Software Testing Is a Game

It may feel like skilled testing is under attack. Fight back by changing your perspective.

by Jonathan Kohl | jonathan@kohl.ca

We're undergoing an enormous shift in technologies right now. As much as we innovate, failures inevitably occur. One way we mitigate against the damage those failures can cause is through testing. As new technology makes computing more pervasive, we have more computing power in cars, homes, public buildings, and in medical and life-support systems than was imaginable not too long ago. As computing devices become more enmeshed in our everyday lives, there is an even greater opportunity for them to cause harm. With new technology comes new opportunities for great things, but there are inevitable, unintended consequences. Skilled, systematic testing can help discover many of these.

But, it feels like skilled testing is under attack. People who see testing as a necessary evil want to outsource it to tools or to other people. This undermines the craft and threatens the value of the products we depend on. Sure, there is a lot of repetition in testing that might seem boring, but many of us are perfectly happy to perform repetitive tasks in other situations. Why do we dislike it in testing? Maybe our approaches to testing need a rethink.

> "Analyzing human group activity by looking at game mechanics is helpful when we want to challenge our assumptions."

Is software testing like a game? Before you start thinking about hopscotch or gold stars, hear me out. In our article in the September/October 2012 issue of *Better Software* magazine, David McFadzean and I describe a game as any situation that involves cooperation and conflict. As testers, we collaborate with team members to help solve problems, and as we point out quality-related information, we end up in situations of conflict. It's a fascinating balance that many testers are experts at reaching. Much of what testers do can be thought of in gaming terms.

Gamification is one approach to provide a structured analysis. A familiar perspective can blind our observation, while looking at the same issue from different perspectives can highlight observations that we might otherwise miss. Analyzing human group activity by looking at game mechanics is helpful when we want to challenge our assumptions and learn more about what we are doing well or potentially missing out on.

Gamification in application design means improving user engagement by applying the mechanics of game play that people find enjoyable or even addicting. Here are some game-like concepts we can use to analyze testing [1]:

- Context and rules around game play
- Goals and desires
- Strategies and tasks
- Risks and rewards
- Skills and chance events
- Cheating and compliance

The rules and context of testing differ from team to team, but when we start out, they are basically spelled out in our roles and our job descriptions (if we have them) and taught and enforced informally on teams. Other team members make it clear what information they like and don't like for us to provide them and the activities they think we should undertake. Once we have more experience, education, and training, we start to take a more active role in the context and rules around game play for ourselves and try to match that with the expectations and styles of the team.

The goals and desires are related to what we hope to achieve. For many testers, our goal is to find important bugs. That's often what team members ask us to do: find and report bugs. Our desires are based on a need to provide value to a team and to be promoted to a job that fits our needs for money and self worth at various points in our career.

Strategies revolve around how we prioritize or how we pick different activities and skill applications to make the best use of our time. Tasks are the day-to-day activities we engage in, related to our testing work.

Risks are something we understand well from a product perspective, but what risks do we face as testers? One is that we spend too much time on low-value activities and miss important bugs that trip up our customers. Rewards can be categorized as extrinsic and intrinsic. Extrinsic rewards are external, often linked to quantitative metrics like coverage and bug counts. Intrinsic rewards are often more qualitative, looking at the value the person gets for doing the activity itself.

Skills are interesting to analyze—from those that we develop as testers, such as observation, evaluation, communica-

tion, and technology skills, to investigation, people, and questioning skills. Chance events are fascinating. In games, they help level the playing field so that skilled players can't always dominate. In testing, we often discover important issues seemingly by accident, but our skills help put us in the right place at the right time to observe a chance event.

Cheating and compliance are also fascinating. How can you cheat at testing? As James Bach and I point out in the satirical "Secrets of Faking a Test Project," [2] a test-case-management system is easy to cheat. You just periodically pass or fail tests in the system to demonstrate progress. Here is a challenge for test managers: How do we encourage compliance with organization goals and discourage cheating?

I've introduced within this article a structure that I would like you to analyze your own testing work with. What areas above can you fill in with details? What areas are lacking? Are there areas that people want to avoid? These are often the best places to start improving.

Games and gaming provide us with extrinsic (numbers, counting, and scores) and intrinsic (doing something because it's enjoyable) rewards. In testing today, we have two polarities: metrics-heavy, scripted testing and qualitative-focused efforts like exploratory testing. If we look to game mechanics, then we can look beyond polarities and focus on effective testing. Games provide wonderful lessons to help us analyze our work. If we take it a step further and apply gaming mechanics to software testing, then we can help make it more engaging, creative, productive, and fun. **{end}**

For more on the following topic go to www.StickyMinds.com/bettersoftware.

- References