TEST DRIVERS

Graham Oakes sings the praises of programming using test suites.

Graham Oakes: done properly, testing is a way of communicating confidence to customers

Test driven development (TDD) is gaining mindshare. The practice of defining tests at the outset and then creating code that passes them came to the fore with extreme programming (XP). Among other things, a good test suite gives extreme programmers the confidence to re-factor fearlessly, keeping their systems conceptually tidy even as they evolve rapidly.

The idea has caught on outside the XP community for a couple of reasons:

- Tests are a good way to capture unambiguous requirements. To write a good test, you need to be very clear about what you're looking for.
- Thinking about testing from the outset causes us to design testable systems and services. Such designs often turn out to be good according to many other criteria (such as modularity and cohesiveness).

By bringing testing forward in the development process, we can bring a new perspective onto some of the thorniest problems of software development and application integration.

However, some people in the testing community are raising questions about TDD. Rather than welcoming this new primacy for testing, I've heard testers question whether TDD isn't steering us into a developer-centric view of testing. Sure, the argument goes, large suites of very granular tests give us the confidence to progressively add new features to the system. Their very size, however, may fool us into thinking that they're a substitute for clear thinking about system testing.

We've all come across the subtle (and not so subtle) problems which emerge as systems are created from complex assemblages of components and services. A large number of granular tests may not be the best way to find these problems. As SOA becomes an increasingly important development paradigm, such assemblages are only going to become more common, so effective system testing will grow in importance.

I wonder if this conflict is happening because people are focusing on test driven development? After all, developing systems is only a subset of delivering them. In many places (some large banks spring to mind), it can take six months or more to take the 'completed' system through the acceptance test and service introduction processes. Any process which claims agility without addressing these phases sounds pretty hollow to me.

This leads to another question: in amongst all the discussion on technology and governance for SOA, who is addressing the vexed question of customer assurance? How do people know that this assemblage of services will work correctly?

Even if we can assemble and reconfigure systems quickly, they're of little value if the customers don't trust them enough to use them. Testing isn't just a tool to help developers: done properly, it's also a way of communicating confidence to our customers.

● *Enterprise Integration, SOA & Web Services Evaluation Centre Expert Dr Graham Oakes is the principal of content management, product development and customer service strategies consultancy Graham Oakes Ltd. Email: graham@grahamoakes.co.uk. Website: www.grahamoakes.co.uk.*