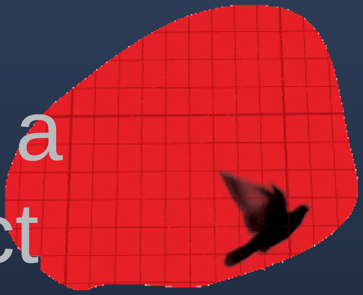
Why Software Projects Fail (Part II)

How to diagnose and fix a troubled software project



Iteration Abuse

- Iteration can be a useful tool, but it is often abused
- The team uses iteration as a “guessing game”
  - Programmers deliver build after build; users and stakeholders make small changes to each build
  - Programmers like it because they can dive in
  - Users and stakeholders like it because they don’t have to read documents or think about their needs

O'REILLY®

Scope Creep

- After the programming has started, users and stakeholders make changes
- Each change is easy to describe, so it sounds “small” and the programmers agree to it
- Eventually, the project slows to a crawl
  - It’s 90% done – with 90% left to go
  - The programmers know that if they had been told from the beginning what to build, they could have built it quickly from the start

Fixing Requirements Problems

- When software requirements are not gathered and specified before the software is developed, it causes scope creep and the team resorts to iteration abuse.
  - The team can adopt software requirements engineering practices to write down most of the changes before the work begins
  - A change control process gives them a handle on the few changes that remain

Haunted by Ghosts of Old Problems

- Programmers find that old bugs suddenly reappear without warning
  - As the code base grows, it becomes harder to keep control of the source code
  - They may use a shared folder to store source code, but occasionally old copies of files are copied over newer ones

Broken Builds

- The programmers deliver a build which does not work – and the testers can't even begin to test it
  - The programmers get frustrated because they feel that they put a lot of effort into testing the build
  - "Isn't it the job of the QA team to figure out the build is broken and tell them what to fix?"
  - The testers spend hours or days setting up a test environment, only to redo it for the next build

Spaghetti Code

- Maintaining old code is the least desirable programming job in many organizations
  - Old, highly modified code turns into a twisted mess of execution paths and patches
  - Spaghetti code is often used as an excuse to do an unnecessary rewrite

Fixing Programming Problems

- When the team adopts good programming habits, they can avoid ghosts of old problems, broken builds and spaghetti code.
  - Get control of the source code with version control software like Subversion
  - Use unit tests and test-driven development to increase the quality of the build
  - Use refactoring to keep the code readable

Requirements Haven't Been Implemented

- The team delivers software missing behavior or even entire features
  - Software is complex, and even with good review practices, it's difficult for programmers to fully implement everything
  - Missing requirements are difficult to spot because even the programmer missed them when looking over his own work

Obvious Bugs Slipped Through

- Inexperienced testers are expected to just “bang on the software”
- Technical support staff, junior programmers, end users, outside temps and sales people are drafted as “testers”
- Even when experienced testers are used, they are not given time to plan
- Decisions about release readiness are made based on the schedule, rather than the quality

"But It Worked For Us!"

- When a product is not tested in all environments in which it will be used, the tests will be thrown off
- Defects are missed when testers can't adequately replicate the environment in which the software will be used
- Test data may not resemble actual production data

Fixing Testing Problems

- When code is delivered with too few requirements implemented and too many bugs included, the team needs better testing practices.
  - ▷ Software testers must be involved in every stage of development
  - ▷ Test planning must be given adequate time on the schedule
  - ▷ Sufficient budget must be provided for a testing environment.

Common Problems Can Be Avoided!

- Almost everyone has experienced at least a few of these problems.
- We know what causes them, and we have tools, techniques and practices that can fix them.
- All it takes is good project management and sound software engineering… and any project team can do it!