A Tale of Two Projects

A discussion of iterative, incremental and agile software development

Mark Collins-Cope

Ratio Group Ltd.
17/19 The Broadway
Ealing W5 2NH
London

Email: info@ratio.co.uk
Web: www.ratio.co.uk

We Know the Object

Table of Contents

1. Introduction

Iterative, incremental, and more recently "agile", software development techniques are becoming increasingly popular as the low risk approach to bespoke software development. But many organisations still have still to see the light – they are still developing using the waterfall process. In this paper I illustrate the "agile" approach as a tale of two projects…

2. Summary Points

1. Software development is fundamentally an empirical process. At inception, almost every project faces a myriad of unknowns.
2. As with any empirical process, feedback is vital to keeping things on track.
3. Feedback, in software development terms, requires round-tripping the development lifecycle at regular intervals – producing fully integrated software releases.
4. Feedback comes in many forms: feedback on estimate quality; feedback on analysis and design modelling; feedback on architecture validity; feedback on quality of bespoke infrastructure; feedback on effectiveness of communication; etc.
5. Many requirements only emerge when users get to *see* a system. Often, these are more important than pre-existing requirements.
6. Agile, iterative and incremental development approaches are far more adept at accepting new requirements than the waterfall approach.

The contents of this paper are based on the one-day Ratio Group course - "A Management Introduction to Agile, Iterative and Incremental Development using UML" Contact Ratio on 0208 579 7900 or info@ratio.co.uk.

3. Background

The competitive world of video rental chains is undergoing a shift again. The latest idea from the marketing departments is online rental and delivery – customers will be able to rent videos on the Internet and have them delivered within preset hours. Recent staff moves mean that the two major chains – Star Videos and First Release Videos - are both vying to be first to market. A six month release deadline has been.

Both Star and First Release are in similar positions. They've recruited new staff, unfamiliar with the video store domain. The project teams are similarly structured, containing an analysis team to develop use cases, an application development team to develop application functionality, and technical team to build the required infrastructure. Both companies are hoping to reduce their time to market by buying in pre-built components where possible, and both companies have put together a 'project board' comprising the project manager, their boss and representatives of the marketing department.

4. The Tale of Star Videos

Star Videos have assigned Sam as project manager. Sam is a practitioner of the waterfall development approach: requirements – analysis – design – build – integration – test. He likes the structure this approach imposes on the project.

Sam initiates the project by putting the analysis team onto requirements gathering. He wants to get a good feel of the depth and breadth of the system to be developed. A marketing support team has been formed to work with and monitor progress on the project. Both they and Sam's boss are very keen to see the project come in on time and to budget.

One month into the project and they seem to be making good progress. A large number of business use cases have been collected and documented in detail. The analysis team have produced estimates against these and - all going well – six months looks just about achievable. The marketing support team has given what time it can – and have been employing their creative energies in coming up with ideas for the system. Inevitably they have been busy in other areas as well, so disagreements between them have been largely glossed over; the analysts do have to get something written down after all. The technical team has also been doing well, and has identified some bought-in components they are going to use to assist in developing the necessary project infrastructure.

Sam has been working hard on his project plan, and having discussed the project with his team, has produced a detailed plan for the next six months. The overall plan goes like this: month one – analysis and architecture investigation; month two - infrastructure design; month three - application design and infrastructure build; month four – application build; months five and six – integration and test.

Months two and three fly by. The technical team have specified and produced a first version of the infrastructure. Sam is feeling particularly pleased this passed it's unit tests - in isolation - with relatively few problems. The application team has been developing sequence diagrams and all seems to be progressing nicely. The only fly in the ointment has been the new requirements that the marketing team have come up with. Sam has had the analysis and application teams rework the models and documentation accordingly, but it has been a bit of a rush.

Month four comes, and more new requirements are still coming in from the marketing team. Sam's team are beginning to complain that they're spending too much time reworking the existing analysis and design, but Sam is having none of it – the plan is the plan and come what may, they're going to meet it. Still confident of success, he begins to crack the whip and the team starts working unpaid overtime to ensure meeting the deadline. Grumblings have also been heard regarding the infrastructure. This, apparently, isn't quite what the application team had been expecting and is causing the application modules to take longer than expected to build. Somewhat bleary-eyed, the team finishes month four with some degree of

satisfaction. They're only a couple of weeks behind schedule on the application modules, but at least the infrastructure is in place.

Month five begins with more problems. The infrastructure team is integrating the bought-in components with their own infrastructure code. Unfortunately, there seem to be some fundamental incompatibilities between the bought in components – they just won't work together. A hasty decision is made to ditch one of the components and buy another instead. Two weeks later they're happy that the new component at least works with the others, but at a cost. The new component has a fundamentally different application programming interface (API) – and they are going to be forced to rework their infrastructure API accordingly. This in turn has a knock on effect on the application developers, who are going to have to modify their code to use the changed API. The application team are not at all happy with this; they are already behind schedule, and additional problems have been found in the analysts domain class models – inconsistencies are becoming apparent as the models are translated into code. Long hours mean tempers are getting frayed. Team morale is getting low, and a blame culture is starting to take hold.

To top this, the marketing department have come up with a new area of functionality that simply "has to be included" – a loyalty bonus scheme. Realising this is a no hoper, Sam tells the marketing department this functionality must be deferred until after the first release. They are not pleased.

Month six is not a happy time for Sam or the team. The team have taken up almost permanent residence in the office, but the quality of their work is in a steady decline. Sam is forced to report a delay to the project board, who had been eagerly anticipating the first demonstrations of the system. Worse still, when towards the end of the month they see the small parts of the system that are working, it becomes immediately apparent that they're not getting the system they thought they were going to get. Sam points to the use case documentation, and sure enough they have got what they signed up for. This doesn't make them any happier however, and an emergency board meeting is called, resulting in Sam's boss calling in external consultants to audit the project. The report is not good – and the project is given one month to get back on schedule.

Month seven is a complete panic. Some new functionality is delivered, but the marketing team is still complaining it isn't what they wanted. Their confidence is dwindling fast, and at the end of month seven they decide taken to cancel the project.

Month eight begins with Sam receiving his P45 in the post…

5. The Tale of First Release Videos

Danny works has worked as a project manager for First Release Videos for a few years now. Having had some initial success using a form of waterfall process, he progressed some time back to using risk based iterative and incremental development techniques. He's recently been adopting the best practices of the emerging field of agile software development, in particular its emphasis on human centric factors and its ability to deal with changing business requirements.

When approached to take on the project, he releases he is facing a substantial challenge. His initial reaction is to say the project is infeasible given the timescales. Danny's boss pushes back strongly, and Danny finally agrees to take on the project with provisos: the project team must be located in a single room close to the marketing department; and a single member of the marketing team must be designated "project sponsor" with decision making authority.

Danny initiates the project with a brainstorming workshop attended by all parties. At the end of two days they have identified - but not elaborated - a long list of use cases that may need system support. They've talked around these quite extensively, and Danny feels confident they have some idea of the services required of the system.

Danny and his team spend the next few days discussing the implications of the use cases in more detail - coming up with a first cut set of estimates. A "sunny day " analysis indicates six months may be enough time. Given the risks and unknowns, however, Danny doubts the sun will be shining all the time!

Accordingly, Danny goes back to the sponsor, and explains they are going to deliver the system in functional increments, with the sponsor being responsible for deciding the contents of each release. The sponsor accepts this when Danny explains it will enable her to add new high priority requirements at regular intervals. They assign use cases to the first six-week increment based on business priority and estimated timescales, and identify the use cases likely to be included in the subsequent release, one month after this.

Danny then produces a detailed plan of work for the next six weeks, which the team estimates against. The technical team are to spend two weeks validating the architecture – making sure the bought-in components work together and that they function correctly under the appropriate loading. The analysis team is to spend two weeks detailing the use cases for the next increment, and then move on to the most stable use cases of the next increment. In parallel with this, the application team are to spend two weeks developing sequence diagrams based on the early use cases, with a particular emphasis on specifying the API they want the infrastructure to provide. They discuss their findings on a daily basis across the desks with their technical team colleagues.

Month two starts with headaches for Danny. The components have been tested and found not to work with each other. Another two weeks of component selection and validation must be undertaken. This time, however, the technical team *knows* what questions to ask the component suppliers, and are able to home in on the right component. As they test the new component, they also investigate how to implement the API specified by the application team. This results in some further to-ing and fro-ing, but eventually the two teams are able to agree on a specification.

As a stop-gap, the technical team produce a stub-library version of the infrastructure to enable the application developers to test their ongoing development work. This in turn leads them to identify problems in the analysis team's modelling – which they feed back accordingly. The technical team spends the rest of the month developing the most important parts of the infrastructure. At the end of month two about 70% of the expected increment one functionality has been delivered in fully integrated form. The analysis team has now reworked their domain model of the system, and this is displayed prominently in the project area, with many other related project diagrams.

Month three begins with a demonstration of the increment one functionality to the sponsor and her support team. Some bugs are noted, but more importantly it becomes apparent that there is some functionality missing that is vital to the high priority use cases already implemented. This is immediately scheduled for the next release, and according some previously scheduled functionality is deferred until later. It also becomes apparent that the team aren't always delivering what the sponsor expected. Amendments are scheduled for the next increment. Whilst not overjoyed, the sponsor is at least pleased to see something of a working system, and agrees to continue the project, albeit aware that the scope of the final system is likely to be less than originally thought.

Danny and the sponsor then produce a detailed plan for month three (increment two). The technical team is to amend and finish the infrastructure based on an updated version of the API specification – which the application team has amended based on their experience to date. The identified bugs are to be fixed. The highest priority use cases are assigned to the application team, and the analysis team start work on the use cases for increment three, whilst assisting the application development team on as-needs basis. The application team work some overtime in an attempt to catch up on functionality missed in increment one.

Month four follows a similar pattern. A review of the system identifies some bugs and some more functionality to be implemented. Project velocity is now about 110% of the adjusted estimates; this is fed back into the estimating cycle. The increment three plan includes working on the remaining bugs in the infrastructure code, more enhancements to the infrastructure API, and the appropriate use cases. The end of month four sees the project delivering broadly to the revised schedule.

The month five review shows things are getting better. There are less bugs – more and more reliable infrastructure has helped here - and less problems of communication – previously identified shortcomings mean that everyone is learning to communicate more effectively.

At this point, however, a major change in requirements - a loyalty bonus discount scheme - crops up. The sponsor decides it is vital that this be included in the first live release of the product. A period of haggling ensues, leading to some other functionality being ditched. A reasonable chunk of time is also put aside to enable a large-ish refactoring of the existing system to be undertaken (the new functionality has some fundamental impact on the application team's existing code). This takes longer than expected, but is supported greatly by the automated test suite the team has been developing throughout the project. Month five ends a little behind the revised schedule – velocity is around 90% of that expected for this increment.

The functionality delivered for the month six review includes the loyalty discount functionality. This pleases the sponsor, who is feeling increasingly "in control", and mitigates the disappointment about the other delays. Danny and the sponsor schedule the remaining work and decide to include only functionality that is absolutely necessary for the first production release. The remained of the time is assigned to additional testing and bug fixing.

Month six ends with the system being delivered. As it turns out, about 60% of the originally defined functionality is included. Of the remaining 40%, 20% is accounted for by "new" functionality, and 20% is just not there. The sponsor department does have a system to go to market with, however, and has the confidence to continue the project further.

Month seven begins with Danny getting a pay rise!

6. References/bibliography:

1. Cockburn, A, *Agile Software Development,* Addison-Wesley, 2002.
2. Schwaber, K, Beedle, M. *Agile Software Development with Scrum.* Addison Wesley, 2002.
3. Ambler, S. *Agile Modelling,* Wiley, 2001.
4. Beck, K. *Extreme Programming Explained,* Addison Wesley, 2000.
5. Ambler, S., Rosenberg, D., Collins-Cope, M., Stephens, M., *Agile Development with the Iconix Process,* Apress, Due for release Summer 2003.
6. Other related articles: see www.ratio.co.uk/techlibrary.html.

7. Author

Mark Collins-Cope has been working in the Software Engineering for over 15 years. He has undertaken a variety of roles within the software industry, including: project management; analysis of requirements; technical architecture, software design and development, etc. Mark is now Technical Director of Ratio Group Ltd, a UK based training, consultancy and development company specialising in object and component technologies (see www.ratio.co.uk for further information), where he combines his management role with ongoing involvement in complex technical projects. Mark speaks regularly at a variety of international events. Mark can be contacted on markcc@ratio.co.uk.