Seven Deadly Sins of Testing

Pitfalls on the Path to Software Quality

Introduction

- Have you seen a smart, otherwise-capable tester or test team sabotage themselves?
- It does happen, often through one or more of the following
  - Irrelevance/redundancy
  - Ignorance of relevant skills or facts
  - Obstructionism
  - Adversarialism
  - Nit-picking
  - Blindness to project/organizational priorities
  - Last-moment-ism
- What are these seven deadly sins, how can they be stopped, and how can the problems be solved?
- Let's take a look…

Irrelevance or Redundancy

- Testing what doesn't matter or repeating tests already run by others
- How to avoid this
  - Use smart test selection strategies (e.g., risk-based testing)
  - Understand what others are testing (e.g., unit tests, in some cases UAT)
- When test teams have made this mistake in the past, close coordination with other test stakeholders is important to show progress

Irrelevance Case Study

- One test team (mine) focused on automating combinations of OS/DBMS
- Meanwhile, important installation and usability bugs escaped
- Heavy focus on automated accuracy testing was seen as irrelevant to higher quality
- We had to add additional manual tests to cover these other areas
- The reputational damage proved very difficult to undo

Ignorance of Relevant Skills or Facts

- Lack of technical, testing, or application skills
- Lack of awareness of organizational or user context
- Test managers should
  - Use task analysis to identify critical skills
  - Create and execute a test team development plan
  - Become well-connected with other stakeholders
- If your team's been guilty of this sin, be sure to market the strides you're taking to improve skills and awareness

Ignorance Case Study

- More than half of people working as testers don't know what an equivalence partition is
- Even fewer know sophisticated test design techniques
- Proponents of ignorance increase the trouble in our profession
  - Beliefs that anyone can test
  - Opposition to structured training
  - Schools of testing as barriers to discourse and exchange of ideas
- Fortunately, the ISTQB program is changing this situation

Obstructionism

- Getting in the way rather than helping
  - Insisting on rigid adherence to entry or exit criteria when conditions have changed
  - Persisting with the exaggeration of obstacles
  - Refusing to adapt to changing needs (e.g., Agile methods, test automation, etc.)
- If the root cause is a misunderstanding of the role of testing (e.g., "process cop"), clarify the test team's role
- If the root cause is a contrary personality, then it's harder to solve and might require personnel changes
- A history of obstruction can create a harmful reputation that requires time and effort to repair

Obstructionism Case Study

- One tester was relentlessly negative
  - Product quality bad
  - Management team incompetent
  - Developers stupid
- This tester was very good as a tester, and as a test lead
- However, his attitude limited his ability to rise above those roles
- While respected by fellow testers, he had problematic relations with non-test staff
- As a contractor, this attitude inhibited his ability to get work

Adversarialism

- Deciding (or being told) that testing is about enforcement of process or quality, and assuming an adversarial role (esp. with development)
- Adversarialism can be resolved through careful clarification of the proper test role, mission, and objectives with stakeholders and managers
- Bad feelings and grudges can persist for months or even years afterward

Adversarialism Case Study

- Test team implemented and enforced rigid process on all other teams
- Worst practices in tool selection process were used, resulting in a hated tool
- This was done at CIO direction, but without political cover
- A poisonous relationship developed
- Ultimately the test team was disbanded and all testing work outsourced

Nit-picking

- Inflating the criticality of defects or reporting trivial defects, especially if important defects are escaping
- Solutions include:
  - Fixing the bug triage process so it's not a poker game
  - Ensuring uniform understanding of priority and severity ratings across all project participants
  - Identifying "repeat inflaters" and fixing their attitude
  - Eliminating any sort of ego-identification ("my bugs")
- Once corrected, the problems associated with this can go away, but credibility takes time to regain

Nit-picking Case Study

- Numerous testers have told me they inflate bug reports to ensure more of them get fixed
- In assessments, when I hear that, I often hear from programmers than they ignore tester ratings
- In some cases, just a few examples of inflated bug reports suffice to tar the reputation of the entire team
- While this tactic can work at first for individual bugs, I've never seen it work in the longer term

Blindness to Priorities

- Lack of awareness of what is critical for project or organizational success
- Can include deflating defect importance or avoiding tests due to individual sensitivities
- If the cause is an excessively isolated test team, build stronger relationships with stakeholders
- If the cause is siloing, work with your manager to open better communication channels
- While this can lead to serious mistakes, the symptoms go away quickly once the problem is solved

Priority Blindness Case Study

- One organization found itself under severe financial constraints during a project
- Developers were kept
- Half the testers were let go
- Test manager was unable to explain the value of testing in terms that related to organizational priorities
- When testers and test managers don't understand objectives and priorities, or can't demonstrate achievement of them, success is unlikely

Last-moment-ism

- Finding important bugs right at the end of a test execution period
- If due to improper sequencing of tests, institute risk-based testing (which you should anyway)
- If due to broken risk-based testing, fix what's broken (e.g., missing stakeholders, rating inflation, etc.)
- If due to late reactive testing, integrate reactive testing throughout test execution
- It usually takes a couple projects before this problem can be completely resolved

Last-moment-ism Case Study

- On one project, testers continued to find major problems late in test execution
- Many of these problems could have been found earlier
- Risk was not used to sequence tests or allocate test effort
- Programmers and project managers were unhappy last-moment surprises
- The testing service provider was not brought back for the next project

Conclusions

- Testers and test managers sometimes make simple mistakes that cause big problems
- Recognize the seven deadly sins of testing so you can resolve them
- Don't assume the damage is over when the sinning is done
- Marketing the successful resolution of these problems is often critical

To Contact RBCS

For almost 20 years, RBCS has delivered consulting, outsourcing and training services to clients, helping them with software and hardware testing. Employing the industry's most experienced and recognized consultants, RBCS advises its clients, trains their employees, conducts product testing, builds and improves testing groups, and hires testing staff for hundreds of clients worldwide. Ranging from Fortune 20 companies to start-ups, RBCS clients save time and money through improved product development, decreased tech support calls, improved corporate reputation and more. To learn more about RBCS, visit www.rbcs-us.com.

Address: RBCS, Inc.
31520 Beck Road
Bulverde, TX 78163-3911
USA
Phone: +1 (830) 438-4830
Fax: +1 (830) 438-4831
E-mail: info@rbcs-us.com
Web: www.rbcs-us.com