Failure IS an Option!

Ever worked on a project that failed? I don't just mean it ran poorly or didn't meet all of the user's needs – I mean it failed SO spectacularly that you refer to the project as a resume stain. I have worked on one project (not to be named, of course) that from day one of the project, the project was secondary – trying to find a new project to transfer to became paramount.

During my 23 years in the Air Force, I had many friends whose careers were tarnished (if not crippled) from association with high profile projects that failed spectacularly. Some projects fail from technical reasons – it's a sad fact that in the DOD, we are often pushing the "bleeding edge" – and we just don't have the technology to complete it. Some projects failure from the sheer size – trying to create 100+ million lines of code projects are doomed from the complexity.

And some fail because we could do better.

I ran across an interesting article online – from a web site called outsource2india.com (there's probably another Backtalk on that topic later). They listed 10 reasons large projects fail[i]. They are:

1. Miscalculated Time and Budget Frames. Well, YEAH. Nobody really knows the time, budget or functionality until AFTER the project is done. You need to plan on this – probably only 25% if software projects meet schedule and budget goals, let alone quality and "essential" requirements.

2. Was it Needed at All? Can you list projects you've worked on that there was not really a business case for? Not all ideas are good – but sometimes it's not politically expedient to say "Are you nuts? Nobody really needs this!". I found a reference to the - The FAA Advanced Automation System (1981-1994) : Cost of $3.7 billion; peak staffing of 2000; maximum run rate of $1 million/day; 13 years duration of development. - Nothing was delivered; no code was ever used. The reason: nobody wanted it in the first place or in other words there was no business case for it

3. Lack of Communication. Five people can communicate. 2000+ developers and 100s of users can't. But you knew that. How many projects have you worked on where the end-users couldn't be identified? Speaking of which…

4. No End-user Involvement, and its companion…..

5. Unfocused Executive Sponsors

6. Failing to See the Bigger Picture. You have to see the big picture, and use common sense. In the UK, officials called off what was considered to be the largest public IT project of all time. It was a project which was intended to provide electronic health records for all of its citizens. After 10 years and costing an estimated 19 billion USD the authorities concluded that the project was not fit to provide the modern services it was intended to. DO YOU MEAN TO TELL ME THEY SPENT $19 BILLION BEFORE THEY DETERMINED IT WASN'T GOING TO WORK? Yep – as long as everybody is getting paid and the money is flowing – why cancel? By the way - $19 billion seemed impossibly high – but I found several references[ii] – they really spent that much!

7. Chasing Technology. Ada. DODAF. DIICOE. New technologies are good – but they are NOT a "Silver Bullet". And if you haven't read Brooks' "No Silver Bullet" paper[iii] – stop now and go read it. I'll wait.

8. Development Downtime. Debugging and fixing errors is going to take longer than you think, even with new time-saving languages and tools. Plan on it. Then double the time planner.

9. Lack of Periodic Assessment. If you don't know where you are – sort of hard to figure out when (or if) you're going to be done. Honesty comes into play here – the urge to report "Almost done" when you really have no clue is overwhelming.

10. Lack of Quality Testing. Testing with live users. Integration testing with all other systems. Stress testing. Security testing. If you don't test, the users will – after you deliver it. And it becomes a failure.

Mind you, there are LOTS of lists as to why projects fail – IEEE Spectrum has a similar list.[iv] Their list is just a list of the COMMON reasons for failure – implying that there are LOTS MORE. Depressing, isn't it?

- Unrealistic or unarticulated project goals
- Inaccurate estimates of needed resources
- Badly defined system requirements
- Poor reporting of the project's status
- Unmanaged risks
- Stakeholder politics
- Use of immature technology
- Poor communication among customers, developers, and users
- Inability to handle the project's complexity
- Sloppy development practices
- Poor project management
- Commercial pressures

When I read such lists, I feel depressed. To make myself even more depressed I go and browse a website called "Coding Horrors". The article[v]"The Long, Dismal history of software project failure" (based upon the IEEE article referenced above) is a real pick-me-up – it starts by pointing out that Sainsbury, the UK supermarket giant, had to write off a $526 million automated supply-chain management system just last October (2015) A half-billion-dollar failure.

Want to feel better? I'm not much help. Software will fail, for all the reasons listed above and more. It's the nature of what we do for a living. BUT – there is a bright side, according to the article: Learn from failures! "Failing is OK. Failing can even be desirable. But you must learn from your failures, and that requires concerted postmortem introspection and analysis. …. Once you know what the common pitfalls are, it's easier to avoid them".

If you are going to fail (and, sad to say, you probably will) at least LEARN from the failures. Don't make the same mistakes again, or at least try to mitigate similar mistakes in the future. Learn, take a deep breath, and move on. Statistics are hard to come by – but some estimate that 50% of all software projects will fail, and about another 25% will succeed but not be used. As mentioned above, only about 25% meet time and budget targets. It's the price we pay for being on the cutting edge of technology. It's the nature of the beast. We do the best we can, learn from the mistakes, and sign up for the next project that comes along. If I'm going to fail, I'm going to go down swinging! You really don't learn that much from successes – but failure? That's where to really get an education!

David A. Cook, Ph.D.
cookda@sfasu.edu
Stephen F. Austin State University

NOTES

[i] https://www.outsource2india.com/software/pricing-structure.asp
[ii] http://www.independent.co.uk/life-style/health-and-families/health-news/nhs-pulls-the-plug-on-its-11bn-it-system-2330906.html
[iii] http://worrydream.com/refs/Brooks-NoSilverBullet.pdf
[iv] http://spectrum.ieee.org/computing/software/why-software-fails
[v] https://blog.codinghorror.com/the-long-dismal-history-of-software-project-failure/