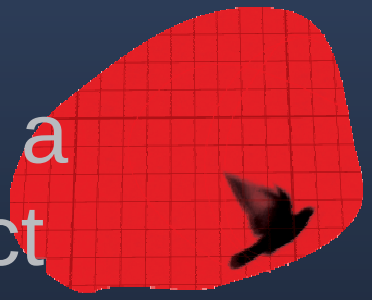
O'REILLY®

Why Software Projects Fail (Part I)

How to diagnose and fix a troubled software project



Lack of Leadership

- It takes more than a talented and motivated team to make a successful project.
- Lack of leadership manifests itself in the team members suffering from:
  - Tunnel vision
  - Over-reliance on gut instincts
  - Repeated false starts in the project

The Mid-Course Correction

- A change in project priorities throws the team into disarray
- This usually comes from a lack of understanding of the scope of the project
- When the engineers don't understand the users' and stakeholders' needs, they build the wrong software
  - And they might not find out that there's a problem until after the work is done!

The Detached Engineering Team

- There is an artificial wall between the people who build the software and those who need it.
  - The business people feel like the engineers are moving too slowly and don't care about their needs
  - The engineers feel like they're always shooting at a moving target because business people don't know what they want

Fixing Planning Problems

- Lack of Leadership, the Mid-Course Correction and the Detached Engineering Team are project planning problems
  - Use a vision and scope document to define the needs of the users and stakeholders
  - Use a project plan to keep every informed about how those needs will be met
  - Use risk planning to keep the plan realistic

Padded Estimates Generate Distrust

- Programmers add extra time to their estimates
  - They may do this because of unknowns
  - Often they have been late in the past, and "know" that they will need extra time
- Project managers and senior managers quickly figure this out, and start to question individual estimates
  - And the programmers don't have good answers!

Self-Fulfilling Prophecy

- A project manager under pressure simply imposes a deadline, and creates unrealistic estimates that meet it
- The team works nights and weekends to meet the deadline
- The project manager feels vindicated
- The team eventually gets frustrated and disillusioned

Fixing Estimation Problems

- Padded estimates and the self-fulfilling prophecy are estimation problems
  - Adopting a repeatable estimation process like Wideband Delphi can help fix them
  - By writing down assumptions, the team can handle risks without padding their time – and even avoid the risks altogether
  - It reduces padding and increases honesty through transparency, by letting the team correct each other in an open meeting

Working Backwards From a Deadline

- Project managers approach a non-negotiable deadline for a project by working backwards
  - They shorten the tasks in the schedule or cutting them entirely until everything fits
  - When the schedule gets tight, any non-programming activities are cut and the software is released before it's finished

Misunderstood Predecessors

- The project manger does not take the time to understand how tasks depend on each other
- Problems are discovered partway through the project one task can't be started because it depends on another
- Delays cascade through the project, getting increasingly worse
- Some programmers are stuck waiting with nothing to do, while others work overtime

Fixing Scheduling Problems

- Working backwards from a deadline and misunderstood predecessors are symptoms of underlying scheduling problems
  - They can be avoided by adopting good planning and estimation practices and creating a project schedule
  - Schedule techniques like critical path analysis can help spot problems early on

Problems Are Found Too Late

- There are preventable defects in the software that aren't caught until late in the project
  - The team may misunderstand a need, but that's not discovered until delivery
  - Requirements may be missed or incorrect
  - The design may be difficult to use or fail to take all of the features into account

Big, Useless Meetings

- A project manager who has previously been burned by problems that were found too late is determined to avoid falling into the same trap
  - He calls a big meeting with everyone who could possibly have input
  - The meeting drags on for hours, without making any real progress
  - Eventually, everyone gives up and goes back to the way they did things before

The Indispensable "Hero"

- One "critical" person is seen as the clear top programmer, and all important work is sent through him
  - He may have a unique skill or experience
  - Sometimes he hoardes information so all tasks that rely on it must go through him
  - He is always working long hours – and causing bottlenecks

Fixing Review Problems

- Problems that are found too late, big useless meetings, and the indispensable "hero" are problems which can be solved with reviews
  - Reviews can catch defects early, when they are cheaper to fix
  - A review meeting only includes the people necessary for the work to be done
  - Reviews – especially code reviews – can help the "hero" spread his expertise and knowledge