

Test rigs, dev machines and psychic debugging

Source:

<http://coding.derkeiler.com/Archive/Delphi/borland.public.delphi.non-technical/2008-05/msg01742.html>

- *From:* Scout <Toothpaste@xxxxxxxx>
 - *Date:* Sat, 10 May 2008 21:55:24 +1200
-

I got a cool new dev machine from my Boss as a sort of late Xmas present a few months ago.

After all the fun of installing Delphi on it, and getting all of my custom components to work to the point where I could rebuild a project successfully, I ran into a bit of a lazy block where I couldn't be bothered installing some of the additional software that it provided by other parties, and that our software relies on.

In the past, my dev machine has been capable of doing anything that a client machine would be able to do, but this new machine is much cleaner, can't do a bunch of stuff, and I am pretty happy about it.

In the worst-case scenario, I can fire up a VM that is fully specced and try to figure out problems that way. This happens rarely.

More often, I'll write a driver that emulates what the 3rd party stuff is supposed to do, and leave it up to the testing dept to do end-to-end checks.

I decided a while ago that my dev machine should not also be a test-rig, so that leaves me with only psychic debugging as a way of tracking down most bugs.

The thing that I have found most beneficial about this approach is that it is way more productive/effective than anything I did earlier.

I've found that debugging most issues ends up being an exercise in solving problems by thinking about them, in much the same way that you might try to solve a Sudoku puzzle without looking at it.

Maybe there's also an element of being able to "hold a program in your head".

Gone are the days of swearing at the IDE debugger and the way it throws focus around indiscriminately (using BDS2006 here). Instead, I can just go to bed and sleep on the problem, secure in the knowledge that in finding the bug, I might also find a good re-factoring

Test rigs, dev machines and psychic debugging

opportunity or algorithm improvement.

The systems that I work on are large and complex. There's nothing really psychic about trying to understand every part of what the system is doing simultaneously, but I do find that the restriction of not being able to single step through what it is up to has helped me enormously. It can, however, be a bit mind-bending.

The testers and customers think that I am psychic.

Do you work this way? Would/could you work this way?

Scout

.