

Nigel Mossman considers what organisations could do when they need a project management process but find traditional methodologies impossible and need something more substantial than XP.

Back in the 1990's when Total Quality Management (TQM) was seen as the Holy Grail for delivering quality, I was introduced to the idea that a software application is pretty useless (assuming it meets the business need!) if it lacks (for example) training material, documentation, trained support technicians and cannot be easily implemented. The whole product is everything needed to turn the application code into a deployed system delivering benefits to business users. In my view many Agile methods, (e.g. XP and SCRUM) do not cover the entire Project Management Product.

Many companies still have too much focus on the coding and testing stage and fail to appreciate that the quality and adoption of a final deliverable is impacted by the work that is not undertaken by the Engineering Team.

Those members of the Agile community that understand Prince 2 will know that it's heavily routed in command and control. The deliverables are fixed at the start of each Work Package (i.e. scope, end date, quality etc). The Project Manager is also at liberty to execute a work package (Prince 2 Process "MP2") using any methodology. The ability to co-habit with Agile depends on the amount of tolerance granted to the Project Manager by the Project Board. Limited or no tolerance to vary scope and priorities etc turns execution of MP2 to, at best, a good RAD delivery and at worst a hack. Prince 2 does offer an environment where the Project Manager is forced to consider all aspects of the overall project delivery (e.g. user documentation or hand over to a support team) and divide these into appropriate stages and work packages.

DSDM provides a good half way house between Prince 2 and other Agile methodologies. The Feasibility and Business study phases provide an opportunity to consider the whole delivery early in the life cycle, while the Functional Design and Design/Build allow for iterative construction of the application and scope to respond to changing business priorities. Implementation is a distinct phase, with a Post Project review to provide feedback for future projects. The methodology is not prescriptive over the length of the pre coding stages; the team could conclude the work in an afternoon if all of the facts are known!

DSDM has recently been overhauled, and I have to say that I am struggling to see how the oddly named Atern variant improved over the previous version. I will be interested to see how DSDM develops and how it will build upon established relationships with Prince 2 and TickIT.

What often surprises me is that many companies adopting Agile have little if any general project management process. This is normally as due to having high volume quick turnaround projects, or due to growth from a small to medium sized business. I have found that the introduction of any formal methodology worries management and gives engineers nightmares about endless amounts of paper work.

For these process phobic businesses, what should a light weight project management tool kit contain?

Getting Started

What does the software need to deliver? This isn't a story list, more a short summary of the

application's intended purpose. I recently learn a useful technique – CATWOE where C stands for customers, A for Actors, W for World View (i.e. why are we doing this), O is for Owner and E is for Environment (the business Environment). Each element of the CATWOE requires a sentence, at most a paragraph.

What is the value to the business of this application? A short summary of quantifiable financial benefits to the business. In some cases – e.g. changing systems to handle legislative changes – cost savings may come as a side benefit and not as a justification for the project. However, a project may also be an enabler to take advantage of a new business opportunity.

At what point does it become too expensive? This is not a question about budget – this is a question about assessing how much a business can afford to spend on a project before its costs outweigh the benefit. Where the expected budget is close to the red line on cost, it may allow all concerned to think of alternative (cheaper and perhaps better) solutions.

Besides the software, what else will be needed to get the software into production? Companies often only think about getting software coded, little consideration is given to the necessary additional work to take code from a development environment and deploy it the real world. Issues include data conversion, links to other systems, user manuals, documentation, training, hand over to a support team. As well as considering the what, consideration must also be given to the who, their availability and costs.

What is the Software Platform and Application Architecture. Another common mistake is not to identify the target deployment environment and underlying architectural components. Windows or Linux, MySQL or Oracle – Java or C#, new “green field” code or extension to current application – use of frame works (which versions).

Best Guess on time and staff. An estimate is just an estimate – I have found that companies that claim to have swallowed the Agile message still treat estimates as binding contracts. What is important is that actual costs of staff to be used on the project are known at the beginning. Setting a budget for permanent staff and then using contract engineers at double the price is a sure fire way to bust the budget. When estimating, remember that engineers tend to under estimate the effort, not all requirements are known – where ever possible look for projects that are similar to the one your costing to see what how they turned out and do remember that the amount of time taken will vary according to the skill of the engineers, their understanding of the target architecture and to an extent their understanding of the business. The overall project estimate must include the costs of producing the whole product – i.e. not just the coding effort! An amount of contingency must be added, the size of this number depends on the risks, issues and assumptions that are known about at the time of the estimate.

One way of improving the coding estimate is to undertake a brief “spike” using engineers that will be allocated to project when it launches. Management can be unwilling to finance a short technical spike – perhaps two to four weeks in duration, however it can significantly improve estimates on the project and help to mitigate (or confirm) issues and risks.

Key Risks, Key Issues and Critical Assumptions. Document these at the start of the project. Not all of these problems will be known about at the beginning, but for those that have been identified suggest potential mitigation (for risks and issues) . Any critical assumptions must be noted as these form the basis of upon which the project has been constructed. I have only suggested documenting the key and critical items. In my experience, executive management tends to give this section area

scant consideration so it's worth only including items that need to be highlighted.

The final upfront piece of work might be a draft story list and an indication of the order (priority) in which the stories will be delivered. An indication of the minimum stories that constitute the first fully usable release might also be an advantage. This helps to get the early iterations off to a more organised start.

Other sections could be added – for example – an “Agile Checklist” showing how the principals the project plans to use – or a section on communication – how the project plans to communicate with team members, interested parties within (“stakeholders”) the business and executive management.

The above may look like a large amount of effort to prepare and (you might think) a large amount of written material. It doesn't need to be. Only write down that which adds value – and remember that both executive management and project team members will have to read it; they will not thank you for writing War and Peace when the key pieces of information could have been delivered in a few sides of A4.

During Build and Test

Daily stand up meetings and end of iteration reviews are a key part of any Agile project. At the End of Iteration meeting, the team should review the key performance indicators (KPI). The team should set their own, however, the KPI's I have commonly used include:

- Story count /Story point count (total, completed, retired, new, work in progress)

- Bug Count (total and number currently open)

- Code Coverage (taken from your build system)

- Engineers view of code quality

- Number of builds in the iteration and of these, how many compiled first time

- Size of code base (measured in lines of code)

- Users view of the application - is it stable, does it meet the business need?

A meeting with the Project Sponsor, Programme Manager or other members of the executive management team must take place on a regular basis – at least one per iteration. The Project Manager, BA (if one has been appointed), Lead QA, Lead Engineer and the business representative should attend these review meetings. As well as considering the KPI's (shown above), the meeting should review the findings from the End of Iteration Meeting and asking:

What more needs to be added to the application to make it suitable for use in the business?

How long is that likely to take given current velocity, resourcing and rate of change?

What risks and issues exist? What action is being taken?

How is the budget looking? How close is it to the red line?

By considering these questions, both management and the project team can determine the strategy for the coming iterations. Also, if the time limit/budget is likely to be exceeded, then action can be taken to bring the project back on track. This could be to reduce the number of stories, increase funding, change the resource profile (for example bring in more experienced resources or provide training to the existing project team; not the obvious choice of adding more people!).

The executive attending the meeting should also understand that they not present to receive reports and issue direction. They must be prepared to take actions to help the project progress. In most companies, Project Managers have little influence on teams and departments beyond their own and will need the executive support to remove obstacles that have proved resistant to the usual mix of sweet talking, horse trading, diplomacy and threats!

Delivering the Project

A good Agile project will have had (or deliberately found) opportunities to deliver code to users outside the development environment during the main build phase. As well as creating a further opportunity for valuable feedback, these deliveries will help to create an installation process and identify installation issues early on.

Post Build and Test activity includes preparation of user training, documentation, pilot process, data migration etc. Although I have labeled these tasks ‘Post’, they can and should start at an appropriate point during the Build And Test phase.

Each of these of tasks groups can be managed in a similar framework to the coding effort. Each task group will have a number of requirements/actions that can be turned into stories. The stories can be organised into iterations, prioritised and tracked.

Each task group will need its own quality measure(s). This should always be easily obtainable and meaningful. For example, user documentation can be given to reviewers to read and compare with the application and “bugs” raised for missing information, spelling mistakes etc. Subjective information from the reviewers can be collected on ease of use etc.

Regular feedback from the team at stand up meetings and end of iteration reviews will assist overall communication and progress. As with the Build and Test Phase, a regular meeting with the Project Sponsor, Programme Manager or other executive must take place on a regular basis. In some respects these review meetings are more important as the project rolls out. As deliverables become more visible to the receiving customer organisation, so the potential for curved balls and political problems increase. A member of the line management team will be invaluable in helping to resolve (or mitigate) these issues.

Post Project Review

Once the project has concluded (or at interim stages for a larger project with several deployment opportunities), hold a Post Project Review meeting to consider how the project has progressed, what worked and why, what could have been better and why? Ideally, all project team members should be involved in this process, however this will be difficult to achieve in most companies. At the very least, representatives from each of the teams/disciplines who delivered the project must be involved.

Conclusion

As with any process, organisations should consider how they work and create a framework that builds upon their strengths and mitigates weaker areas. This article is intended to generate ideas

rather than present a definitive blue print.

By discussing the activities prior to and following on from the coding effort, I have highlighted that many Agile methods do not cover the entire project life cycle. By drawing on some of the concepts and principals in formalised methodologies **and** Agile, a light weight and adaptable framework can be created to manage the whole delivery cycle. Managing the whole delivery cycle decreases risks, gains maximum benefit from the Agile process and increases the chances of achieving a well received, high quality deliverable.

© Standard Data Systems Limited, June 2007

-end-