White Paper: IT projects & Agile

Do we all hate change that much?!

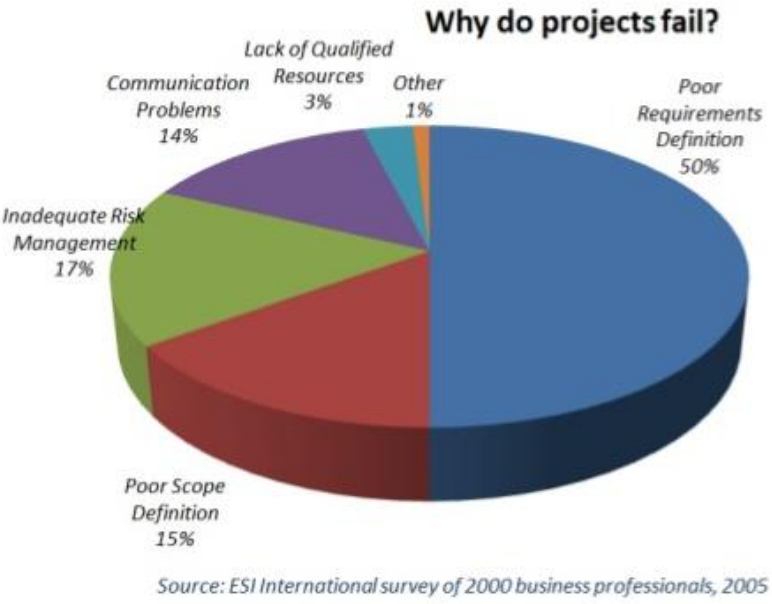
As we discussed in our Blog, far too many projects end up a disaster. Here at Ethical IT we hear horror stories of badly run projects that cause outages, stress and even data loss – in fact we are often brought in to pick up the pieces, set things back on track and pull it all together after a project has hit trouble or failed completely.

In this short paper we cover why projects seem to fail so often, and provide some practical advice - based on our own collective experiences – on how to use bits of the Agile methodology to help you learn from lessons past and hopefully manage change as best you can in an ever evolving world

Some of the stats on project success are car crash reading! Why?

Requirements

As you can see in this graphic, 50% of projects fail due to poor requirement gathering. Sometimes this may be down to neglect on the part of the project team, but more likely it's because they simply change as the project is in progress. Life goes on, things change, people come and go, trends emerge then vanish (remember Friends Reunited?!) – and this is a nightmare if you set out a lovely spreadsheet of prioritised requirements which everyone understood and agreed which then completely changes on day 2



Heraclitus observed way back in 500 BC that "*change is the only constant*" in life, therefore how your project is set up to handle shifts in requirements, usage or access will go a long way to determining its success and relevance once complete

Old school project managers may file this under "scope creep" - to be avoided at all costs - but technology moves at such a pace nowadays that any project longer than a few months is likely to encounter unforeseen external factors like a new killer app that changes the landscape (disruptors) and offers a new way to do a task that your stakeholders prefer over your project. Game over? It needn't be.

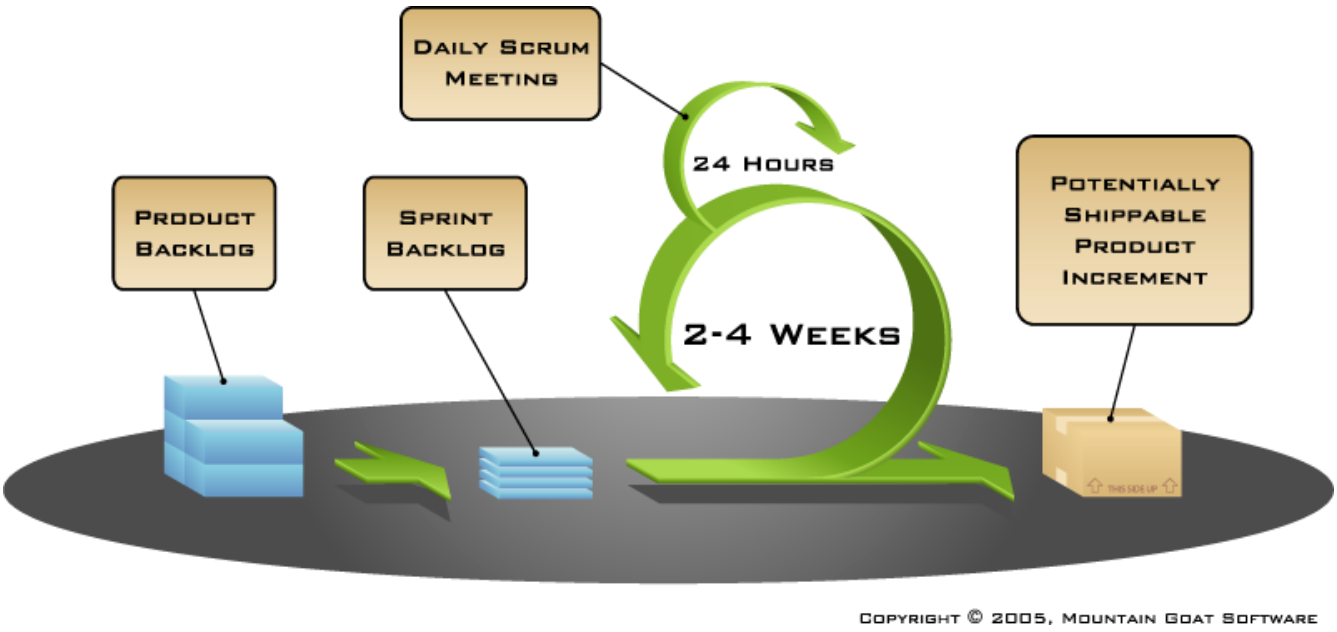
Agility

"Agile Project Management" has been around for about 10 years now, and you can spend a lot of money on expensive consultant experts, but our big take away from the Agile framework is that we make allowances for scope to flex and adapt as the project goes along.

It seems too good to be true but it's actually quite a simple concept. Giving the end users something to play with – however rough – early doors is a huge win; a big learning exercise that your project will take away priceless feedback from at a very early stage and update the requirements when the sums of money and time spent aren't too scary.

Iteration

The life and soul of Agile Project Management is iterations. The idea is that you do a short, focused burst of work and then stop to show the stakeholders the results. Typically this means your project team are flat out for a 2 week period known as a "sprint", then there is a demo session with the end users to allow them to "oooh and ahh" over what has been done - click on things, hold them, break them and think: "will this be useful?"



This diagram sums up the key processes: 2 – 4 weeks is the "sprint" and each day the Project Manager (in Agile he or she is known as a "Scrum Master") holds a daily meeting with the team to review progress made and what's happening that day.

At the end of the sprint you have something that is demonstrable to the business, which they can review and feed back on, which in turn shapes how the next sprint will focus energy - adapting all the time to ensure what you are delivering is still relevant. Sprint after sprint happens until you arrive at an end product that is useable

How to prototype

Not all projects can be done in an agile way – building a bridge cannot really be done in 2 weeks then a demo given to passengers wishing to cross it. But in the IT world, most projects can be shown to the end users in a raw format – a new desktop model or a beta version of a program for example.

Perhaps the early stages of a new CRM system – even if the menus do nothing and there is no data in the database, simply having *something* to click on and look at will result in reams of feedback that will almost certainly alter the course of the project and – most importantly of all – keep your stakeholders engaged and enthusiastic about what is coming their way and how it will help them once the project is finished.

The minimum viable product

Another benefit of going through iterations like this is the possibility of delivering benefits earlier. We have often found that the prototypes end up being largely accepted by the users as good enough to achieve the result of the project, early, and so any time remaining can be recouped or spent on minor refinements and polish - while the users can get straight on with actually using what you have delivered, ahead of time.

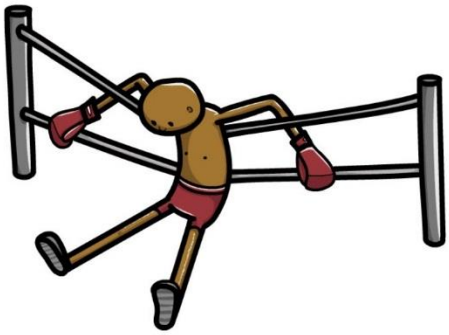
Achieving this is only possible by strictly prioritising your requirements and constantly reviewing this.

During a sprint, there will be “Must Have” requirements e.g. it must turn on, then “Should Have” such as it should boot up inside 2 minutes, then “Could Have” such as the desktop wallpaper being your company logo.

	Requirement	*MoSCoW*
A	Users can log onto the web site.	Must
B	Users should be able to avail of a "Forgotten Password" utility .	Should
C	Users can change account details.	Must
D	A user can send an email to the system requesting a change to the account page.	Could
E	When a user clicks on a phone number on the web page a call is made automatically from their desk phone to that number.	Won't

This approach, known as MoSCoW, is done before each sprint and means the team will always deliver the Must Have features and, where time permits, they will get the Should' s and Could' s done too. The result is a usable product as it meets the minimum spec; nobody has to wait an extra few weeks whilst the design team faff around with the logo on the desktop before the whole thing gets released.

Risks and Threats

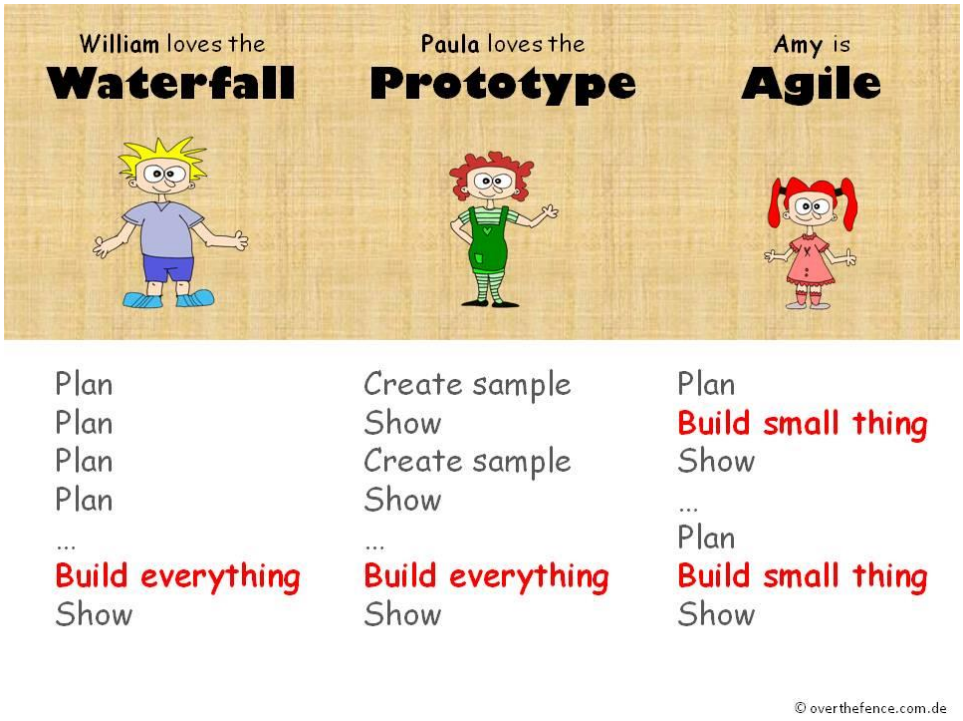


Agile projects come with their own pitfalls of course; no matter what your approach, methods and tools you will not succeed unless everyone on the team is completely clear on the objectives, and your stakeholders or end recipients are fully engaged and playing an active role in the team.

Equally, good governance is vital to ensure you are protected when thing go wrong and the project sponsor starts asking questions.

1. **Documentation** – this is often cited as a weakness in Agile projects. The product being made is allowed to evolve, so keeping up with accurate documentation can be hard. It needn't be; the acceptance criteria at the end of every sprint should always include updated & signed off documentation based on the current iteration
2. **Scope Creep** – we identified this as a rather old school issue earlier, however the prioritisation of requirements must be rigorous and not allowed to slip, especially during a sprint where they may come under the most pressure ("do they *really* need this feature or is it more of a nice to have? We are short on time so let's skip it")
3. **Prioritisation** – the process of ranking the requirements by importance must be very carefully facilitated to ensure fairness and balance. People look at a product from their perspective only, and often what the CEO considers a "Must Have" requirement gets recorded as such because she is a VIP when it may be not what the people who will use it day in day out feel. Equally you may end up with all the requirements being "Must Have" – in this case you have not got a viable Agile project and it should be broken down into smaller projects
4. **Legislation** – such as the new General Data Protection Regulation (GDPR) which applies to data you hold, the measures you or your cloud provider take to protect it, and how you remove personal data when asked to, may be overlooked as your product evolves into something brilliant that was not even considered when you started out, but which may not comply with the law. Taking a step back and looking outside the "scrum" is vital.

So try to start small, fail fast, learn quickly and – using controlled iterations – evolve, shape and mould your project harnessing the creative power of your engineers and the imagination of your end users to deliver something that is relevant; think about how you can move your project from the right to the left of this:



Further Reading

10 Key Principles of Agile: http://www.allaboutagile.com/what-is-agile-10-key-principles/

Rescue and Recovery of failing Projects: https://www.projectsmart.co.uk/rescue-and-recovery.php

Would you like to discuss any of this?

At Ethical IT we are here to help Ask us about this subject or indeed any other current trends like Office 365, Charity Donation Schemes, Remote Working, Cloud Security and more – we're all ears. Our Twitter feed also contains regular tips and tricks about these sorts of things, so please follow us @ethicalituk.