In the Press

De-risking technology projects

As featured in IT Pro Portal & Information Age

Fewer than 1 in 3 software projects produce successful outcomes, according to the Standish Group's *Chaos report for 2015*[1]

66% end in partial or total failure. This paltry rate of success, based on analysis of 50,000 projects worldwide, is broadly unchanged over the last five years. Whatever the reasons for failure, it seems that project teams aren't learning from their mistakes.

Of course, success is a relative term. It can be defined and measured in any number of ways, and often depends on context – and on what the story needs to be. The Standish Group's definition requires projects to be on time, on budget and to produce a satisfactory result – an examination of value, user and sponsor satisfaction, and target requirements. Whatever your take on this, anyone involved in technical projects knows that far too many fail to deliver the benefits envisioned at their outset. Extrapolating from those experiences, it's likely that billions of pounds and millions of hours are wasted annually on delivering changes that either don't add value or end up being cancelled altogether. Clearly, there are huge gains to be had if we can just avoid some of the factors that contribute frequently to project failure.

Some pre-requisites for a successful project are obvious and well-established: getting the requirements right; providing effective leadership; and having the full support and engagement of sponsors and users. Without these in place, no project is likely to succeed. This article considers some of the less apparent means of reducing the risks to your technology projects.

Scope and Timetable

This is a matter of methodology and development mindset. A purely waterfall or purely agile approach is seldom best; the most effective method is usually somewhere between these extremes. Understanding requirements and business benefits is essential, but spending months – or longer – producing reams of documentation is not the answer: apart from being difficult to digest, it is often out of date by the time it's finished. A word of warning though: don't allow project teams to cherry pick the easiest elements from each methodology. It can become an excuse for not documenting anything!

A set of the fundamental requirements, with sufficient detail to develop against, is the ideal starting point. The rest can be delivered iteratively. This ensures that you don't lose sight of business benefits, while realising the main benefit of an iterative approach: engaging stakeholders and acting on their feedback. Iterative doesn't have to mean agile: it's perfectly possible to have well-defined key requirements for each phase and to proceed iteratively, although a process for prioritising requirements becomes imperative.

A benefit of this methodology is that the project scale becomes more manageable, and the timescales become more immediate, allowing for better focus. If the first deliverables on any single element of your project are more than a few months away, you need to question your approach. You may be tackling the problem in the wrong way; you may be using inappropriate technology; you may even be doing the wrong thing altogether – not everything has a solution that is rooted in technology. Clearly, the less time spent doing it wrong, the better, so aim to deliver something as soon as possible. Delivering earlier doesn't just mean that users can begin to evaluate and feed back sooner. It also provides a useable tool for the business: the sooner it is live, the sooner the benefit; and a fraction of the final benefit is better than none at all.

How and What to Deliver?

Given the choice, many organisations prefer to develop in-house. This is usually because they believe that internal projects will produce a solution tailored to their own needs rather than one compromised by others' requirements; or that they will enjoy greater control; or that it will be less costly.

These notions don't always stand up to scrutiny. Recruitment and training for new work takes time and costs money, and there is always an opportunity cost involved. Staff turnover frequently results in loss of expertise and thus project control – how many 'in house' technology projects are managed and staffed by contractors? It can be easier to manage suppliers, who are bound by commercial contracts, than in-house teams; and third-party suppliers bring experience, which can save time and money as well as increasing the likelihood of the end product being well designed and future-proofed.

In the Press

If the decision is taken to go outside the organisation, are the requirements to be met with an off-the-shelf product, a bespoke solution, or a platform? All products are customisable to some degree, but it's rare that requirements are so similar across different organisations that one size is a perfect fit for all.

Furthermore, the future direction of your solution is at the mercy of a third party's product roadmap. On the other hand, it's rarely necessary to start from scratch, either: almost any new requirement can use common components, and if the work has already been done, it makes no sense to reinvent it. Therefore, a platform-based solution, with common foundations and a custom business logic layer, often makes the most sense. Some of the time and cost of development and testing is eliminated, along with much of the risk inherent in new development. If coding is necessary, the buyer should ensure they understand which elements are configurable and which require code-based changes. This is not to say that coding is problematic but it inevitably extends timeframes and increases project risk.

Designing and Implementing the Solution

When requirements are determined, the capabilities of the technology must not be the starting point. The point of the technology is to support the best way of running your business; it is not to dictate how it should operate. If this is occurring, the first thought should be to change the technology, not to adopt sub-optimal requirements and reduced expectations.

Adequate testing is a non-negotiable element of any technology project, yet an alarming number of software vendors have no formal testing function. Some features do lend themselves to automated testing, but most require a dedicated test team. Test activities should mirror those of the development team, with testing performed over the course of the project and with a longer tail. If they're left to the end, as in traditional methodologies, delays in production often lead to cuts in testing time, increasing the risk of a flawed end product. There is also less opportunity for design flaws or missing requirements to be flagged early; User Acceptability Testing (UAT) on its own is not a suitable testing methodology.

Prioritise Simplicity and Performance

The success of a technology project depends on more than just its technical underpinnings. Developers often think of the externals as just 'lipstick', but the user experience is absolutely critical to success. This doesn't just mean generating wireframes and design guidelines, but also considering storage, network requirements and overall performance before starting out. The attitude must be that if users ever have to wait more than a second or two to load information, there needs to be a good reason for the wait, and consideration given to what it means in terms of their experience.

Ultimately, a journey through the product should be smooth and intuitive; and tools and alternative routes must be logically placed without being intrusive. The process itself might be complicated, but completing it should be as simple as possible. Indeed, this is usually the rationale behind the project in the first place: the simplification and improved efficiency of a process is what adds value. Remember that developers are experts in software development. They are not user experience experts and should not be responsible for such output.

In summary, successful projects will:

1. Focus on delivering early rather than scoping out comprehensively.
2. Select a platform solution with reusable components and the flexibility to apply custom business logic.
3. Ensure requirements determine the technology, not the other way around.
4. Incorporate continuous testing within an on-going development programme.
5. Make the user experience as intuitive and enjoyable as possible.

ENDS