Sabotaging Quality

Joy Shafer

October, 2011

Introduction

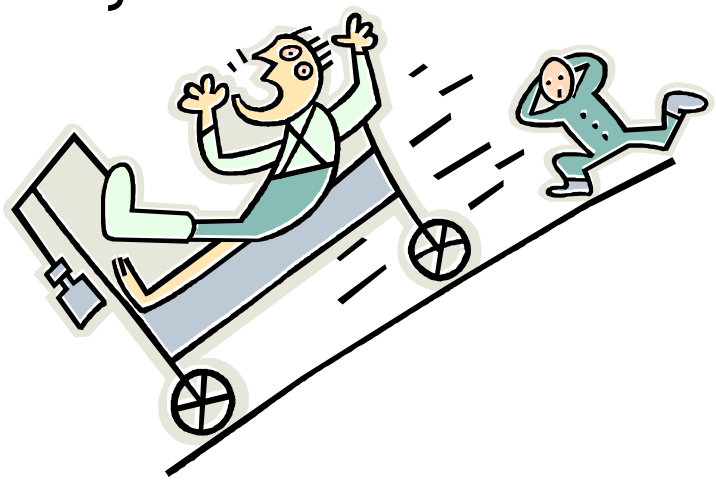
- Why am I here?
- Why are you here?

Sabotaging Quality

Sacrificing long-term for short term

- Investment in test, particularly test automation
- Software quality
- Ability to maintain software or service
- Investment in infrastructure
- Employee morale and vitality

Most of the time, no one sets out to sabotage quality, things just get away from them.



Investment in Test

- Automated BVTs
- Continuous Integration
- Unit testing
- Automated regression tests
- The ability to emulate external components

Automated BVTs

- Robust (reliably functional)
- Not dependent on other systems or services
- Definitive
  - If it fails it's because of a code issue not a timing issue, data issue, setup issue, etc.
- Start with basic tests and add on

Continuous Integration

A cornerstone of efficient software development

- Daily integration
- Scheduled BVTs & Automated Regression tests

Adequate test automation is required

Unit Testing

- Cleaner code
- Better designed code
- Tools
  - Test hooks
  - Mock objects
- Better appreciation by developers of the challenges testers face

Management support is required

Automated Regression Tests

Without automated regression testing, teams of manual testers will need to comb through the software with every release, laboriously running the same tests they've run countless times before.

The ability to emulate external components

- Don’t get too complicated
- Check with other teams for emulators you can modify
- Keep up-to-date
- Extremely useful, maybe essential, for performance testing

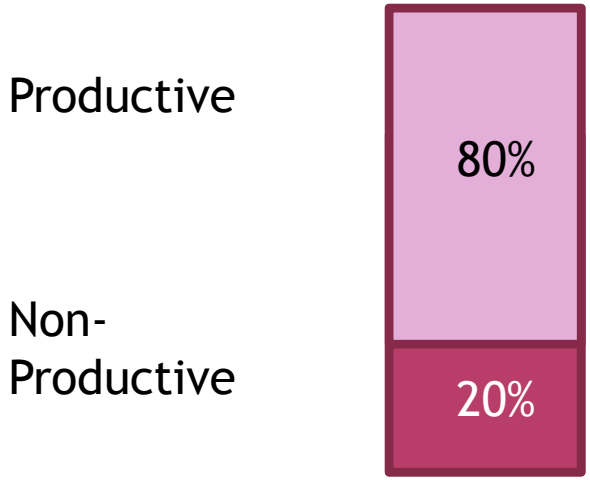
Invest early, invest often

Software Quality

- Fix bugs as you find them
  - Less time overall on bug fixes
  - Ship cycles will be shorter and more predictable
  - No backlog to carry from release to release
  - Cleaner code
- Consider refactoring
- Techniques for quality improvement
  - Do a ‘quality release’
  - Implement a ‘bug jail’

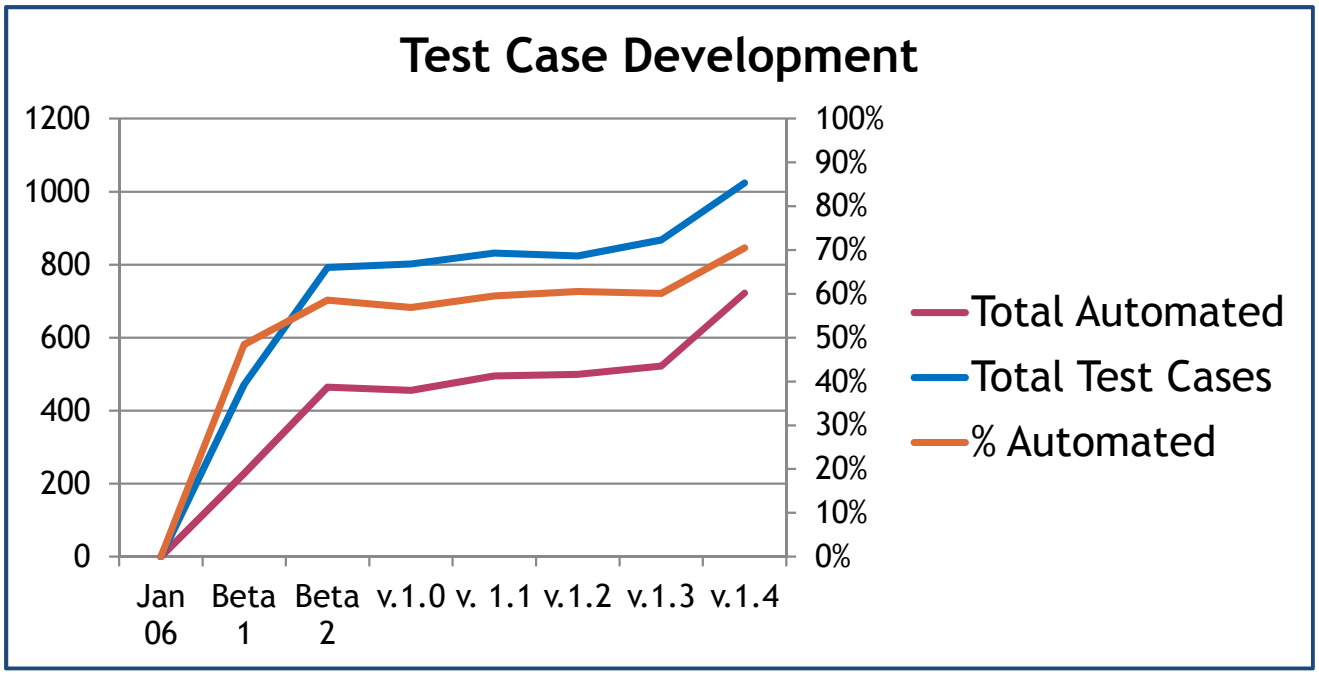
Ability to Maintain your Service

- Think about maintenance during the design phase
- Design for ease of deployment
- Keep hardware/OS/software current



Investment in Infrastructure

- Physical Infrastructure
- Logging
- Metrics



Employee Morale and Vitality

- Long hours lead to
  - Low productivity
  - Job dissatisfaction/Burnout
  - High turnover
  - Mistrust of management

Strategy: Appropriate Rewards

- Software development is a team sport
- Reward successful teams

Tactics

- Know best practices
- Be polite but persistent
- Find your allies
- Understand the real problem
- Measure yourself
- Set clear goals
- Celebrate success

Most importantly, don’t give up!

Questions