AVOIDING PROJECT FAILURE: PUTTING PEOPLE FIRST

Luke Barrett and Marc McNeill PHD.

ThoughtWorks Ltd.
9th Floor Berkshire House
168-173 High Holborn
London, WC1V 7AA

The consistent failure of software development projects to deliver what is expected of them is a significant headache for businesses. It has been claimed that only around a third of such projects can be regarded as successful and that one of the primary reasons for this lack of success is poor requirements management. Requirements are what drive any development process, yet without a shared understanding between those who produce the requirements (the business), those who turn them into tangible software (the developers) and those who ultimately use the software (the end-users), project failure becomes ever more likely. Using a case study this paper discusses the use of participatory techniques and lightweight models to describe and explore the problem and its potential solutions; applying highly iterative, feedback-driven and people-centred techniques from the outset to provide clarity in what is really required and so position teams for success.

Introduction

Software projects have a poor record of delivery. Whilst methodological issues have been raised (Glass 2006), the Standish Group's periodic CHAOS Report (Standish Group International 2004) suggests that only around a third of such projects can actually claim success. Defects in requirements are a major source of the defects that are later identified during testing, and problems with requirements are among the top causes of project failure (Schwaber 2006), indeed it has been estimated that 71% of projects fail due to poor requirements management (Lindquist 2005). The Standish Group point to lack of user input together with incomplete / changing requirements as the leading causes of "challenged" projects. On the flip side they identify user involvement, executive sponsorship and clear requirements as the most important in ensuring success.

This paper discusses the role of effective participation of the right stakeholders in helping to drive project success. Focusing that participation around lightweight models that are tangible and visible, and doing 'just enough' analysis, helps ensure buy-in from the outset and increases the likelihood of successful project delivery.

Participation

Ultimately the vast majority of software solutions are about helping people (end-users) to achieve their desired goals more effectively. The challenge for the software development

community is how to go about consistently delivering solutions that do just that. Addressing this challenge begins with communication between three important groups:

1. The people paying for the solution (in commercial software development often referred to as ‘the business’)
2. The people who are going to use the solution (the end-users)
3. The people who are going to implement the solution (the do-ers: project managers, business analysts, quality analysts and software developers)

While the importance of including end-users in the development process seems obvious (after all they are the people whose improved productivity, reduced error rates and increased morale established the business case for the solution in the first place) all too often they are missing from the equation. Despite the work of Norman and Nielsen (and many others) over the last decade to raise the profile of end user involvement; despite Standish's work pointing clearly to the importance of end-user involvement; and despite the rise of the Web (and particularly Web 2.0 with its emphasis on rich user interactions not previously associated with web applications), too few software development projects take a user-centred design approach.

The increasing popularity of 'Agile' software development approaches (e.g. Martin 2002) provide an opportunity to address this. Agile approaches are inherently people-centric being highly iterative and feedback-driven with that cycle of feedback being remarkably short in comparison to traditional methods. Whereas a traditional 'waterfall' approach to development might see a nine to twelve month gap between the business specifying a requirement and seeing that implemented, in an agile project that might be as little as a week. Such a rapid turn around of a requirement into functioning software allows teams to 'fail fast' - that is obtain feedback from end-users at the point of need, not many months after an implementation decision has been irrevocably (or at least expensively) embedded in a design.

While the omission of input from end-users is one of the more obvious issues to address, more challenging is ensuring the correct balance of representation from the business. The goal is to achieve frequent input from those who have the best grasp of the business need - this may mean input from multiple departments and certainly means input from various levels of seniority. In addition there will be members of cross-cutting functions who will also have necessary contributions to make (e.g. legal, security, operations, etc.) although the frequency of their required input is likely to be less.

The point about seeking input from multiple levels of seniority for the key owning business area (or areas) is worth restating. It is likely that on a day-to-day basis the business area representative on the team will be someone of moderate seniority, that is they are senior enough to have a good grasp of the business (and to be empowered to make decisions concerning the solution up to a point) but junior enough that this is their day job (or a good chunk of it). Again one leading cause of project failure is lack of executive sponsorship so using the highly tangible output of the process (see below) at regular showcases (weekly / twice weekly) to engage with, and extract feedback from, senior business representatives is essential.

Lightweight models

With a multi-disciplinary team in place the challenge becomes to find a quick and clear way for the team to generate and evolve potential solutions to the problem at hand. The solutions must be captured in such a way that all three key constituencies can grasp them and provide rapid feedback. It is here that simple lightweight models of the problem domain and possible solutions to it come into their own, taking the following form:

1. **Strategic**: A simple prioritised list might be used to call out and agree key project objectives with their associated metrics while a financial model might be used to quantify the benefits case. It remains remarkable how many projects proceed with no clear quantification of the benefits they may return.

2. **Process**: The high value, high frequency processes that the solution needs to support are modelled. Typically these models explore how key user types achieve their highest value goals. Again the models are lightweight ideally at a level of abstraction that requires no more than seven to nine high-level steps. The aim here is to provide a framework for the more detailed decomposition of the solution which will occur at the next level down - that of implementation.

3. **Implementation**: The team is now distilling the output of the higher level models and capturing them as business requirements of sufficient detail that the development team can estimate the delivery effort. Importantly, at this point, the team's understanding of the written requirements should also be validated by the creation of low fidelity prototypes of key usage scenarios. On the development side the need to provide estimates should also prompt, as required, the creation of a proposed architectural model and options for key technology choices.

The process by which these artefacts are evolved is of at least equal importance to success. Experience suggests that, for a project of sensible scope, two to four weeks of effort evolving the models, distilling the requirements, prioritising them (from a business perspective) and estimating them (from an implementation perspective) will prepare a team for release planning and then development.

The initial portion of the process (the first week or so) tends to be heavily workshop focused. The team work together in short timeboxed sessions to collaboratively evolve the models and requirements. Where possible updates occur 'live' in the workshop, otherwise feedback is incorporated in dedicated consolidation time between sessions. As an example, using marker pens and index cards, it is easy to model and refine a high-level process 'live': cards can be re-ordered, torn up if necessary, replacements provided, new ones added. Likewise high-level requirements captured on index cards have the same flexibility.

As the process continues the intensity of workshops reduces (although a basic heartbeat of workshops remains in place to ensure focus) allowing more time for basic consolidation of the artefacts, 'offline' detailed analysis, workplace observation and usability testing. Key to this processes ability to address the two themes of engagement and clarity of requirements, is the way in which the multi-disciplinary team is intimately involved in the evolution of the artefacts that represent their shared understanding. The

team see these artefacts taking shape in front of them based on their continuously requested feedback. And that feedback is in response, not just to written requirements, but to models (particularly the low-fidelity prototypes) that give them a really tangible feel for what's being discussed. Combine this with regular (e.g. weekly) presentations to senior stakeholders and the approach provides a powerful mechanism for engaging stakeholders across multiple areas and levels of seniority as well as for driving clarity around any proposed solution.

Process in practice

A case study for this approach follows an investment bank developing a new Client Relationship Management (CRM) tool. Four weeks were set aside for requirements definition with a focused core team and subject matter experts identified when appropriate. A dedicated project room was allocated. The project commenced with a kick-off meeting where all stakeholders were invited. The executive sponsor introduced the project, stated its importance and the need for people to make time in their diaries if required. Borrowing from Hohmann (2006), innovation games were played to help the team create a vision of the future, and what the risks to the project were. From this the project objectives were distilled and an initial risk log drawn up. A second workshop, (again with the whole team) identified high value high frequency user types and created personas (Cooper and Reimann, 2007) for each of them. Each workshop lasted ninety minutes, giving the team time to consolidate in the remainder.

The core team then walked through the 'as-is' business process, using index cards to illustrate each process step. These were stuck on the wall. On a second wall the 'to-be' process was mapped out. Using cards allowed elements to be moved around or removed and torn up if not necessary. Having them on the walls ensured visibility, and interest from stakeholders. They could see progress being made and found it easy to comment and make suggestions.

As the 'to-be' process stabilised (i.e. became less volatile following a number feedback loops), a lo-fi prototype began to take shape. Again, this was modelled on paper and key screens were again put on the walls. A technical architect was present during all workshops and was able to begin to propose a technical solution to support the process. By the end of the first week the team were able to present ('showcase') a proposed 'to-be' process supported by pen and ink drawings of how the application may look / behave. The stakeholders provided feedback and some radical changes were made – with the low fidelity of the artefacts nobody felt precious to the work done and changes were easily accommodated.

As the volatility of the design reduced, the lo-fi prototype was committed to PowerPoint and a business analyst started to document requirements as stories in the format "as a [user], I want to[requirement], so that[value]". Guerrilla usability testing (Nielsen, 1994) of the prototype was undertaken to refine the interaction design. Technical "spikes", rapid and time-boxed technical investigations of potential functionality, were performed (such as a Google map – Microsoft Outlook calendar 'mash-up') to demonstrate feasibility and help with the estimation process.

By the third week the team had an extensive list of validated requirements that had high level implementation estimates. These were printed on cards with values on them (these values were arbitrary numbers for the exercise, reflecting an indicative magnitude of effort but not expressed in real days or cost). The business were then invited to prioritise the cards by "buying" features. This was done in four rounds simulating four releases, with each release comprising of end to end functionality (rather than cherry picking high value features without the supporting low-value functionality). This process was iterative, indeed the planning continued through the development lifecycle.

At the end of the four weeks the stakeholders had a shared and common vision, prioritised and estimated requirements, a release roadmap and a lo-fi prototype that articulated how the application should look and behave. Four months later release one development was complete with the application rolled out to the business two months after that. During the development process changes and improvements were made based upon the real software rather than the lo-fi prototype. Release one delivered functionality that delivered immediate value to the business, with further features being delivered in subsequent releases. As users had been involved in the design and usability had been built into the process from the start, the tool was sufficiently intuitive to require no training.

Conclusions

The seeds of success or failure are often sown early in the life of a project. By applying highly iterative, feedback-driven and people-centred ways of working right from the get-go businesses have a better chance of addressing the lack of engagement with key stakeholders (senior management, end-users, etc.) and poorly defined requirements that, time and again, lead to project failure.

References

Cooper, A. & Reimann, R., 2007, *About Face 3: the essentials of interaction design*. Wiley Publishing.

Glass, R. L., 2006 *The Standish report: does it really describe a software crisis?* **49**, 8, 15-16

Hohmann, L, 2006 *Innovation Games.* (Addison Wesley)

Lindquist, C, 2005 *Fixing the Software Requirements Mess*. CIO Magazine, November issue

Martin, R. C., 2002 *Agile Software Development, Principles, Patterns, and Practices*. Prentice Hall.

Nielsen J, 1994 *Guerrilla HCI: Using Discount Usability Engineering to Penetrate the Intimidation Barrier*. http://www.useit.com/papers/guerrilla_hci.html

Schwaber, C, 2006 *The Root Of The Problem: Poor Requirements.* Forrester Research, Inc.

Standish Group International, *Chaos report: Chaos Chronicles.* http://www.standishgroup.com/