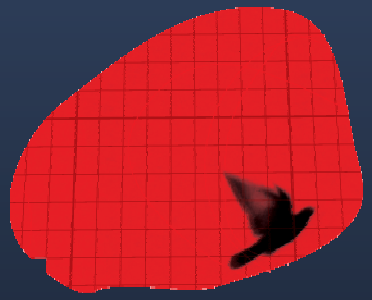
O'REILLY®

Applied Software Project Management

Introduction



Why do software projects fail?

- People begin programming before they understand the problem
  - Everyone likes to feel that they're making progress
  - When the team starts to code as soon as the project begins, they see immediate gains
  - When problems become more complex (as they always do!), the work gets bogged down
  - In the best case, a team that begins programming too soon will end up writing good software that solves the wrong problem

Why do software projects fail?

- The team has an unrealistic idea about how much work is involved.
  - From far away, most complex problems seem simple to solve
  - Teams can commit to impossible deadlines by being overly optimistic and not thinking through the work
  - Few people realize the deadline is optimistic until it's blown

Why do software projects fail?

- Defects are injected early but discovered late.
  - Projects can address the wrong needs
  - Requirements can specify incorrect behavior
  - Design, architecture and code can be technically flawed
  - Test plans can miss functionality
  - The later these problems are found, the more likely they are to cause the project to fail

Why do software projects fail?

- Programmers have poor habits – and they don't feel accountable for their work.
  - Programmers don't have good control of their source code
  - Code written by one person is often difficult for another person to understand
  - Programmers don't test their code, which makes diagnosing and fixing bugs more expensive
  - The team does not have a good sense of the overall health of the project.

Why do software projects fail?

- Managers try to test quality into the software.
  - Everyone assumes that the testers will catch all of the defects that were injected throughout the project.
  - When testers look for defects, managers tell them they are wasting time.
  - When testers find defects, programmers are antagonized because they feel that they are being personally criticized.
  - When testers miss defects, everyone blames them for not being perfect.

How can we make sure that our projects succeed?

- Make sure all decisions are based on openly shared information
  - It's important to create a culture of transparency, where everyone who needs information knows where to find it and is comfortable looking at it.
  - All project documents, schedules, estimates, plans and other work products should be shared with the entire team, managers, stakeholders, users and anyone else in the organization who wants them.
  - Major decisions that are made about the project should be well-supported and explained.

How can we make sure that our projects succeed?

- Don't second-guess your team members' expertise
  - Managers need to trust team members.
  - Just because a manager has responsibility for a project's success, it doesn't mean that he's more qualified to make decisions than the team members.
  - If you don't have a good reason to veto an idea, don't.

How can we make sure that our projects succeed?

- Introduce software quality from the very beginning of the project
  - Review everything, test everything.
  - Use reviews to find defects – but don't expect the review to be perfect.
  - Use reviews to gain a real commitment from the team.
  - It's always faster in the long run to hold a review than it is to skip it.

How can we make sure that our projects succeed?

- Don't impose an artificial hierarchy on the project team
  - All software engineers were created equal.
  - A manager should not assume that programming is more difficult or technical than design, testing or requirements engineering.
  - Managers should definitely not assume that the programmer is always right, or the tester is always raising false alarms.

How can we make sure that our projects succeed?

- Remember that the fastest way through the project is to use good engineering practices
  - Managers and teams often want to cut important tasks – especially estimation, reviews, requirements gathering and testing.
  - If it were faster to build the software without these practices, we would never use them.
  - Every one of these practices is about saving time and increasing quality by planning well and finding defects early. Cutting them out will cost time and reduce quality.