

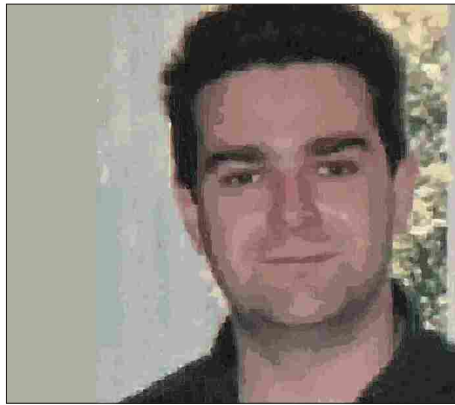
Meet the New Tester [PART 1]

This is the first article in a two-part series

How do we reduce the number of defects migrating into the system testing phase? By introducing a policy of thorough testing at the development phase, with the help of a new type of tester.



**David Evans and
Nick Pointon
of Cresta**



It is often cited that the cost of correcting errors or defects increases tenfold for each phase in the software development cycle. Defects may be introduced to a system at various stages of its development life-cycle, but in a testing regime that is heavily dependent upon a post-development testing phase, defects that are introduced at the earliest stages of the project are allowed to 'migrate' into later phases until they are finally exposed during testing, or worse, in the live system. This syndrome is known as Defect Migration

Defect migration is costly, because it can cost ten times more to isolate and remove defects for each phase that they remain in the system. Placing the onus of defect detection entirely on late-stage testing leads to greater total development cost, and significantly increases the risk of project overruns.

Whether a traditional waterfall or an iterative development cycle is used, typical software projects involve a system testing phase that follows the completion of coding. The principal purpose of this phase

is to ensure that the whole system behaves and performs correctly in an environment that anticipates (or at least closely represents) production usage.

Even if system test coverage is comprehensive and a high proportion of latent defects are found, correcting them will consume far more resources than if the same defects were trapped during development, due to the additional cost of the defect management and triage process.

System testing is a valuable activity that is meant to find the sort of defects that are not easily detectable in a development environment. So it's necessary, but costly. Test rigs and environments are complex to set up, external system interfaces must be connected or simulated, tests are generally longer to run, and in many cases expert business users are required to assess results or investigate anomalies.

Ironically, the more effective the system testing, the more difficult it is to keep to the testing schedule. The greater the number of defects encountered during system testing, the greater the triage burden and subsequent pressure to run extra test cycles to allow for retesting and regression checks.

Too often, valuable system testing time is eaten up with finding and resolving basic functional defects that could (and should) have been detected by unit testing in the development phase. Furthermore, many such functional defects effectively 'block' access to certain functions or areas of the system under test. These blocking defects will further impact the testing schedule as they may render other valid and important tests unable to be executed until the blocking defect is fixed.

Clearly, in order to avoid this double-whammy of high costs and unpredictable time-frames in the system test phase, we must improve the quality of the system under test before it reaches this phase. Defects are an unfortunate fact of life on a

development project, but their impact on the project schedule (and stakeholder confidence) can be vastly reduced if some effective 'border control' is applied to prevent them migrating.

Some of the most cost-effective methods to minimise defect migration from the development phase are:

- Introduce test-driven development: automated, programmatic unit testing by developers which applies to all new code.
- Provide automated functional regression tests to be applied to a daily software build.
- Perform directed, exploratory manual testing by an experienced tester on the software under development.

All of these methods require an awareness and respect for testing within the development phase, which unfortunately most teams are missing. That can be resolved by adding a new member to the development team responsible for ensuring the quality of testing activities in the development phase.

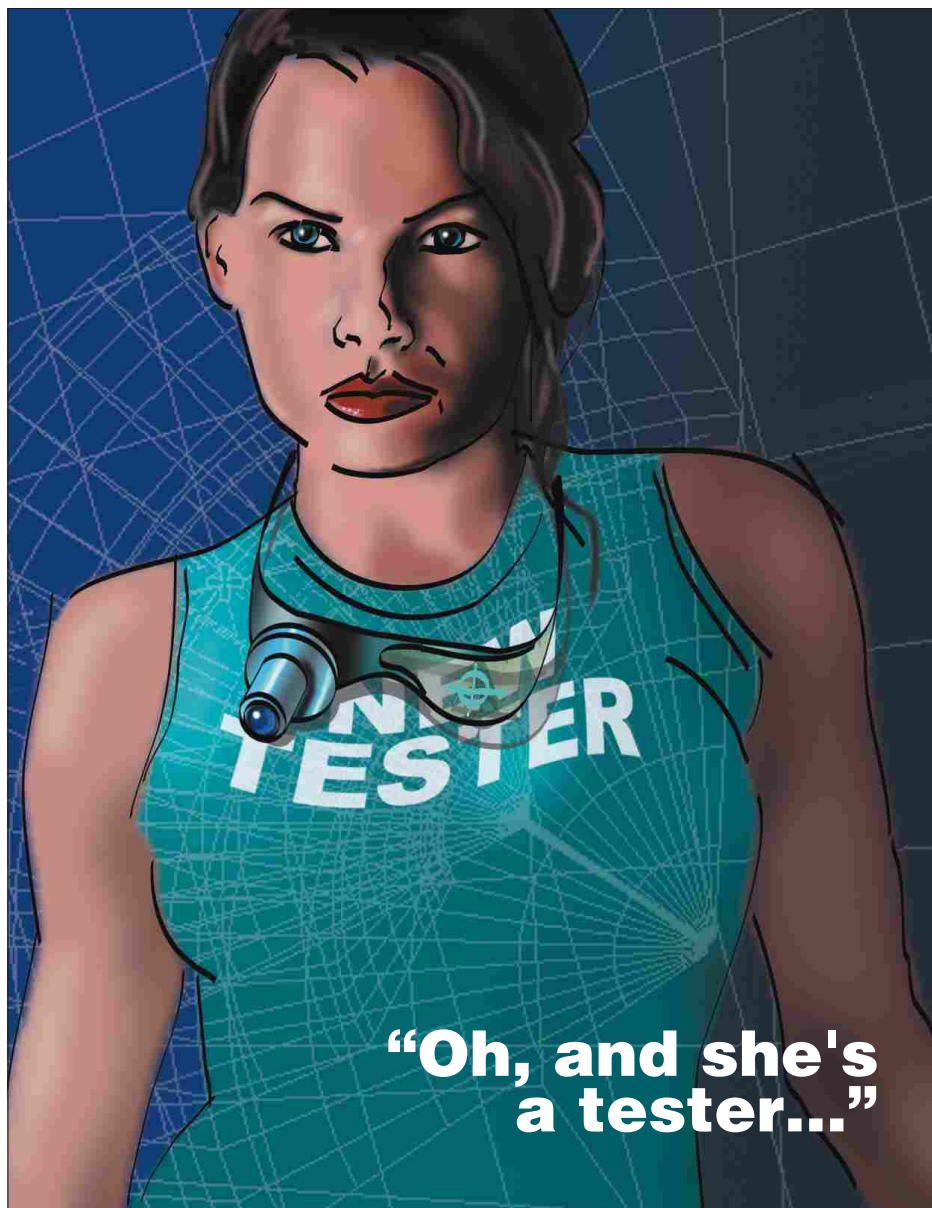
Testing in Development

There are two main types of testing that can be carried out during development:

- White box unit testing intended to confirm the programmer's expectations about the code they write.
- Black-box functional testing intended to confirm the user's expectations about the system's functionality.

Until the relatively recent advent of test-driven development, unit testing by developers was something of a forgotten art, an unpleasant chore or even a waste of developer's time. As a result, unit testing is still typically unstructured, ad-hoc and undocumented. It therefore provides low information value about quality and risk.

Unit testing and other quality control measures in the development phase are a



great opportunity to restore quality, stability and value in later testing phases. A defect discovered in development can be 'nipped in the bud' and resolved without the need for the high visibility and cost overhead of the whole defect management process.

Effective testing means any 'additional' time spent on defect correction is reflected in the development time for that task, which is precisely as it should be. The development manager can monitor any schedule slips at the task allocation level, and de-scope low priority tasks and features if necessary. This is far preferable

to proudly announcing 'completion' of all features, only to discover during testing that a whole lot of remedial work is required.

So, we have a case for better testing in the development phase. But who is going to take responsibility for this?

Enter the New Tester

"OK guys... meet the newest member of the team. She will be sitting right here with us, she'll attend all the daily stand-up meetings and she'll take part in the coffee-round too. Oh, and she's a tester..."

The new tester will be an active and co-

located member of the development team. By working alongside developers and helping the team to test code as it is written, the tester will provide immediate feedback on quality and catch defects at a stage when they can be fixed quickly and cheaply.

The hard task gaining trust and acceptance of the team means she'll need to be no ordinary tester. She'll need to be a new breed of individual who is capable of proudly carrying the banner of quality while understanding the realities of pressures on a development team. A kind of ambassador of the testing group, preparing the way for smooth negotiations later on.

This 'embedded' tester on the development team has a number of activities to carry out, but the role is essentially this: to assure the quality and functionality of the code produced, so that subsequent testing can reliably focus on

proving business processes rather than application functionality.

The primary tasks for the new tester are:

- Assisting developers with Unit Test design
- Exploratory manual testing of new features
- Providing test automation services and expertise to the development team

To be continued....

In the next edition of Professional Tester we will be looking in greater detail at the role our new tester can play when she's embedded in the development team. We'll also look at what skills and qualities she should have and the management structure required to reap the full benefits of this role.

nick.pointon@cresta.net;

david.evans@cresta.net

Subscribe to
Professional Tester Magazine

FIVE FACTS ABOUT

WEB APP TESTING

OTHER VENDORS

DON'T WANT YOU TO KNOW.



With some Web app testing vendors, the thing to ask yourself isn't "what are they selling" but "what are they hiding"?

Consider these five facts:

- FACT #1: It's hard to test dynamic, complex Web apps with a tool originally built to test client-server apps.
- FACT #2: It's even more complicated when you have to manually develop test scripts using a proprietary programming language.
- FACT #3: Developing separate scripts for functional tests, load tests, and performance monitoring is inefficient and unnecessary.
- FACT #4: You don't have to put up with restrictive licensing agreements, endless training, and expensive consultants.
- FACT #5: You can download our free evaluation and test it against your application in the same time it takes to watch their stale demo.

It's time to get the facts about Web app testing.

Get your FREE Web App Testing Fact Pack

- Call: 01344 668080
- Visit: www.empirix.com/facts2
- Email: facts@empirix.com

EMPIRIX
www.empirix.com