Full paper in Proceedings of the 16th Annual NACCQ, Palmerston North, New Zealand, July, 2003 (eds) Mann, S. and Williamson, A. www.naccq.ac.nz

Putting the P in Programming: The importance of people and process

Carol Kelly
Professional Development Ltd
Palmerston North, NZ
carol@prodev.co.nz

ABSTRACT

The causes of software failure have been well documented over the years, yet still the failures continue. This paper looks at causes of software failure, and compares IT projects to other disciplines. It is written as an opinion piece from an industry professional and uses four illustrative examples of computing practice where IT projects did not realise their potential, to highlight possible reasons why computer systems underachieve.

The main reason established for the problems was that the development activity occurred in relative isolation from the final users, highlighting the fact that the programming experts felt the need to 'go it alone', as they were the specialists in the system.

It then suggests a framework plan, which may reduce the risks of implementing a new system; covering the areas of problem, previous work, procedures, policies, perspectives, people, politics and process. It also suggests ways in which the IT industry can learn from other disciplines to achieve better results.

Keywords:

process, project management, software failure

1. INTRODUCTION

Software engineering is no longer a new industry. While not as old as some other professions, it has reached an age where we should expect systems to be built with little fuss. So why do we still hear stories of applications gone wrong and users not satisfied?

Gathering reliable statistics on software failures is not an exact science. Most of the information comes out of the US, and it is not clear how repeatable the statistics are in other countries. The news is disturbing through. In 1994 it was estimated that people working on projects that were subsequently cancelled made up as much as 15 percent of total U.S. software projects, totaling up to $14 billion per year. Cancelled projects are usually 1 year behind and 200% over budget (Jones, 1994).

McConnell noted the following software failures:

- *"Problems with the baggage handling system caused a delay of more than a year in opening Denver International Airport. Estimates of the delay's cost ranged as high as $1.1 million per day.*
- *The FAA's Advanced Automation System overran its planned budget by about $3 billion.*
- *The IRS bumbled an $8 billion software modernization program that cost U.S. taxpayers $50 billion per year in lost revenue.*
- *After topping $44 million, a long series of overruns forced California to cancel its motor vehicle registration system.*
- *The B-2 bomber wouldn't fly on its maiden flight because of a software problem" (McConnell, 1999).*

In 1996 a survey was undertaken of 360 companies in the US which showed that 73% of corporate software projects were canceled, over budget or late (King, 1997).

This paper is an opinion piece from an industry professional, using four short illustrative first-hand examples of computing practice to highlight possible reasons why computer systems underachieve, and suggests ways in which the IT industry can learn from other disciplines to attain better results. While study of the human factor in software development may not be considered novel or new, the fact that software development projects continue to fail because of the human factor means that continued attention needs to be given to this area.

2. CAUSES OF FAILURE

A 1995 KPMG survey listed the following as the top 6 reasons why software projects fail.

- ♦ Project Objectives Not Fully Specified (51%)
- ♦ Bad Planning and Estimating (48%)
- ♦ Technology New to the Organization (45%)
- ♦ Inadequate/No Project Management Methodology (42%)
- ♦ Insufficient Senior Staff on the Team (42%)
- ♦ Poor Performance by Suppliers of Hardware/ Software (42%)

(KPMG, 1995)

One possible contributing factor to all of the above could be that our industry has traditionally tried to "go it alone", and undertake the whole project itself. "We turn to our IT experts (programmers/analysts) and expect them to also be experts in project management, process and communication…What many of those analysts lack are the communications and people skills needed for key tasks such as eliciting exact systems requirements from users...Soft skills are talked about the least, but they're probably the most important'' (Durbin, cited in King 1997).

There is a wealth of experience in other industries (such as architecture and engineering) that could provide advice from hundreds of years of making mistakes. Much of what the IT industry is 'discovering' in research, particularly as it relates to interacting with customers and process, has already been concluded in other research areas, most notably social sciences. Yet we seem to think we were the first ones to encounter these problems. For example, it seems to be a revelation when it was concluded that a customer changes their mind frequently when they are unclear as to what they wanted in the first place. There is even a name for it – 'scope creep'. This has been known in the building industry since the first structures were created. For example in a report on Florida's roading project overruns, it was concluded "In the projects we examined, cost overruns were predominantly related to problems that occurred during the planning and design process prior to construction, including: (1) errors and omissions in design plans, (2) inadequate coordination with local governments and utility companies, (3) problems in identifying the scope of work to be done during project development, and (4) changes in project specifications after designs had been completed" (Turcotte, 1997).

Architects have been leading building projects for years. The design statement on the website of Fennie and Mehl, Architects, San Francisco reads:

"All really good design, whether the project is a piece of furniture or a towering skyscraper, is based on one premise ... meeting the expectation and need of the user, both on a functional or physical level and on a spiritual or psychological level…The design or building must meet the requirements of our physical world as well. It must be of sound form and be constructable; it must function efficiently for its intended use, and finally it must be a pleasurable and memorable place... a place that invites the user to return.

The Architect's task then is to blend all of these seemingly varied forces into a cohesive whole, and effectively communicate the design in such a way as to guide the client through the design process, and subsequently, the builder during the construction of the building. The successful completion of the original design intent is dependent on how well the Architect can shepherd the project through the entire process" (Fennie and Mehl, 2003).

As experts, programmers and systems analysts create the system. Often it is concluded that users do not understand the technological requirements, and therefore are not 'useful' during this creative process. Then the focus is on ensuring the requirements of the physical world rather than necessarily meeting the expectation of the user. What has been missed in this assumption is that the users are experts in how the system needs to be used, and therefore should play an important role in its creation from start to finish.

The situation is changing. Evidence of this can be seen in one of the latest methodologies for software engineering. Extreme Programming is considered a radical new method of programming. It relies on programmers and customers working in the same environment and a design-as-you-go approach (Krill, 2002).

Comments from a panel of experts at a CTO forum in April 2002 about Extreme Programming included such statements as:

"Part of what changes when a customer is in the same room is the software ends up getting used…There's just a whole emotional binding process that happens when involving the whole team through the whole process." Rowland Archer, CTO at Haht Commerce, in Raleigh, N.C.

"If you can get people in the same room, you get people at least communicating, in ways that they otherwise might not have" Dave Burleigh, Technology Visionary, ValutationRepairman.com (both cited in Krill, 2002).

It should not take radical programming methodology to reach the conclusion that the more the users are involved in the system design, the more likely we are to build a system that will get used. They may not be able to solve the data structure questions, but they could give all the help necessary for the light bulb to be turned on in a programmers head.

3. ILLUSTRATIVE EXAMPLES

The following four examples have been written from the personal experience of the author. They are not intended to be case studies of the organisations: they have been written to give enough detail to highlight the points raised. The organisations come from a number of different countries and sectors, over the past three years. No names have been included to protect the privacy of the organisations concerned, and to maintain the professional confidential relationships the author has with these organisations. The examples have been chosen to highlight the issues, rather than a detrimental analysis of the organisations concerned.

Example 1

A telecommunications company was looking to streamline its systems. The company had started out as the amalgamation of a number of smaller companies, and had many legacy systems making up its operations. As an example, it had 24 separate billing systems. The majority of these had come from the previous companies, and most only processed a few, but very important customers. Much of the billing to the largest 20 customers was very customised to how they wanted to receive their information. These customers included whole small countries, and it was considered very important from a marketing point of view to keep them happy. If a new billing system did not meet the needs completely, those customers were left on their old one. Marketing did not consult with Billing (and vise versa) as to the possibility of new features or alternative methods, as they considered that if the customer was happy already, they were not going to upset them with a new way of charging.

The operations side of the business also had a number of software applications, that had been created and evolved as new telecommunications systems became available. However the old networks could only be run by some of the old applications, as they were too entrenched to move around.

The company had decided that with new products and a more global perspective, they wanted integrated, end-to-end systems. It enlisted the services of an IT consulting company, that sought to implement three new linked systems for Customer Relationship Management (CRM), Order Management and Billing. This new 'super-system' would replace the existing legacy systems.

It was the job of the consulting company to configure the new system, eliminating legacy where possible and incorporating it seamlessly where it was not possible.

Many people were involved in the configuration of the new system, from within the telecommunications company and the consultant company. There were approximately 300 people linked to the project in some way. They were mostly housed in one building, that was in a different city from the majority of the users (who were spread across many cities).

The process people were largely business analysts that had come from a systems background. They documented the new processes end to end. The difficulty was that some of the crucial legacy systems in Order Management (mostly related to the allocation of telecommunications network resources for customers) could not be replaced, and therefore required work-arounds. The systems people designed the interfaces between the new and old, and tested them internally. The process people documented how they would flow between one system and the next, based on what the systems people had done. When the trainers started working on how to train the users, they discovered that the system had turned a five minute task into a 25 minute one, as much of the information from the old system had to be back-filled manually into the new system in a number of different screens. They raised the alarm that users might refuse to accept the new system, but the IT consultant company was too far through the implementation to be able to change. The telecommunications company management initially approved the system, but when users refused to use it, they rejected it and after two more attempts over three months to improve the timing, the IT company was eventually removed from the job with large losses. The billing system was implemented, but became the 25th billing system that the company maintained. The CRM system was configured and modules of it were used, however it did not link to any order system, and operated stand alone.

In hindsight, many lessons were learned. The project was ambitious, and as such was conducted mostly behind closed doors. The 300 people working on the project mostly communicated with each other, rather than getting out into the user groups. Many had not been to the locations that the users of their system would be operating in, to see what they did and how they did it. They were doing ‘blue sky’ development, the perfect system. They did not want to take the ‘baggage’ with them, so chose to ignore that fact that it may exist. They discovered in hindsight that one or two ‘friendly’ superusers do not necessarily represent the organisation, and that the baggage does not go away just because we cannot see it. Process people should work with the people, not the system designers. They should be telling the designers what is needed, not the other way around. If the system could not be successfully configured, it would not have been purchased from the vendors in the first place, rather than letting the IT company take the loss after the system could not deliver. Had the users been involved in configuring the system, seeing the benefits of the new one and helping creating the workarounds, there could have been acceptance.

Example 2

This example concerned an educational institution’s student management system. The system collected student information - mostly personal details, academic history and course details. The system was designed and built in-house and had grown over the years as needs were identified. Many sections used this system, from front line staff, through to enrolment staff and departments. The computing department were moving to a Windows environment and decided to take the existing student look-up module and recreate it in a GUI environment. Given that it was mostly an exercise in how they would develop in the new tool, they relied on their knowledge of how the system worked to create the new look. Without consultation with the users as to how they interacted with the look-up screens and what they felt could be improved, a system was created and released it to the users.

The old system had its uses, but had serious limitations in terms of functionality. Users had to go to other parts of the system to act on what they had found in the look-up part. The users found all the frustrations with the old system were contained in the new one, and consequently there was limited take-up of the new application. The new system only contained what the old one did, but because users now had to go between two systems, rather than between two modules of the old system, it did not add the value it could have done.

Example 3

This involved a visit to a bank. While opening an account at a bank, the author experienced an interesting situation. The customer service person needed to print some forms to open the account from their local intranet. The intranet had been recently revamped, and from a non-user prospective looked well organised, and visually interesting. However, the forms the officer required were not where she was expecting them to be. She found one, but could not locate the second one needed. Luckily there was a link on the new site to the old intranet, which had not been taken off-line. She quickly went to the location where all of the forms were, and printed the required ones.

This leads the author to question if the process of setting up a new customer was considered when designing the new intranet. If it was, surely all of the forms would have been in the one location.

Example 4

An investment bank had been taken over by another bank two years earlier. The decision was made to implement the parent bank’s system for storing and valuing share portfolios. The problem was that the old system did not require the rigourous balancing of the new one, and the data was too corrupt to do an easy migration. They had to employ a large number of temporary staff to clean up the data and migrate to the new system. These staff had little idea of what was required, and took a long time to get up to speed with what the new system was supposed to do, and the process by which a portfolio was valued. When the job was completed after many months, they were then given the task of documenting the process of how to value a portfolio - something they would have found very useful at the start. Many hours were lost by using the new system incorrectly, as the process was not clear. Had the project started with the process analysis instead of going straight into the data conversion, it would have been less stressful for the team and the customers.

4. SUGGESTIONS FOR IMPROVEMENT

When approaching a systems development, it is important to realise that there is more to the exercise than the technology. Long after the developers have completed the system, the users will still have to make it perform their tasks. Therefore they have a huge stake in what is developed, and often their concerns are overlooked, they are labeled ‘difficult people’ or their requirements are not developed due to constraints of time or programmer skill level (ie we do not know how to do it that way). None of this information is new, but it has been the experience of the author that it continues to be overlooked. Therefore a framework for ensuring that these factors are considered is a useful tool.

As part of the plan for a systems development, a people perspective needs to be highlighted. The following factors should be considered as part of that plan:

4.1 Problem

Make sure that the system being developed is actually going to solve the problem. In the telecommunications company case, they had 24 different billing systems. The last five were supposed to be the one that replaced them all. They ended up just adding to the list, because they could not meet the needs of all the existing customers. In the educational institution's case, a product was developed that no-one had asked for. We did not solve any of the problems the users had, so there was limited use of the product.

4.2 Previous work

Careful notice needs to be taken of what has occurred previously. This will alert developers to the possible minefields that might be lurking in an organisation. In the telecommunications case, an alert would be the increasing not decreasing number of billing systems that had been implemented over the years. Why was that? What were the lessons learned previously? What can we do differently?

4.3 Procedure/Policies/ Perspective

What is the organisations procedure for approaching staff? How much access will the developers have to the people who will be using the system? If management provide a 'superuser', how much do they know outside their own job description?

What are the policies for implementation? How difficult will it be for people to access the system? What training is envisaged, and to who?

The project needs to be put in perspective of the wider organisational requirements. What other projects are underway? What is the organisation's focus? How important is the project and how critical is the timeline?

4.4 People/Politics

A careful study of previous work, and conversations with key champions of the project may alert the developers to the people and political issues that may surface with the proposed project. It is a good idea to get the users involved early, to discuss any barriers that might be on the horizon. When we need to change processes and systems, there is much discussion about difficult people, and blockers to any change – these people need to be assessed early to determine whether they have genuine concerns, and if they do, to work through them.

The advantages with them being involved throughout the whole project include:

♦ They may have experience from a wide range of development projects which enables them to add to discussions about issues that other users may have with the product.

♦ They can highlight issues such as time to do tasks and difficulty of operating.

♦ They can provide valuable feedback for corrective action throughout development before the software goes live.

♦ They can assist testers in the development of UAT material.

♦ They can excite other users to want to use the product.

♦ As they have working knowledge of the product, they can troubleshoot minor problems if required during implementation.

♦ They can help create full resources (a programmers nightmare task), so others can refer to them after the development is over.

4.5 Process

A good system helps streamline a business process. Be clear about what the process is, and what the requirements are in terms of input and output. Look at exceptions to the process - maybe in peak periods it is done differently. If the new system has not allowed for that, it may that it is unable to be used in that time, and the old system will have to remain and be maintained.

When documenting individual processes, look at where they fit into the overall end-to-end process. This will give an indication of how seamless the integration and implementation of the new system will be.

5. CONCLUSION

Project development has been happening in other industries for centuries. Tall buildings, ships and space rockets are all examples of projects that have been completed. These disciplines have all gained experience in dealing with the human factor in their projects, and while not always successful, they are a long way towards getting it right.

Although analysing the human factor in software development may not be considered new or novel, the illustrative examples used in this paper have demonstrated that there are still recent situations within the IT industry where we are not getting it right, and therefore it is a subject which should continue to be explored and reiterated. If the users are not consulted or involved throughout the software development process, the likelihood of the software getting used to its potential is greatly diminished.

The good news is that the software development industry does not need to go through hundreds of years of learning to get it right. We are experts in the technology: we need to look at other disciplines for the pieces we are not experts in. These people are all around us, but because they are not IT professionals, it is easy to overlook them. You do not have to know how to write the software to manage a team of people or develop processes for a new system. In fact, it may be an advantage not to. Our process people need to be experts in people, rather than systems.

By ensuring that our preparation is sound, we can start to make the significant gains required, so our industry as a whole can come of age.

REFERENCES

Fennie & Mehl, 2003. Website. Accessed 10.3.03. http://www.fm-arch.com/Firm/design.htm

Jones, C. 1994. “Assessment and Control of Software Risks”. Englewood Cliffs, N.J.: Yourdon Press

King, J. 1997. “IS reins in runaway projects:users fight failures with better management.” Accessed 10 March 2003. http://www.computerworld.com/news/1997/story/0,11280,14265,00.html

KPMG. 1995. “Runaway Projects — Causes and Effects”, Software World, vol. 26, no. 3.

Krill, P. 2002. “CTO Forum: Taking programming to the extreme”. Accessed 25 February 2003 http://www.infoworld.com/articles/hn/xml/02/04/10/020410hnctoxp.xml

McConnell, S. 1999. “After the Gold Rush: Creating a True Profession of Software Engineering (Best Practices)”. Microsoft Press.

Turcotte, J. (1997) Follow-Up Report on the Florida Department of Transportation’s Performance In Controlling Cost Overruns When Building Roads and Bridges. Accessed 10 March 2003. http://www.oppaga.state.fl.us/reports/pdf/9722rpt.pdf