

Four Tips for Technique Seeking

by Julie Gardiner

Finding a testing buddy was one of my most valuable learning experiences. During the three years we worked together, our lunchtime discussions covered a variety of testing topics including the use of formal testing techniques. It was our lifeline to improvement. These discussions allowed us to question the what, how, and why of testing in the large organization where we worked, and they helped us discover gaps in our own knowledge and experience.

In nearly every organization in which I have worked since, I have been surprised by the lack of application of formal testing techniques.

Remembering the importance of those lunchtime discussions to my growth as a testing professional, I now challenge my testers on the objective of each test they create, how they've created it, and why—creating an environment for discussion and learning.

I suggest you challenge your team and yourself to learn as many testing techniques as possible, evaluate their applicability, and apply them to your work. Here are some tips on how to get started:

Study your bugs—James Whittaker, author of a number of books on testing software, is a passionate advocate of the “study your bugs” philosophy. Look at the bugs found in production. Analyze the reasons why a bug was introduced. Discover why you missed it. Was it because of time pressures? Technique? Lack of understanding? What can you do to prevent it from happening again?

Avoid Boris Beizer's pesticide paradox—A pesticide is a chemical that is used to kill insects that would otherwise destroy crops. However, if a pesticide is overused, the insects become resistant to the pesticide and the treatment loses its effectiveness. In the same way, software becomes “immune” because we keep applying the same test cases built from the same techniques.

Improve your knowledge and experience—Know the standards. There are a number of testing standards, but you should know Software Component Testing Standard BS7925-2. It describes a testing process as well as techniques and how to measure their effectiveness. The draft version is free. Some say reading standards documents is the best cure for insomnia, but standards do have value. For example, state-transition-based testing is excellent for navigation of GUIs and Web pages, boundary-value testing finds faults around the boundaries of processing rules, decision tables are great for testing business rules, and syntax testing is fantastic for testing inputs. Each of these techniques is described in the 7925-2 standard. Make sure your team is familiar with them all.

Learn lesser-known techniques—My favorite technique is called classification trees. It's not in the 7925-2 standard, but it is great for achieving buy-in from your clients because it is a simple, visual technique that is easily understood. My second-favorite technique is all-pairs testing using orthogonal arrays. Because this technique reduces the number of combinations to a manageable level while preserving a high probability of finding defects, it is very useful when you have to test many combinations of input data—configurations, browsers, operating systems, etc.

Web sites such as James Bach's www.satisfice.com, with its secrets of software testability, and Elisabeth Hendrickson's test heuristics cheat sheet, www.qualitytree.com, help direct us in testing software without limiting our creativity. Conferences such as STAREAST, STARWEST, and EuroSTAR feature presenters sharing their experiences with these techniques. You hear the good, the bad, and perhaps the ugly stories of applying techniques. Read books such as



Lee Copeland's *A Practitioner's Guide to Software Test Design* and James Whittaker's *How to Break Software*. Both books describe a plethora of techniques and how and when to use them. A number of other books are available on this topic, but these are my favorites.

Finally, there's no substitute for on-the-job learning. Challenge each other! If you have a day's worth of experience but the right attitude, you can still provide an immediate contribution to a team. Asking why a test is being created and which technique is being used may jog experienced staff and improve the testing.

What techniques do you currently use? Which ones have you considered? Do you study the bugs you have missed? There are many techniques to be considered, so seek new ways of finding those nasty bugs. **{end}**

Julie Gardiner has more than sixteen years' experience in the IT industry. She is a regular contributor to Testing in the UK, a marker for the ISEB Practitioner qualification, a committee member for the British Computer Society's Specialist Interest Group in Software Testing, and has been EuroSTAR's England country coordinator for the past four years. Julie is a regular speaker at software testing conferences including STAREAST, STARWEST, EuroSTAR, ICSTest, and the BCS SIGIST. Julie recently joined Grove Consultants and she is a certified ScrumMaster.