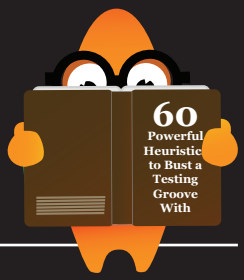
60 Powerful Heuristics to Bust a Testing Groove With

By Simon Knight

THE DOJO
Part of the Ministry of Testing family

60 Powerful Heuristics to Bust a Testing Groove With



One of the things you've probably noticed when you've been software testing for a while, and particularly when you've been testing the same product for any length of time, is that your brain starts to settle into some established ways of thinking about the software you're testing.

After a while you already know where lots of the more challenging areas of your product are, and when you begin to do some testing you make a beeline for those areas because they've already proven themselves to be the most fertile hunting grounds for juicy bugs.

But how did you get into that area of they system? What route did you follow, and what might you have missed along the way?

What if instead of making that beeline - you branched off in a completely different direction?

What might you find instead?

Getting your groove on

The brain is a funny thing. Incredibly powerful of course, but also with annoying tendencies to get stuck in ruts or grooves of comfortable thinking.

In fact - when you're carrying out your software testing, your brain is probably finding ways to think fast (lazily) rather than slowly (deliberately, analytically, creatively) much the time. And the temptation will be to let it, because, well - that's just what it does, right?

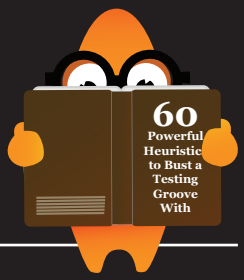
In the field of evolutionary psychology there's a growing body of evidence that serves to demonstrate how our brain can deceive us.The list of cognitive biases (deceitful shortcuts in our thinking) is a long one - but sadly, just being aware of them isn't a complete solution.

As professional software testers we have to not only be aware of our biases, we have to take control of them and constantly challenge ourselves to break out of comfortable patterns of thinking.

> *Imagine you have a piece of paper and you make marks with a pen on that surface. The surface records the marks accurately. Previous marks do not affect the way a new mark is received.*
>
> *Change the surface to a shallow dish of gelatin. You now put spoonful's of hot water on to the gelatin. The hot water dissolves the gelatin. In time, channels are formed in the surface. In this case previous information strongly affects the way new information is received. The process is no different from rain falling on a landscape. Streams are formed and then rivers. New rain is channeled along the tracks formed by preceding rain. The gelatin and landscape have allowed the hot water and rain to organise themselves into channels or sequences.*
>
> **Edward de Bono. "Think!." Random House, 2009**

Those channels, sequences or grooves that de Bono talks about are exactly what we need to break out of in order to deviate from established paths or sequences when you're testing a piece of software.

60 Powerful Heuristics to Bust a Testing Groove With



We have to break out of our biases, old or comfortable ruts - busting out of the established groove and opening up new channels for thinking about and looking at the systems and software we're paid to test.

Pattern interrupts

One way of doing this that de Bono talks about in his same book (Think!), is the use of random words, in order to steer thought away from those established ruts and patterns. His technique is actually very simple:

- You already have or identify a focal point for your thinking.
- You choose a random word.
- You use the random word as a launching point for thinking creatively about the focal point

So by way of an example - let's say the focal point of your thinking is a grooming meeting, and you want a new, more creative way of thinking about the stories being presented. At random - you choose the word "Absent".

All you need to do now is follow the new direction the word takes your thinking in. So in the case of a grooming meeting, you might start thinking about or asking questions like:

- What's missing or absent from this story? Performance requirements? Security requirements?
- What if a specific piece of data is absent or missed due to, e.g. User error?
- What if a step is missed or becomes absent?
- What if a piece of architecture becomes absent (or fails over)?

You might come up with many more ideas depending on your circumstances, product and team. But hopefully you get the idea.

The results of using this tool can be very powerful, because now instead of following a familiar route (train of thought) and arriving at the usual destination (conclusion, opinion etc.) - you're liable to end up somewhere completely different, and probably via a very different route to what you've been used to.

De Bono has a selection of words he suggests you use for this process, nouns typically - like the following: Letter, Barrier, Ear, Tooth, Bomb, Soap.

But the context he's using is slightly different. He's coming from a creativity angle. Testing is absolutely a creative discipline, but perhaps there's some other words we can use to jog our thinking instead...

Heuristics

Many folk, particularly within the Context Driven Testing community like to talk about heuristics and mnemonics. Powerful reminders that can be used as frames of reference and to steer testing efforts towards areas of risk. Often they'll come in the form of a checklist, or a mind map, or a cheatsheet.

60 Powerful Heuristics to Bust a Testing Groove With

For this technique though, we don't need any of that. Just a list of the heuristics like the one below will suffice:

1. Sequence
2. Concurrence
3. Confluence
4. Synchronisation
5. Share
6. Interaction
7. Continuity
8. Hierarchy
9. Priority
10. Dependency
11. Repetition
12. Loop
13. Parameter
14. Prerequisite
15. Configuration
16. Rule
17. Customise
18. Constraint
19. Resource
20. Access
21. Lock
22. State
23. History
24. Rollback
25. Restore
26. Refresh
27. Clone
28. Temporary
29. Trace
30. Batch
31. Void
32. Absent
33. Feedback
34. Saturate
35. Sort
36. Scale
37. Corrupt
38. Integrity
39. Invoke
40. Timing
41. Delay
42. Customers
43. Information
44. Developer
45. Team
46. Tools
47. Schedule
48. Deliverables
49. Structure
50. Functions
51. Data
52. Platform
53. Operations
54. Time
55. Capability
56. Reliability
57. Usability
58. Scalability
59. Performance
60. Compatibility

Handy Tip!
Watch the clock

For randomness you can use another technique straight out of De Bono's playbook.

The more observant among you will have noticed there are 60 words. That means you can use the second hand on your watch, or whatever other clock you have to hand. Just glance at the time, make a note of the number of seconds, and use the matching word to bust you out of your testing groove.

60 Powerful Heuristics to Bust a Testing Groove With

Whenever you feel stuck (in an established groove, rut or pattern of thinking), you can just pick one of the words from the list and use it to generate some ideas for new ways to carry out your software testing.

I've been testing a repayment calculator recently. So I'll use that as the basis for a couple of examples, just to get you started:

When I looked at the clock, I saw the second's hand was pointing at 41, Delay:

- What happens if I delay the user actions? Say I go off to grab a coffee while filling out the various stages? Does my session expire? Do I get logged out? Does the application do anything to protect sensitive information from prying eyes?

- What happens when the server is under load? Are responses delayed?

Let's try a different one. This time, I got 47, Schedule. I'm still working with the same application:

- The application gives me the option to schedule repayments. I'll explore that for a bit. Does the schedule work?

- Can I re-schedule once I've submitted a plan?

When I tried again, I got 11, repetition:

- What happens if I repeat the calculation? Do I get the same result?

- What if I keep repeating the same step? Am I able to do so? Should I be able to do so? Do I see any errors?

And so on and so on. You can use this technique whenever you need to. It's important that the selection is random though - otherwise your brain will just choose words you feel comfortable with, which is not going to achieve the desired effect of taking you off in a completely new direction.

60 Powerful Heuristics

1. Sequence
2. Concurrence
3. Confluence
4. Synchronisation
5. Share
6. Interaction
7. Continuity
8. Hierarchy
9. Priority
10. Dependency
11. Repetition
12. Loop
13. Parameter
14. Prerequisite
15. Configuration
16. Rule
17. Customise
18. Constraint
19. Resource
20. Access
21. Lock
22. State
23. History
24. Rollback
25. Restore
26. Refresh
27. Clone
28. Temporary
29. Trace
30. Batch
31. Void
32. Absent
33. Feedback
34. Saturate
35. Sort
36. Scale
37. Corrupt
38. Integrity
39. Invoke
40. Timing
41. Delay
42. Customers
43. Information
44. Developer
45. Team
46. Tools
47. Schedule
48. Deliverables
49. Structure
50. Functions
51. Data
52. Platform
53. Operations
54. Time
55. Capability
56. Reliability
57. Usability
58. Scalability
59. Performance
60. Compatibility