

PrinceLite in an Agile World

Summary

This paper explores the difference between an agile engineering method such as Scrum/XP, and an agile governance method such as PrinceLite. It demonstrates how the two disciplines can work well together by allowing agile methods to be more palatable to medium/large organisations that are uncomfortable with an overt acceptance of uncertainty centred around requirements change.

Introduction

It is important to understand the useful distinctions between a project governance method and a software engineering method. It can be argued that to be effective it is necessary to have one of each; governance and engineering. Together this can be termed a project delivery method.

- Governance method: a style of control to measure project progress. Example: Prince2™, PMBOK, DSDM Atern, PrinceLite.
- Engineering method: the style of organisation for the software delivery. Example: waterfall, RUP/DSDM (iterative delivery), Scrum/XP.
- Delivery method: a combination of governance and engineering methods. Therefore it is possible to combine Prince2 and Waterfall, or PMBOK and RUP, or PrinceLite and iterative.

Many organisations fail to implement a unified delivery method. They believe that because they have, for example, implemented Prince2, that this is sufficient. They are unaware of the interplay between governance and engineering.

Project control exists on two levels. There is internal control within the engineering method, and there is external control (management control) represented by governance.

A governance method measures the effectiveness of engineering progress. It can be argued that if the projects are being delivered satisfactorily, then governance is unimportant. This is the outlook of the Agile engineering community.

This paper introduces the fundamentals of agile in the context of Scrum/XP as an engineering method for delivering software.

What is Agile?

Agile is a word that has a particular connotation in software engineering circles. The most famous Agile methods are Scrum and XP both of which are software engineering methods. Scrum/XP are primarily concerned with organising themselves to respond to emerging requirements. These two methods are very similar but not identical (<http://blog.mountangoatsoftware.com/?p=8>). Here we will specifically discuss Scrum.

Scrum is an approach to building software. The word 'Scrum' comes from the game of Rugby where it is necessary for the entire team to move up the field as a unit in order to score. It is appropriate where the culture of the organisation is receptive to an upfront acceptance of uncertainty. Uncertainty is caused by business requirements being only partially understood. Successful Scrum implementations are characterised by small motivated teams, being comfortable with changing requirements, and little traditional project governance.

Therefore an organisation that is familiar with large teams, 'complete' specifications, and significant governance will often be resistant to Scrum principles.

Scrum Process

The process that underpins Scrum is straightforward. The project is divided into units of time known as Sprints. A Sprint is an iteration of time that typically lasts 30 days after which there is a software release that is shown to the stakeholders and evaluated. Based on what they see, the stakeholders may modify their requirements, this is known as requirements churn.

The requirements that are delivered in a particular Sprint are taken from the list of all known requirements, known as the Product Backlog. Those in the current Sprint are catalogued in a subset of the Product Backlog which is termed the Sprint Backlog.

Because changes are to be expected upon the release of a Sprint, it is not possible to do in-depth planning. Therefore only appropriate planning should be undertaken.

In Scrum, team meetings may include everyone, but only members of the delivery team may contribute. Meetings are typically short perhaps lasting only 15 minutes. They always start on time and there is a penalty for anyone who are late. Meetings take place at the same time and in the same place each day.

Each person in the delivery team must answer the following questions:

- "what have you done since yesterday?"
- "what are you planning to do for the rest of the day/tomorrow?"
- "what problems do you envisage having?"

The Scrum leader takes particular note of the problems and works to resolve them if necessary.

At the end of each Sprint (i.e. 30 days) there is a Sprint retrospective meeting that should not last more than 4 hours. In this longer meeting, the changes and challenges the team face are discussed. Changes are factored into the Product backlog and the next Sprint backlog is defined. The process then continues.

Roles

Roles in Scrum can be broadly defined into those who are responsible for delivering the solution (*pigs* or the 'responsible team'), and those who have an *interest* in the project being delivered (*chickens* or the 'interested team'). By this definition, programmers are 'pigs' and users are 'chickens'.

Combining Agility and Governance

Agile methods have no lack of internal structure and there is no fundamental contradiction between combining it with an external governance method. The problem arises where the governance method requires planning certainty that the agile method is not able to deliver. Therefore it is clear that to combine governance with agility requires an agile governance method.

Agile planning incorporates a strategy for dealing with change and uncertainty. A 'command and control' governance structure such as Prince2 views change as an 'issue' to be managed in an exceptional 'issues list'. It indicates a problem that must be quantified to understand the risk it poses. Substantial resources must be deployed to deal with these issues to try to bring the project back on track; to conform to the original plan. Firm planning gives management a sense of security at the beginning of a project. As the project progresses however, changes are seen as jeopardise the plan, leaving the project in a constant state of having to react without the structured mechanisms to do so.

Agile planning views change as normal so no exceptional action is required.

It is natural that organisations crave certainty. Senior management want to be able to rely on estimates of time and cost. They want to be able to quantify their risk; to understand their uncertainty. Therefore there is structural resistance in many organisations to accepting agile engineering because of its lack of guarantees. Yet, in the light of experience of projects that fall behind in their deliverables, this resistance is unfounded because a guarantee that cannot be enforced provides no comfort.

PrinceLite as an agile governance method

Uncertainty in planning arises from uncertainty of business requirements. PrinceLite addresses this fact by layering business requirements into a hierarchy of requirements abstraction; high level (more abstract) to low level (not abstract). The objective of this approach is to insulate requirements change to the lower levels. Thus the high level requirement to ‘Place an order’ is liable to remain stable regardless of the requirements implementation that may change during a software release once the stakeholders have had sight of the result.

PrinceLite puts light emphasis on the production of project artefacts other than those necessary for the management of business requirements. The effect of this approach is to insulate the ‘Product backlog’ (Business requirements specification) from iterative change. Where the requirements are insulated from change at the business level of representation (high), the management of those requirements is simplified. This has the effect of contributing to a planning round that is less susceptible to uncertainty based on the requirements changing, because all that is liable to change is the requirements detail; how the requirement is satisfied.

PrinceLite advocates high level requirements captured in a combination of use case model, activity diagrams and user stories. PrinceLite iteratively refines requirements through project brief, PID and finally expressed in the business requirements specification. This is compatible with Scrum/XP where the project brief and PID may contain the high level requirements prior to the project having secured full funding. When the project begins, the Product backlog can be populated by the requirements captured in the PID.

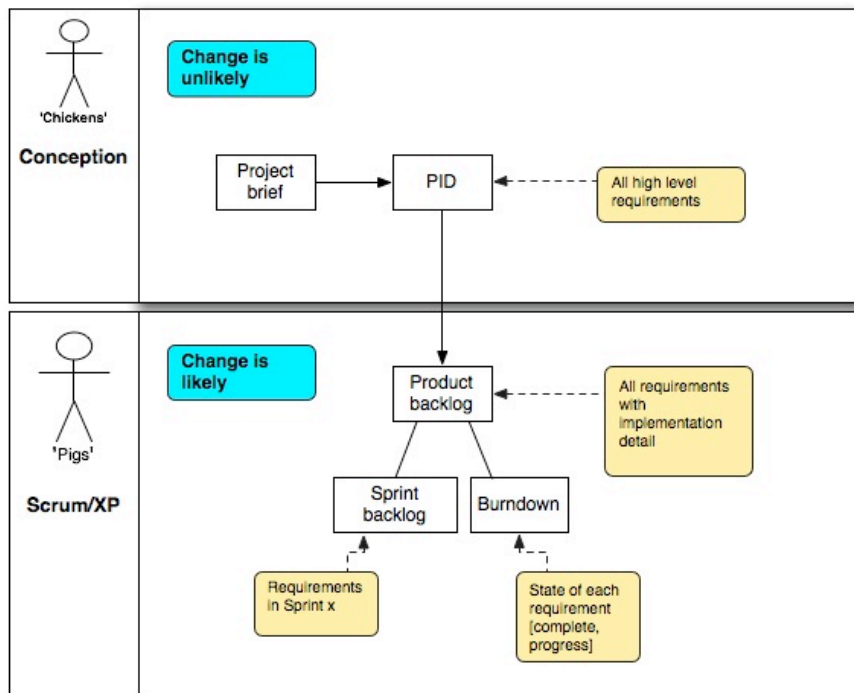


Figure 2: High level requirements are contained in the PID. Implementation detail regarding requirements is articulated in the Product backlog.

Requirements change is normally restricted to implementation detail. At the management level, it is generally possible to produce a robust model of what the system must do.

Figure 2 illustrates that PrinceLite is a good governance fit with agile methods because it insulates requirements change by employing an abstraction of requirements expression. The by-product of this approach is to combine the terminology of Prince2 (Project brief, PID) with the agile engineering method. This is an advantage because medium sized and large organisations have likely made a considerable investment in typical governance methods and are generally more comfortable with a planning approach predicated on reasonably stable requirements.

Conclusion

The uptake of agile methods is hampered by organisational cultures that are uncomfortable with the uncertainty that comes with an overt expression of business requirements that are expected to change. Although in many ways this is a statement of fact, it is not a welcome message. To stimulate the uptake and acceptability of agile methods, they would benefit from being combined with a governance method such as PrinceLite that expresses business requirements in a structure of hierarchical abstraction that insulates them from change at the management level. In addition, PrinceLite can be considered a dialect of Prince2. Therefore, those organisations that are currently committed to a Prince2 governance framework may be more receptive to the approaches of those practitioners who wish to work using an agile engineering approach, because requirements change management is restricted to the engineering domain. The governance domain remains considerable more stable and the planning cycle more traditional.