Appropriate Process

Mature Agile White Paper

Introduction

The purpose of this white paper is to explore aspects of projects which add complication and exacerbate project delivery, how to overcome the problems with minimum fuss and cost and, as a by product, how to maximise the reader's expertise.

Why am I bothering to do this? Surely running a standard process model, like Scrum or DSDM will solve the problems: These processes are sound and have really improved software development. Yes, but they are not suitable "off the shelf" to work effectively in all software development project profiles, thankfully the processes are not carved in stone and because they are goal based, adapting the way of working is not difficult.

Recommended steps with their associated pro's and con's are discussed and these steps can be implemented regardless of the process model being used. How to integrate the changes into a specific process model will not be covered.

Project Profiling

What we mean by a Project Profile; is the environment and conditions that a project is delivered in, and the goals that the project has to deliver. There follows a partial risk analysis on 2 Project Profiles, project A and project B, these are vastly different, firstly lets highlight the major risks related to each project profile:

Project A : A Team of 15 developers, comprising; 3 user interface developers located in London, 6 Database / backend developers in Bangalore, and 6 Security/Network developers in San Jose CA, there are also 5 Test Engineers located in Mumbai and 1 Systems Analyst based in London. The Goals of the project are to deliver changes to an Electronic Payments System to comply with new payment security standards. The requirements are spread across all teams. The payments system is deployed in 1 million stores in Europe and interfaces to 100 financial institutions. The project delivery needs to synchronise with a PCI:DSS consultant for a validation of compliance session prior to go live.

Project A : There is a significant level of risk here, so changes will be required to the Agile process model if the project is to be completed and delivered successfully.

Project A : A Software Development Project delivered by a 2 developers and 1 test engineer in an office in the middle of London. The project is delivering small defined changes to an interactive web site for a customer also based in London.

Project B : In contrast, this project profile is relatively risk free, so as long as the team ensure the requirements are understood, and keep the customer updated with progress, then there should be no problems working in an unmodified Agile Process Model using SCRUM or DSDM etc. The rest of the document will concentrate on aspects likely to be encountered in more elaborate project profiles Project A profile types.

Author : Neil Harrison **Software Development Consultant** Director of BelieveIT Ltd (UK registered company).

Project A : cont. ....

There is an obvious difference of scale between Project A and B that will add its own level of complication, but that's not the whole story - Here are some specific situations in Project A which need to be addressed for it to complete successfully: the list is by no means exhaustive;

Table 1 : Profile 2 , potential problems.

	Description
Problem:	The requirements are very detailed and need to be implemented fully.
	Misunderstood or incorrectly implemented requirements will result in non compliance to the security standards and the inability to go live.
Problem:	Teams' skills are split across multiple locations and time zones.
	The team will be unable to collaborate on a daily basis, in the way that a co located team would; due to location and time zone differences.
	Development difficulties and questions will be difficult to resolve.
Problem:	Functional areas are delivered by different teams in different locations.
	Database people are in Bangalore and security people are in San Jose, so details of the functional interdependencies to be delivered by each team will be difficult to communicate as there are no common working hours due to the time zone differences.
Problem:	Rework due to misunderstood implementation requirements will be very expensive due to the size of the team and the rework may be required to be completed by multiple teams.
Problem:	The testing required is significant and must be defined and reviewed prior to commencement of testing to ensure that the tests cover the security requirements.
	Insufficient testing could result in non compliant release, that could not go live.
	Insufficient testing could result in an unstable release, which could impact the transaction processing for 1 million stores.
Problem:	Timelines must be met, for completion of the changes and testing, in order that compliance testing can be achieved.
	Changing the compliance testing timeline will result in a delay as the compliance testing will need to be re-planned with the compliance consultancy.
Problem :	Project will be of significant length.
	Project profile needs to be such that delivery is efficient and progress is reported accurately, as it is a critical project.

It is clear that the risks for the type of profile Project A is will demand that steps, and ways of working, be put in place to address these challenges. How this can be achieved within the Agile Process being adopted follows...

Solving The Problem

Project Profile Risks

In the previous section we identified potential problems for Project A, now here's a look at the various aspects of Project Profiles in general, and how processes can be modified to resolve those problems encountered ;

Profile Element	Problem
Project Size:	⇒ Is the project more than 2 weeks development work ?
Team Location:	⇒ Are the Teams Co located (in the same office) ? ⇒ Are the Teams Distributed (in different offices) ? ⇒ Are the Teams in different time zones ?
Project Complexity:	⇒ Are the project requirements simple to communicate ? ⇒ Does the project require a complicated and strict technical solution ? ⇒ Does the project depend upon meeting strict process requirements ?
Requirements Status:	⇒ Are the requirements fixed and defined ? ⇒ Is some prototyping or development required to complete the requirements ? ⇒ Will the requirements be evolving over the duration of the project ?
Project Scope	⇒ Are the deliverables incremental changes to an existing product ? ⇒ Are the deliverables the creation of a new product ? ⇒ Does the project deliver API's or components that interface to other components (servlets, enterprise beans, backing beans etc.) ?
Project Impact	⇒ Is the project part of a long term product that will be supported ? ⇒ Is the support team separate to the delivery team ?
The Project is dependent on External Resources:	⇒ Is some of the functionality delivered by an external team, a team that is not designated as part of your team (e.g. maybe outsourcing pieces of work or a different department altering the core) - and you have to request specific changes to be made ?
Team Skills :	⇒ Are all team members familiar with the source code being modified ? ⇒ Is the team composition dynamic (i.e. individuals on a team may come and go throughout the project duration) ?

The next section will be looking at the project profile elements listed above, identifying the associated risks, risk impacts, and then putting forward suggestions as to how to mitigate the risks: in a cost effective way.

Risk Mitigation

Project Size :

The Project Size is relevant to the Project Process with respect to the setting of targets and being able to report accurately on progress. Scrum uses the 2-3 weeks sprint model in order to provide achievable targets and to keep the teams focussed on the delivery of the sprint requirements.

Problem :
However, on larger projects it is not efficient to stick to the 2-3 week sprints as the development's delivery is more efficient if the development cycles are sized on the delivery of the user stories (not time boxed), once the story (not the sprint) is code complete then testing and bug fixing commences.

Fix:
In this scenario a method of setting targets and tracking is required so that the team members have clear objectives and progress against these can be tracked. Clear objectives are important in large project deliveries, see the Agile Project Planning White Paper for Suggestions on Project Planning.

Part of the planning process is the estimating of the tasks required to deliver the requirements, The Agile Estimating Spread Sheet provides some help on this.

Planning is all about setting expectations and setting goals. Plans are built on the estimates provided by the team: Any team will tell you, it is impossible to estimate accurately on all tasks, so, we add a confidence level indicator that affects the timescales being put into the plan.

Team Location :

When teams are located in the same office and are working together then there is normally no need for any process changes to the standard Agile Models. If teams are distributed, in different offices and/or time zones things get more complicated.

Problem :
Teams split over multiple locations can suffer from; poor communications of requirements, teams half solving problems in isolation, and misunderstanding on goals etc. These issues can result in functional issues being identified after development work is completed, requiring rework from one or multiple teams, Project completion can be delayed significantly. Teams may suffer demotivation because they have put loads of effort in and had to re-write code, hostility between the teams may ensue, as it is always "the other teams fault".

Fix:
A fix for this is a lightweight Agile Technical Design approach which has real benefits, we are not talking about a full System Design covering every aspect of the application but a design that describes the interaction required between components to complete a User Story and results in component interface agreed between the teams.

For Example : Where a remote team is delivering a backing bean for a Web application, the Web Team can walk through User Experience flow and then in collaboration with the Back End team define the functionality required in the backing bean, along with the API and the information returned from methods. The backing bean team can do the same with the Database team.

The technical design model starts with Robustness Analysis and results in a Sequence Diagram that defines the API. It is important to only capture the detail needed to define the API, and not to define functionality that is not part of the interface.

Conclusion :
So for very little work up front, the confusion is removed, all the teams are on the same page and the risk of rework is reduced enormously. A further benefit is that, when the teams walk through the User Experience at the start of the project then issues with the requirements are also identified early in the project, so any problems can be resolved with the customer early on.

Project Complexity :

Where the requirements on the project are straight forward and clearly communicated in user stories then there is, again, no need to move from the Agile Process Model. However, some simple strategies can be employed to help keep on top of complexities.

Problem :
Not all requirements can be fitted comfortably into a User Story, things like performance and response time can be. But, for example, if specifying a requirement to develop a story to a particular compliance standard, then the user story would not cover the detail of the compliance implementation which every member of the team would need in order to successfully complete their work to the appropriate standard.

Quite often there are requirements in addition to the user stories, such as; requirements for load management, expected number of concurrent users, specific technical requirements (such as using specific libraries and technologies), requirements to use a specific look and feel, these may not be specified in each story but they are an overall requirement, that all members of the UI team needs to be aware of.

On some projects there will be architectural requirements, with respect to implementation that the team need to be aware of.

Not implementing the requirements correctly will result in a displeased customer, the project not being accepted or in the case of architectural requirements a product that does not work with the other planned applications.

Fix:
A user story incorporating every known requirement could be written for each story, however this would be a significant amount of extra work! Creating a set of none functional requirements and then graphically linking them to user stories could overcome this problem, see the link below.

Where a technical analysis is needed then a System/Business Analyst will be needed to ensure that the requirements are defined in full, then there are a couple of other approaches to how to capture the information to pass to the team;

1. Use a task management tool like trac, to capture the user story and then enter each of the extra requirements, group them to the original user story task – http://trac.edgewall.org/demo-1.0/
2. Use a UML requirements tool, the EA one will generate web pages the URL of which can be passed over to the team members, this is my preferred approach as the links and detail are more easily captured and described. http://www.sparxsystems.com/products/ea/requirements.html

Conclusion :
To be effective and efficient teams need to have a set of well defined requirements. Ambiguities in the requirements will lead to unsuitable software and demotivated teams.

Requirement Status :

There is Agile and there is Agile: But entering a project without defined goals, even if the project is just a 3 weeks sprint, is not good. How can you know if the work on the project is complete if what was to delivered is not defined?

Problem:
Software development projects are expensive, without clear targets on what is to be delivered, they tend to evolve over extended periods and become even more expensive, possibly never ending. Not having goals can make small tasks seem never ending and lower morale.

Fix:
Not all project steps need to deliver software; a positive approach when there is doubt or uncertainty is to run a Requirements Sprint, which has a goal to define the requirements for the development phase and is time boxed to an acceptable cost. Having the team involved in this prototyping phase is really productive and gives the teams time to investigation new approaches and possibly technologies and really helps in building interdepartmental relationships. See, http://www.sparxsystems.com/products/ea/requirements.html

Conclusion:
Spending the time in requirements sprint is generally cost effective and a really positive move for the delivery team as they can be actively involved in the definition of the development as well as identifying possible issues with the requirements and resolving them before actually starting the project delivery.

Project Scope :

Small incremental changes to products can normally be developed using the standard Agile process model, Where a new product is being specified and developed, this is a great opportunity to define an architecture for the application that will be supportable long term.

Problem :

Where a product architecture is not defined upfront work, and actually evolves during the delivery of the project, then there is little opportunity to organise the architecture in such a what way that ongoing changes will be simplified to ensure that the evolution of the product can be cost effective.

It is not uncommon for products developed to be thrown away after a few releases due to the lack of upfront architectural design. Patches are put in because the application framework is not structured such that additional functionality is simple to implement. This can be for a number of reasons, one of the early warnings of such a problems is where relatively small changes are needed in multiple parts of the application so that functionality is duplicated and therefore cost of modification and defect fixes will increase as the fixes need to be replicated and not every location that needs to be changed can be identified easily.

Re-Architecting existing products is not normally cost effective as there would be significant effort required in determining the current structure and then re-working the existing architecture would normally be a very large amount of effort.

Where External API's are part of the deliverable, some work will be required to document the API's so that the External parties can use them effectively.

Fix:

Where the development of a new product is starting, or the project is providing external interfaces, then it is well worth spending a bit of time defining an architectural framework for the application. There are many guides on the internet as to best practices in architecture.

The following is recommended;

1. Restrict the design to the architecture of the core functionality, avoid defining things like User interface forms etc.... Keeping an architectural model up to date is then achievable, trying to keep a design updated, including behaviours on button clicks etc. is not feasible, it will be out of date in days.
2. By architecture, we mean the "UI layers interfaces to a middle layer that talks to a layer that interfaces to the database", then showing the way interface to the database is extended to access Customer records. Etc.... Not detailed design
3. The goal is that changes to data, UI, or Business logic can be made with minimal changes being required, one standard and very robust approach is using state machines.

One approach I have used successfully to achieve this is the Iconix Process, see http://en.wikipedia.org/wiki/ICONIX and SparxSystems which have some good Tutorials at http://www.sparxsystems.com/resources/index.html

Conclusion:

Building an architectural framework up front, can be a big saving in the future.

Project Impact :

The questions to ask yourself are; Is the project part of a long term supported product, and/or will be supported by a team that is not the development team ?

Problem :
When there is a support team that fixes defects on a software product then normally they have to support a number of different products. One factor that really increases the cost and reduces effectiveness of support in this environment is having no map of the application so the engineers have to reverse engineer the software to fix defects and this can take a very long time, where do you start ?

Fix:
Providing an architectural model of the application, with the Robustness Model and Defined interlayer API's can bring real benefits to the resolution of defects. Because you have a map of the application and can therefore start the investigation into the defect in an appropriate part of the application. It is important that too much information is not provided, such as implementation details of methods or UI elements, as fixing and tweaking will invalidate the details very quickly.

One approach I have used successfully to achieve this is the Iconix Process see http://iconixprocess.com/

Conclusion:
Defining the architecture upfront has many benefits in the right environment, where this is not done and there is a 3rd line support team then generally a document is required at the end of the development to hand over to support. This approach does not benefit the project and takes up developers' time. Creating a product architecture model achieves a better result for support but also improves the product and speeds up the delivery.

Project is Dependent On External Resources::

External resource teams on projects are quite common now: They provide more flexibility in providing resource than internal teams: However, getting the process wrong can be very expensive.

Problem:
Where external resources are involved in a project then it is paramount that the external team are given clear direction on the work to be delivered – the last thing you want is for the external team to be working on the wrong implementation!

Not all, but a fair proportion of external (outsourced) teams have a !we will do exactly what we are told" attitude, they would not flag a problem but would soldier on because that is what they where asked to do : This is mainly a symptom of time and material contracts, not apathy.

Fix :
I have had good success in getting the outsource team to define the requirements to be delivered by having a time boxed sprint at the start of the project - where they work with the customer to define the requirements and agree what will be delivered with the customer. See, http://www.sparxsystems.com/products/ea/requirements.html

In addition to this, I do recommend getting the external team to complete a technical design of the implementation proving an overall architecture for the changes required. Once these items are in place then you can be sure of the functionality and quality of the product you will get from the team.

Giving the external team to do a discreet piece of work, rather than working as part of the internal team is preferable, then you can easily see how they are performing.

Another key item is to ensure the external team gives delivery estimates and that they deliver to an agreed plan. See, the Agile Planning White Paper on the Process Mashup Site. Part of the planning process is the estimating of the tasks required to deliver the requirements, The Agile Estimating Spread Sheet provides some help on this.

One of the major drawbacks of T&M contracts, and why some outsource teams love Scrum, is that there is no fixed timelines for the delivery, so it takes as long as it takes.

Conclusion:
Outsource teams need to be managed much more rigorously than internal teams just to ensure you are getting the optimal value for your money. They don't have a stake in your product so where an internal team may raise issues, a lot of external teams won't.

Team Skills :

If the teams are skilled in the product being modified in the project, a Standard Agile models work fine.

Problem:
When new staff join a project, or team members change regularly, then problems can arise if they are not given clear guidance, they may implement code the way they think is right, there is no reason why it wouldn't be a better solution than the current implementation, but quite often they are duplicating existing functionality – if this continues, the application will undoubtedly be more complicated to maintain and the product can easily end up like a pile of spaghetti.

Fix:
Having a technical architecture diagram can resolve a lot of these issues, the Iconix process is a very lightweight way of modelling and defining the architecture http://iconixprocess.com/ and SparxSystems have some good Tutorials http://www.sparxsystems.com/resources/index.html.

New team members will benefit by working through the implementation with an experienced team member who can hand hold them through the modelling of the implementation and review it once completed.

Conclusion :
Some thought upfront on architecture should bring to light how much time should be spent

Project A Project Process

So, we have looked through the project profile and descriptions of how will this effect the process for the delivery of the Project A.

The requirements are very detailed and need to be implemented fully

- A Requirements sprint will be executed collaborating with the delivery teams at the start of the project. Once the User Stories and the detailed requirements are complete, they will be modelled and made available to the team via a shared server.
- See, http://www.sparxsystems.com/products/ea/requirements.html

Teams skills are split across multiple locations and time zones

- The Technical Architecture will be modelled, defining the story flow and detailing the API requirements, to complete the development.
- The model will be in a 2nd Sprint once the requirements are confirmed.
- See, http://en.wikipedia.org/wiki/ICONIX and http://www.sparxsystems.com/resources/uml2_tutorial/uml2_sequencediagram.html

The testing required is significant and must be defined and reviewed prior to commencement of the testing to ensure that the tests cover the security requirements

- Once the Story and development requirements are complete, the test team will be provided with the requirements to specify the testing required and the tests to be executed.

Timelines must be met for completion of the changes and testing in order to ensure that compliance testing can be achieved.

Project will be of Significant Length

- Plans will be created for the execution of the project and progress will be tracked.
- The plan will include
  - Some time for customer review and rework of parts of the development.
  - Testing definition / planning and execution
- See Agile Project Planning , Agile Project Tracking, Task Estimating Spreadsheet, Sample Plan