product focus, as it draws a parallel with the 'Coach' and sporting team alongside the ideals of Agile methodologies and those that support them.

Process Pressure

Anyone who has participated in a team sport will have come across at sometime in their life a coach who has every manual, guide, and 'how-to' book on the sport that has ever been written. On top of that, add the seemingly endless stream of files and folders containing print outs of the field outline and tactics. Of course, the reverse is equally as frustrating. Not so much as a rulebook or a clipboard in site will leave a team wondering if they have a coach that really knows what they're doing.

The two extremes are exactly that – extreme – but that is not to say that they do not appear in real life, both on and off a sporting field. It appears that in many walks of life, rigorous process documentation and the lack of documentation of any kind are scenario's most people would rather avoid.

A school of thought is emerging that would sway away from the pressures of process. Traditional methodologies were 'documentation heavy', where the emerging Agile methodologies are of the view that the focus on process can often be detrimental to the success of the project. This is not to say that doing away with them all together is preferred, rather, if promotes reducing the pressure of process and focusing on the product.

Here entails the question of balance. 'How much process do we need?', 'What do we keep and what do we discard?' Tough questions for a coach of any team, a how will they know if the discarded processes will later be needed?

Jim Highsmith has coined the phrase, "A Barely Sufficient Methodology" (Highsmith, 2002, p.xxvi) which effectively means streamlining the methodology to focus on the practice rather than the process, but perhaps the best explanation here is his own:

> "There are two reasons to pursue barely sufficient methodologies: value and innovation. Streamlined methodologies concentrate on those activities that create value and ruthlessly eliminate non-value-adding activities." (Highsmith, 2002, p.xxvi)

It is more a question of discarding those processes that may slow the team down. A coach wouldn't ask his team to sit for hours watching a video of endless team tactics or training drills, they would have them out there on the field putting those tactics and drills into practice. But how does the coach relay this to the players? – By a certain level of process of course, but to what extent?

Team Selection

We are lead back to the team selection – have we picked the best team? If so, we can fairly assume that the team will step out onto the field and put the tactics into action, or complete the drills with little instruction, but no team will be able to do so with no instruction at all.

A team assembled of "under-talented" (Highsmith, 2002, p.271) players may indeed require more instruction or they may not succeed. Jeff De Luca (cited in Highsmith, 2002, p.269) and his experience with a "large Singapore bank" demonstrates this clearly. He followed a team that failed after 2 years on a project, with a team of his own, consisting of more talented players and was able to successfully deliver a quality product to its customer.

What developed out of that, and several other projects that Jeff De Luca was involved in, in the late 1990's, and also his work with Peter Coad, gave rise to a new methodology, an Agile methodology, called Feature-Driven Development or FDD. De Luca's 'Singapore' team was formed with talented players and delivered the product feature by feature over 15 months, (Highsmith 2002, p.271). The parallel to the sporting team is evident, the features being delivered being likened to games won week-in and week-out.

The team preceding that of De Luca's, did not manage to post a win – ie: deliver a feature – as they were too caught up in developing the plan and were, in the end, relegated. The team planned for each game and not for the whole season at once and it was successful for them.

Feature-Driven Development (FDD) is an Agile Methodology – or "Ecosystem" (Highsmith, 2002, p.xxiii) – encompassing more than just the processes involved in Systems Development. FDD relies on the belief that "software development [is] a decidedly human activity." (Highsmith, 2002, p.271) It relies on the inclusion of talented players on the team and good processes that don't hinder the work of the team. It does acknowledge that this is no guarantee for success, but it surely increases the probability of it.

The very nature of FDD, the simple approach to processes, and the inclusion of quality 'players' on the team should be enough to be successful and move up to a higher grade. The 'grade' in the systems development world may to bigger and better projects.

Regular Scoring

Much like a coach wouldn't want his players to score all their goals at the end of a match, there is no real need to wait to the end of a project to deliver working software to the customer. Waiting until the end could result in one of two things … the score is not enough, and they've been outscored by the other team. Or, time will run out and the buzzer sounds before the score is posted.

Why wait until the end? Many changes or additional requirements not specified in the original requirements specification appear during the course of the systems development. If there is no way for the customer to relay this to the development team, then there will be problems. The developers may have to rewind their product to accommodate the late changes.

However, if they were to meet with the customer on a more regular basis, as Agile methods recommend, changes to requirements are much easier to build into the development project.

Time Outs

Ken Schwaber's Scrum methodology emphasises short, regular meetings – "daily or every other day" (Rising, 2002, p.42) – and this may seem like a pressure of it's own, a meeting every day, but the contribution to the sense of team is invaluable. Three questions are asked of each team member:

> "1. Relative to the backlog, what have you completed since the last meeting?
> 2. What obstacles (if any) got in the way of your completing this work?
> 3. Relative to the backlog, what specific things do you plan to accomplish between now and the next meeting?" (Rising, 2002, p.42)

Other methodologies also encourage regular, and short, meetings in order to keep them productive and not a waste of time for team members and project managers. A team may come off the field for a 'time out' or at half time and have a meeting of sorts to review the previous play or period and plan quickly for the next. An effective coach and team will be able to do this successfully in the time allotted.

However, this may not appear evident until the team actually takes the field again. There is no time available for a full analysis of the previous play, nor a chance to plan for the next in great detail. If changes are required, the coach (and team) must identify them quickly – this may have happened in the field – and make the suggestions for how they will change their approach in the next period.

During a game, the coach will take advantage of these 'time outs' to give on the go instructions and tactics to his team members. Without them, the team may be heading on a course that may not produce a winning result, the team must be as adaptable to the changing conditions *during* the match, as they are in their preparation for it.

Of course, reference to the period in a sporting match is likened to the individual cycles in the systems development process. And as in a sporting match, the individual cycles in the systems development process may, and most often will, be repeated. The team will not have the luxury of time to analyse the previous cycle in today's current turbulent marketplace.

Another Agile method of software development is Catalyst, which takes it's underlying principles from that of XP (discussed further in the next section) and "yields teams able to manage and adapt to change." CC Pace Systems incorporates the use of Agile Project Management (APM) and Usage-Centered Design (USD) to develop an Agile methodology, tried and tested on at least one Fortune 500 company. (Agile Development Methodologies: a Catalyst to Success, 2003, pp.3-5).

Catalyst provides some additional weight to the theory behind Agile methods that conditions are likely to be unpredictable and upfront planning should be kept to a manageable, or minimal level to allow for adaptability to change. Catalyst uses "time-bound development cycles" to manage 'scope creep', and promotes "[o]pen, frequent communication between customers and developers…" across "…a dynamic project and team environment." (Agile Development Methodologies: a Catalyst to Success, 2003, p.7).

Scalability & the Burden of the Individual Developer

FDD is not alone in it's ideal of producing working software, or features, during the course of the system's development. eXtreme Programming (XP), introduced by Kent Beck in 2000-2001 (Highsmith, 2002, p.297)

XP contains elements that are similar to that of FDD, in that Story Cards are developed – much like FDD or ASD "Features" – and it is also geared towards quality, flexibility and speed, (Highsmith, 2002, p.297)

As a proponent of XP, Sanjay Murthi writes that "[g]ood management and reporting systems help detect issues before they become serious enough to cause major delays." (New Architect, 2002, http://www.newarchitectmag.com/documents/s=2412/na1002e/index.html) This obviously leads to quicker end results by helping to streamline the system's development. The coach will have in place processes or systems to detect injuries and problems quickly and enable them to be rectified as soon as possible and get his "best team" back on the field.

Traditional methodologies often rely on team members having to complete a substantial amount of documentation – "Documentation should be a working by-product of the entire systems development effort" (Whitten, Bentley, 1998, p.76). This sentence alone speaks volumes – and appears to suggest that the developer should produce volumes!

The individual developer must feel a sense of burden when it comes time to produce the documentation for a systems development project using a traditional methodology. I propose that the energy and enthusiasm mentioned in countless articles written about successful Agile projects must partly stem from the fact that, generally speaking, the Agile methods require far less documentation than their traditional forefathers.

The individual developer must welcome introducing an Agile method as their burden of documentation will almost certainly reduce. This means a developer can get on with just that, the development!

Team Players

But what about the burden of competency? A critical factor to a development teams success on a project is the competency of the individual, (Cockburn, 2001, p.133). For many team members, this is a burden in itself. How this part of the burden is perceived, is partly dependent on the methodology chosen and partly dependent on the team members self confidence. The coach and each of the team members must have confidence in their ability to get the job done and done well. In addition, the coach must try to foster an environment that will encourage this confidence in the team members.

Mike Beedle's e-Architects, Inc. have developed XBreed – or Cross Breed – a mixture of components from XP, Scrum and other ASDs and "the result of developing multiple applications and shared components as fast as humanly possible." Beedle, 2001 (http://www.xbreed.net/)

Laurie Williams discusses XP extensively in her work at the University of Utah and advocates that "… XP developers would never spend extended periods of time on any one development practice, as in a waterfallish process." (Williams, 2003, p.16) Quickening the processes in a systems development project does not necessarily mean just brushing over the important things. On the other hand, it actually promotes iterative and quick development, which leads to quick feedback and the ability to jump onto bugs or issues as soon as they arise rather than waiting for a designated review period.

A common thread across several of the Agile methods is the emphasis on 'teams' and 'collaboration'. Not only is this encouraged, but encouraged to occur often, sometimes daily, as with Scrum, or weekly, as with Jim Highsmith's Adaptive Software Development (ASD). Collaboration is not a new concept of course, and Rapid Application Development (RAD) "user-centered approach" (Whitten, Bentley, 1998)

clearly expresses the importance of collaboration with the users. After all, if the users are left out of the picture, who provides the user requirements?

Agile methodologies promote holding short, regular meetings in order to make efficient use of time and the other resources being assigned to the project. To the individual developer, this must appear as a measure that will also help reduce the burden. Shorter meetings, more often, mean that the iterations are smaller and the chance to catch bugs or issues is undoubtedly improved.

The Big Arena

An often asked question though, is 'what about the larger project?' Are Agile methods appropriate for the bigger project?' It is being proven over and over that the answer to these questions is yes! XP uses a concept called 'Pair Programming' which takes software inspections to a different level – one of "continuous learning", (Highsmith, 2002, p.302)

Fred Brooks discusses the "added burden of communication" and includes as part of this burden, the training of the team members and also their intercommunication. He writes that the larger the team, the longer the time for training and the higher the complexity of the intercommunication (Brooks, 1995, p.18). This leads to a requirement for additional coaches, and, as you will also find in the sporting arena, they become more specialised depending on which teams and tasks they are assigned to.

Purely via this edict, increasing the number of people on the project actually adds to the schedule. It is often the mistake that project managers make when time overruns are occurring – add more people ! This does not mean that a large project cannot succeed, as many of them do, and these are merely discussions and observations made to assist those undertaking a large development project.

One Agile methodology that does consider the scalability to large projects specifically is Crystal. Developed by Alistair Cockburn in the late 1990's, the Crystal Family includes a method of selection that is based on both the number of people involved (size) and the criticality of the system (project). "As the number of people involved grows, so does the need to coordinate communications." Highsmith & Cockburn, 2000, (http://www.cutter.com/freestuff/ead0002.pdf)

Crystal does recognise though, the fact that large projects are not without their risks. One may argue that as the project gets larger, the documentation required by Crystal has the effect of moving back towards a more traditional approach.

But is this what the Agile movement seeks to avoid?

Not necessarily. The supporters of Agile methodologies do not discard the ideals of traditional methodologies – they have formed their own out of the experiences they have had with these traditional methods after all. They have sought to improve the systems development process by incorporating what is seen as good from the old and delivered agility to them to meet the requirements of today's turbulent environment of rapid and unpredictable change.

They have regard for all parties to a systems development project, and consider reducing the burden on all involved by streamlining the processes in place and focusing on the product and collaboration with the customer. Improving the communication within the teams is also of high value to the proponent of an Agile methodology.

Conclusion

No Silver Bullet

> "It [is] predicted that a decade would not see any programming technique that would by itself bring an order-of-magnitude improvement in software productivity."
>
> (Brooks, 1995, p.vii)

Brooks bold "No Silver Bullet" statement was made in 1986 and it was indeed provocative, but a decade did pass since he made it and his prediction is now widely accepted as one of truth. It is also true that the decade following is also unlikely to provide a Silver Bullet answer.

Agile methodologies were developing as early as this statement was made, perhaps even earlier, but they did not really bare the 'Agile' name until the signing of the Agile Manifesto in February of 2001. 17 highly regarded individuals from the software development industry came together to form what is now termed 'the Agile Movement.' (Highsmith, 2002, p.xvii)

These proponents of the Agile methodologies all believe that the traditional methodologies of heavy documentation and rigorous processes are in many cases no longer appropriate. The new systems development environment where the internet has become all encompassing is now more unpredictable and rapidly changing as than ever before and Agility is now the key factor.

The Agile methodologies that exist today are by no means Brooks' "Silver Bullet", and those that support them do not suggest that they are. The factors involved in software development are far too wide and varied for just one methodology to be the answer in today's turbulent environment.

The use of any of the existing or emerging Agile methods does not exclude the use of another. It is becoming increasingly evident that a development project can, in fact, utilise elements of more than one methodology, as demonstrated by Mike Beedle & e-Architects' 'XBreed' and CC Pace Systems' Catalyst.

The focus is clearly shifting, from processes to practice, from heavy documentation to iterative releases of working software, from price fixing and contract negotiation to collaboration with the customer and from following a plan to adapting quickly to change.

> *And that is the set of underlying philosophies that form the Agile Manifesto, and what better way to wrap up this discussion, than with the Agile Manifesto itself.*

The Agile Manifesto

Individuals and Interactions over *Processes and Tools*

Working Software over *Comprehensive Documentation*

Customer Collaboration over *Contract Negotiation*

Responding to Change over *Following a Plan*

(Highsmith, 2002, p.xvii)

References

Beedle, M., 2001 'Welcome XBreed', December 2001. Retrieved September 17, 2002 from http://www.xbreed.net/

Brooks, F.P.Jr, 1995, *The mythical man month,* Addison-Wesley, Boston.

Cockburn, A., 2001 'Agile software development: the people factor', *Computer*, November 2001, IEEE.

Highsmith, J. 2002, *Agile software development ecosystems,* Addison-Wesley, Boston.

Highsmith, J. 2002, 'What is Agile software development?', *Crosstalk: the journal of defense software engineering,* October 2002. Retrieved August 21, 2002 from http://www.stsc.hill.af.mil/crosstalk/2002/10/highsmith.html

Highsmith, J., Cockburn, A., 2000 'Crystal light methods', *Agile project management advisory service white paper,* February 2000. Retrieved September 24, 2003 from http://www.cutter.com/freestuff/ead0002.pdf

Murthi, S., 2002, 'Scaling Agile methods', *New Architect: Critical Decisions,* October 2002. Retrieved September 17, 2002 from http://www.newarchitectmag.com/documents/s=2412/na1002e/index.html

Rising, L. 2002, 'Agile Meetings', *STQE,* May/June 2002, STQE.

Whitten, J. & Bentley, L. 1998, *Systems analysis and design methods,* Irwin McGraw Hill, Boston.

Williams, L. 2003, 'The xp programmer: the few-minutes programmer', *IEEE software: focus,* IEEE Computer Society. Retrieved September 22, 2003 from http://collaboration.csc.ncsu.edu/laurie/

-, 2003, *Agile Development Methodologies: a Catalyst to Success,* CC Pace Systems. Retrieved September 22, 2003 from http://www.ccpace.com/resources/documents/agileassessmethodology_whitepaper1.pdf