December, 2008

testing experience

The Magazine for Professional Testers

printed in Germany

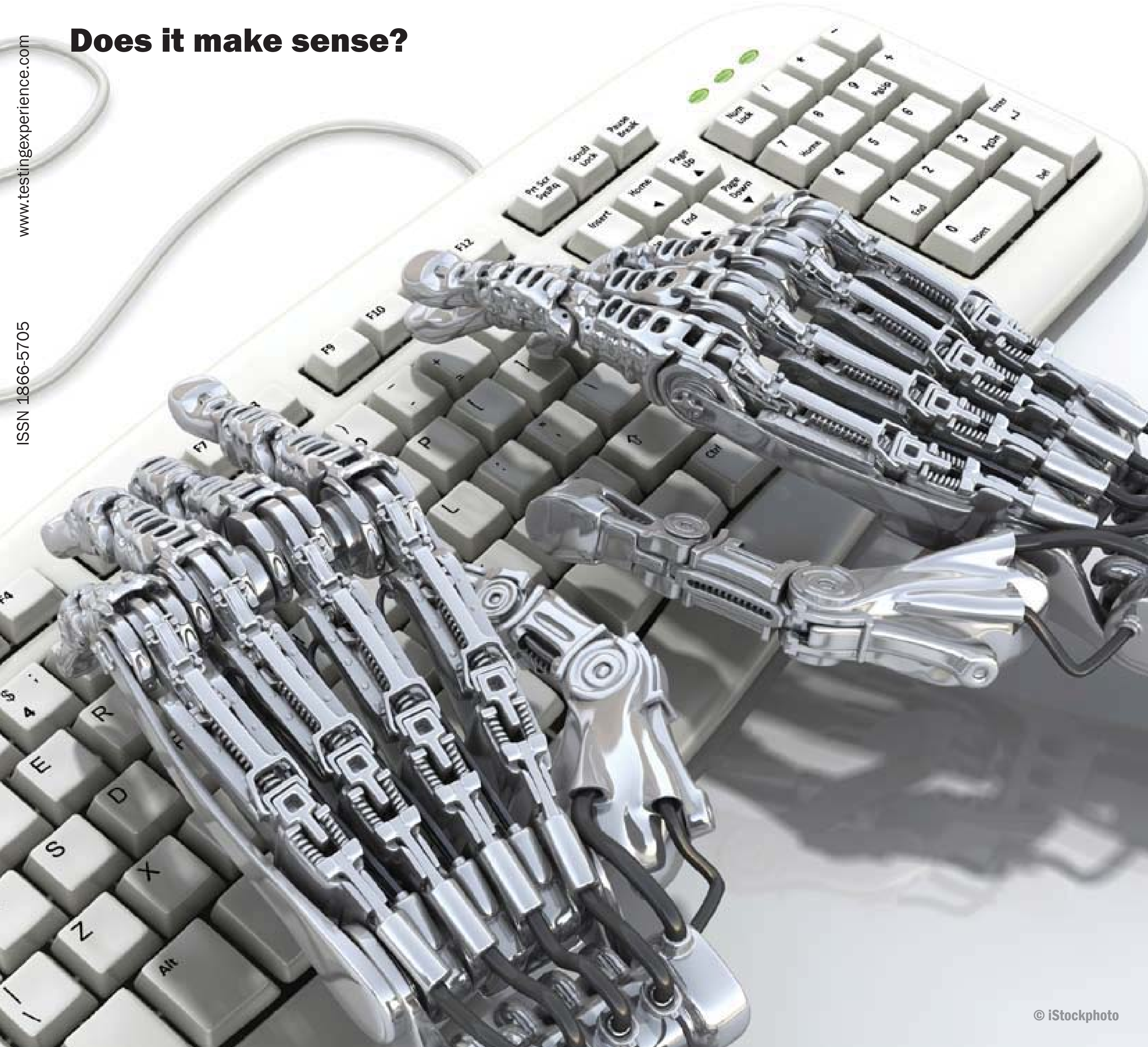
free exemplar

www.testingexperience.com

ISSN 1866-5705

Test Automation -

Does it make sense?



© iStockphoto

© iStockphoto

The Seductive and Dangerous V-Model

by James Christie

The Project Manager finishes the meeting by firing a question at the Programme Test Manager, and myself, the Project Test Manager.

"The steering board's asking questions about quality. Are we following a formal testing model?"

The Programme Test Manager doesn't hesitate.
"We always use the V Model. It's the industry standard."

The Project Manager scribbles a note. *"Good"*.

Fast forward four months, and I'm with the Programme Manager and the test analysts. It's a round table session, to boost morale and assure everyone that their hard work is appreciated.

The Programme Manager is upbeat.
"Of course you all know about the problems we've been having, but when you look at the big picture, it's not so bad. We're only about 10% behind where we should have been, and we're still on target to hit the implementation date."

This is a red rag to a bullish test manager, so I jump in.
"Yes, but that 10% is all at the expense of the testing window. We've lost half our time."

The Programme Manager doesn't take offence, and laughs.
"Oh, come on! You're not going to tell me you've not seen that before? If you're unhappy with that then maybe you're in the wrong job!".

I'm not in the mood to let it drop.

"That doesn't make it right. There's always a readiness to accept testing getting squeezed, as if the test window is contingency. If project schedules and budgets overrun, project managers can't get away with saying 'oh – that always happens' ".
He smiles and moves smoothly on, whilst the test analysts try to look deadpan, and no doubt some decide that career progression to test management isn't quite as attractive as they used to think.

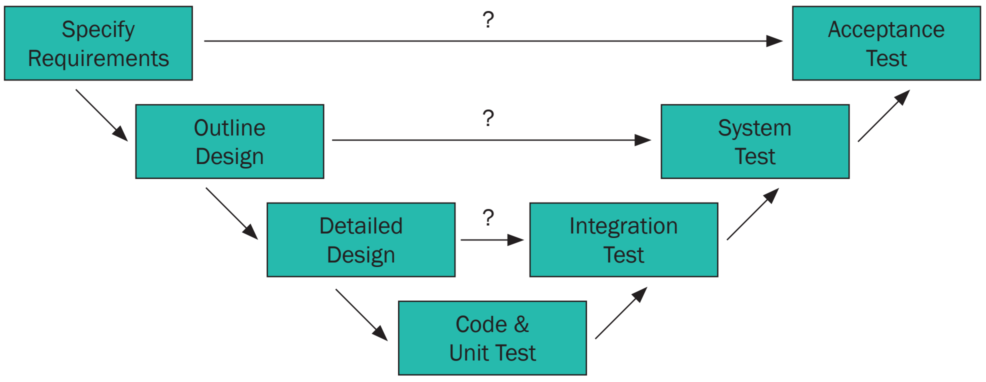
Test windows do get squeezed

The Programme Manager was quite right. That's what happens on Waterfall projects, and saying that you're using the V Model does nothing to stop that almost routine squeeze.
In "testing experience" issue 2, Dieter Arnouts [1] outlined how test management fits into different software development lifecycle models. I want to expand on the V model, its problems and what testers can do in response.
In the earlier meeting the Programme Test Manager had given an honest and truthful answer, but I wonder if he was actually wholly misleading. Yes, we were using the V Model, and unquestionably it would be the "test model" of choice for most testing professionals, in the UK at least.
However, I question whether it truly qualifies as a "test model", and whether its status as best practice, or industry standard, is deserved.
A useful model has to represent the way that testing can and should be carried out. A model must therefore be coherent, reasonably precisely defined and grounded in the realities of software development.

Coherence at the expense of precision

If the V model, as practised in the UK, has coherence it's at the expense of precision. Worldwide there are several different versions and interpretations. At one extreme is the German V-Model, [2], the official project management methodology of the German government. It's equivalent to Prince-2, but more directly relevant to software development. Unquestionably this is coherent and rigorously defined.
Of course, it's not the V Model at all in the sense that UK testers understand it. For one thing the V stands not for the familiar V shaped lifecycle model, but for "vorgehens", German for "going forwards". However, this model does promote a V shaped lifecycle, and some seem to think that this is the real V Model, in its pure form.
The US also has a government standard V Model [3], which dates back about 20 years, like its German counterpart. Its scope is narrower, being a systems development lifecycle model, but still more far more detailed and rigorous than most UK practitioners would understand by the V Model.
The understanding that most of us in the UK have of the V Model is probably based on the V Model as it's taught in the ISEB Foundation Certificate in Software Testing [4]. The syllabus merely describes the Model without providing an illustration, and does not even name the development levels on the left hand side of the V. Nor does it prescribe a set number of levels of testing. Four levels is "common", but there could be more or less on any project.
This makes sense to experienced practitioners. It's certainly coherent, in that it is intuitive. It is easy for testers to understand, but it's far from precise. Novices have no trouble understanding the model to the level required for a multiple choice exam, but they can struggle to get to grips with what exactly the V Model is.
If they search on the internet their confusion will deepen. Wikipedia is hopelessly confused, with two separate articles on the V Model. These fail to make a clear distinction between the German model [5] and the descriptive life-

cycle model [6] familiar to testers. Unsurprisingly there are many queries on internet forums asking "what ***is*** the V Model?".
There are so many variations that in practice the V Model can mean just about whatever you want it to mean.



It is interesting to do a Google images search on "V Model". One can see a huge range of images illustrating the model. All share the V shape, and all show an arrow or line linking equivalent stages in each leg. However, the nature of the link is vague and inconsistent. Is it static testing of deliverables? Simple review and sign-off of specifications? Early test planning? Even the direction of the arrow varies. The Wikipedia article on the German V Model has the arrows going in different directions in different diagrams. There is no explanation for this. It simply reflects the confusion in the profession about what exactly the V Model is.
Boiled down to its most basic form, the V Model can mean simply that test planning should start early, and test plans should be based on the equivalent level of the left hand development leg. In practice, I fear that that is all it usually does mean.
The trouble is that this really isn't a worthwhile advance on the Waterfall Model. The V Model is no more than a testing variant of the Waterfall. At best, it only mitigates some of the damaging impact that the Waterfall has on testing and quality.

Why the Waterfall is bad for testing and quality

In 1970 Royce wrote the famous paper [7] depicting the Waterfall Model. It is a strange example of how perceptive and practical advice can be distorted and misapplied. Using the analogy of a waterfall, Royce set up a straw man model to describe how software developments had been managed in the 1960s. He then demolished the model. Unfortunately posterity has credited him with inventing the model!
The problems with the Waterfall, as seen by Royce, are its inability to handle changes, mainly changes forced by test defects, and the inflexibility of the model in dealing with iteration. The implicit assumption of the Waterfall that any iteration can be safely restricted to successive stages doesn't hold. Such reworking could extend right back to the early stages of design, and is not a matter of polishing up the later steps of the previous phase.
Although Royce did not dwell on the difficulty of establishing the requirements accurately at the first pass, this problem did trouble other, later writers. They concluded that not only is it impossible in practice to define the requirements accurately before construction starts, it is wrong in principle because the process of developing the application changes the users' understanding of what is desirable and possible and thus changes the requirements. Furthermore, there is widespread agreement that the Waterfall is particularly damaging for interactive applications.
The problems of the Waterfall are therefore not because the analysts and developers have screwed up. The problems are inherent in the model. They are the result of following it properly, not badly!

The Waterfall and project management

So if the Waterfall was originally depicted as a straw man in 1970, and if it has been consistently savaged by academics and expert practitioners, why is there still debate about it? The answer lies in project management. The Waterfall was shaped by project management requirements, and it therefore facilitates project management. Along with the V model it is neater, and much easier to manage, plan and control than an iterative approach, which looks messy and unpredictable to the project manager. The Waterfall allows project managers to structure projects neatly, and it is good for ***projects not applications***!
Defenders of the model often argue that in practice the Waterfall is not applied in its pure form; that there's always some limited iteration and that there is invariably some degree of overlap between the phases. They argue that critics don't understand how it can be used effectively, and that changes are time-consuming and expensive regardless of the model followed. They dismiss the criticism that the Waterfall doesn't allow a return to an earlier stage by saying that this would be down to individual project managers being too inflexible, rather than a problem with the model.
This is naive. The problem is not solved by better project management, because project management itself has contributed towards the problem; or rather the response of rational project managers to the pressures facing them.
The symbiosis between project management and the Waterfall has meant that practitioners have frequently been compelled to employ methods that they knew were ineffective. This is most graphically illustrated by the UK government's mandated use of the PRINCE2 project management method and SSADM development methodology. These two go hand in hand.
They are not necessarily flawed, and this article does not have room for their merits and problems, but they ***are*** associated with a traditional approach such as the Waterfall.
The UK's National Audit Office stated in 2003 [8] that "*PRINCE ... fits particularly well with the 'waterfall' approach*", and that "the waterfall ... remains the preferred a*pproach for developing systems where it is very important to get the specification exactly right*".
This is current advice. The public sector tends to be more risk averse than the private sector. If auditors say an approach is "*preferred*" then it would take a bold and confident project manager to reject that advice.
This official advice is offered in spite of persistent criticism that it's never possible to define the requirements precisely in advance in the style assumed by the Waterfall, and that attempting to do so is possible only if one is prepared to steamroller the users into accepting a system that doesn't satisfy their goals.
The UK government is therefore promoting the use of a project management method partly because it fits well with a lifecycle that is fundamentally flawed because it has been shaped by project management needs rather than those of software development.
The history of the Waterfall in the USA illustrates its durability and provides a further insight into why it will survive for some time to come; its neat fit with commercial procurement practices.
The US military was the main customer for large-scale software development contracts in the 1970s and insisted on formal and rigorous development methods which effectively ruled out any alternative to the Waterfall. The reasons for this were quite explicitly to help the DoD to keep control of procurement. This bias in favour of the Waterfall lasted into the 90s, by which time the Waterfall was embedded in the very soul of the IT profession.
Even now traditional procurement practices, which fit much more comfortably with the Waterfall and the V Model, are being followed throughout the world because they facilitate control, not quality. Hence the frequent insistence on mind-numbing amounts of documentation in order to obtain approval to move on to the next stage, and the funding that goes with it. The danger is that supplier staff become fixated on the production of the material that pays their wages, rather than the real substance of the project.
It is surely significant that students of the Association of Chartered Certified Accountants are taught that the V Model is an excellent basis for planning testing. This is the only testing model that their syllabus mentions. It is the model for accountants and project managers, not developers or testers.

V for veneer?

What is seductive and damaging about the V model is that it gives credibility to the Waterfall. It gives a veneer of respectability to a process that almost guarantees shoddy quality. The most damaging aspect is perhaps the effect on usability. The V model effectively discourages user involvement in evaluating the design, and especially the interface, before the formal user acceptance testing stage. By then it is too late to make significant changes to the design. Usability problems can be dismissed as "cosmetic" and the users are pressured to accept a system that doesn't meet their needs. This is bad if it is an application for internal corporate users. It is potentially disastrous if it is a web application for customers.

None of this is new to academics or test consultants who've had to deal with usability problems. However, as Rex Black commented in his article on ISTQB Certification in issue 1 of "testing experience"; "*Common practices lag best practices by around 30 years*" [9].

Black provided a fine metaphor for this quality problem in 2002 [10]. After correctly identifying that V Model projects are driven by cost and schedule constraints, rather than quality, Black argues that the resultant fixing of the implementation date effectively locks the top of the right leg of the V in place, while the pivot point at the bottom slips further to the right, thus creating Black's "ski slope and cliff".

The project glides down the ski slope, then crashes into the "quality cliff" of the test execution stages that have been compressed into an impossible timescale.

The Waterfall may have resulted in bad systems, but its massive saving grace for companies and governments alike was that they were developed in projects that offered at least the illusion of being manageable! The Waterfall will therefore survive for a long time to come.

The V Model's great attractions were that it fitted beautifully into the structure of the Waterfall, it didn't challenge that model, and it just looks right; comfortable and reassuring.

What can testers do to limit the damage?

I believe strongly that iterative development techniques must be used wherever possible. However, such techniques are beyond the scope of this article. Here I am interested only in explaining why the V Model is defective, why it has such a grip on our profession, and what testers can do to limit its potential damage.

The key question is therefore; how can we improve matters when we find we have to use it? As so often in life just asking the question is half the battle. It's crucial that testers shift their mindset from an assumption that the V Model will guide them through to a successful implementation, and instead regard it as a flawed model with a succession of mantraps that must be avoided.

Testers must first accept the counter-intuitive truth that the Waterfall and V Model only work when their precepts are violated. This won't come as any great surprise to experienced practitioners, though it is a tough lesson for novices to learn. Developers and testers may follow models and development lifecyles in theory, but often it's no more than lip service. When it comes to the crunch we all do whatever works and ditch the theory. So why not adopt techniques that work and stop pretending that the V Model does?

In particular, iteration happens anyway! Embrace it. The choice is between planning for iteration and frantically winging it, trying to squeeze in fixes and reworking of specifications.

Even the most rigid Waterfall project would allow some iteration during test execution. Try to persuade project management that test execution provides feedback about quality and risks and that it cannot be right to defer feedback. Where it is possible to get feedback early it must be taken. That means feedback to analysts and designers early enough to let them fix problems quickly and cheaply. This feedback and correction effectively introduces iteration. Acknowledging this allows us to plan for it.

Defenders of the V Model argue that that is entirely consistent with the V Model. Indeed it is. That's the point. However, what the V Model doesn't do adequately is help testers to force the issue; to provide a statement of sound practice, an effective, practical model that will guide them to a happy conclusion. It is just too vague and wishy washy.

A fundamental flaw of the V model is that it is not hooked into the construction stages in the way that its proponents blithely assume. If testing does not start early then the V Model is worthless. It is therefore important that testers agitate for the adoption of a model that honours the good intentions of the V Model, but is better suited to the realties of development and the needs of the testers.

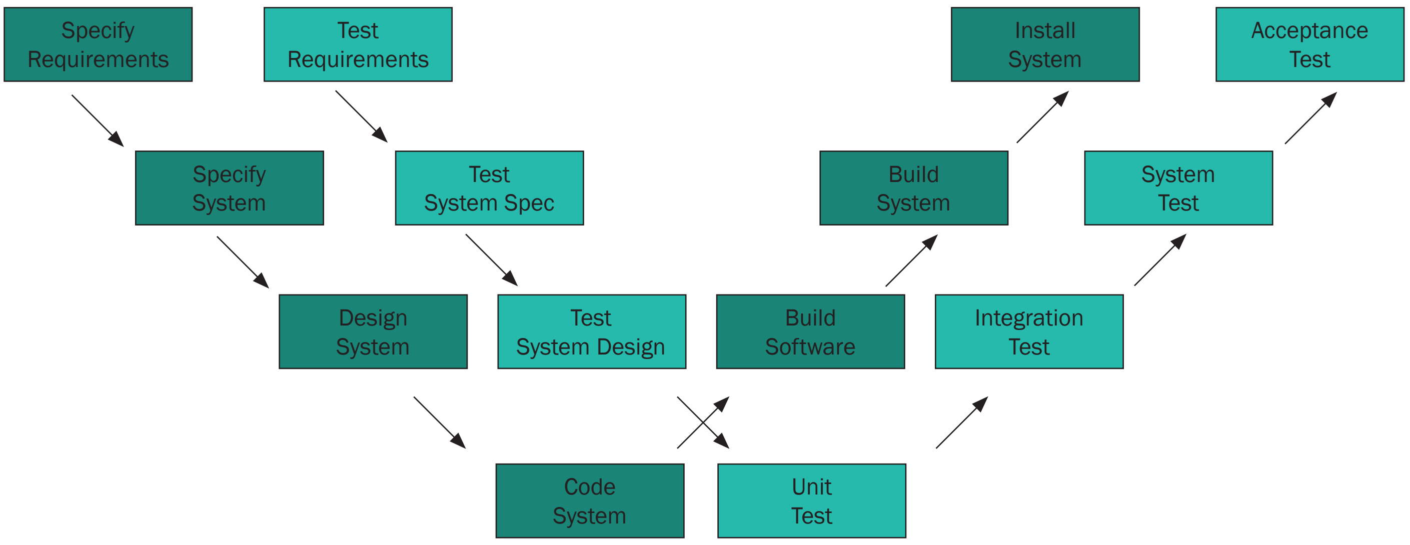
Herzlich's W Model

An interesting extension of the V Model is Paul Herzlich's W Model [11].

The W Model removes the vague and ambiguous lines linking the left and right legs of the V and replaces them with parallel testing activities, shadowing each of the development activities. As the project moves down the left leg, the testers carry out static testing (i.e. inspections and walkthroughs) of the deliverables at each stage. Ideally prototyping and early usability testing would be included to test the design of interactive systems at a time when it would be easy to solve problems. The emphasis would then switch to dynamic testing once the project moves into the integration leg.

There are several interesting aspects to the W Model. Firstly, it drops the arbitrary and unrealistic assumption that there should be a testing stage in the right leg for each development stage in the left leg. Each of the development stages has its testing shadow, within the same leg. The illustration shows a typical example where there are the same number of stages in each leg, but it's possible to vary the number and the nature of the testing stages as circumstances require without violating the principles of the model.

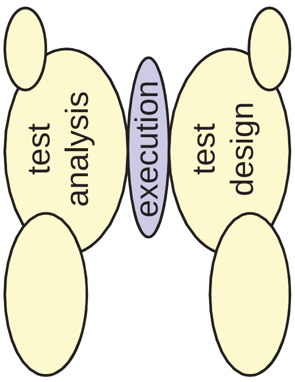
Also, it explicitly does not require the test plan for each dynamic test stage to be based on the specification produced in the twin stage on the left hand side. There is no twin stage of course, but this does address one of the undesirable by-products of a common but unthink-



ing adoption of the V Model; a blind insistence that test plans should be generated from these equivalent documents, and only from those documents.

A crucial advantage of the W Model is that it encourages testers to define tests that can be built into the project plan, and on which development activity will be dependent, thus making it harder for test execution to be squeezed at the end of the project.

However, starting formal test execution in parallel with the start of development must not mean token reviews and sign-offs of the documentation at the end of each stage. Commonly under the V Model, and the Waterfall, test managers receive specifications with the request to review and sign off within a few days what the developers hope is a completed document. In such circumstances test managers who detect flaws can be seen as obstructive rather than constructive. Such last minute "reviews" do not count as early testing.

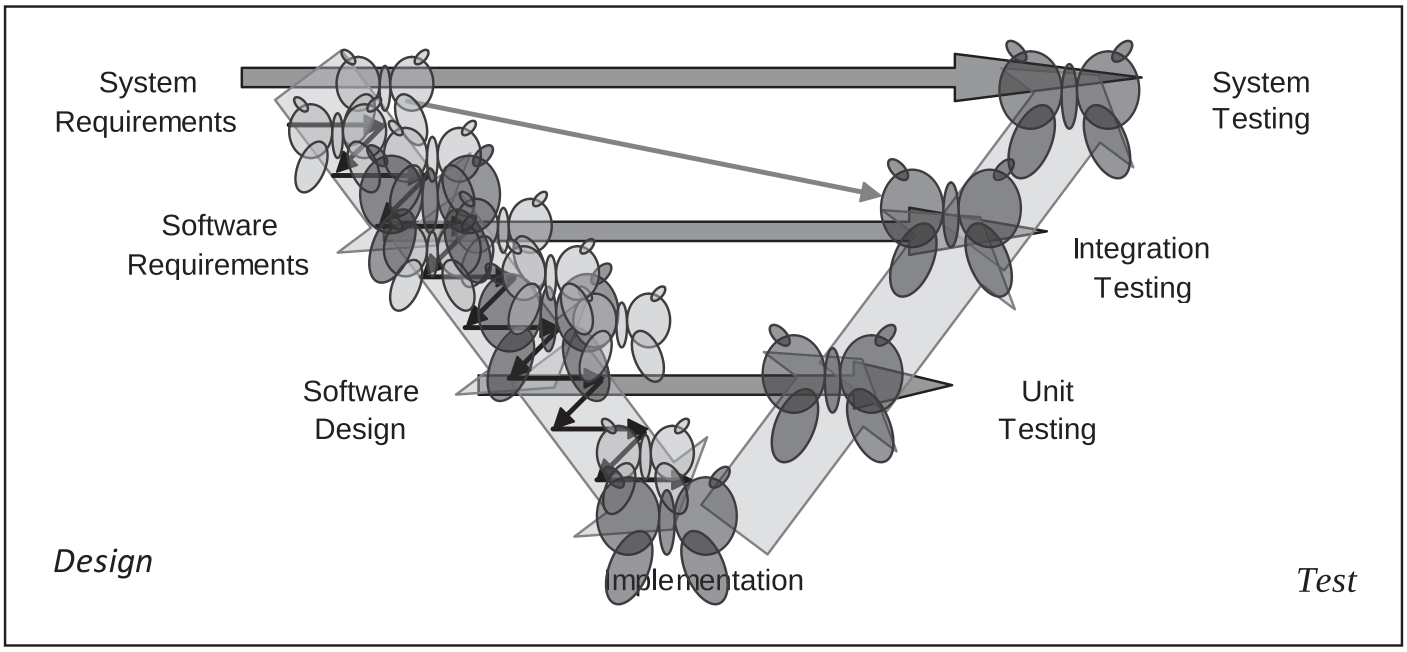


Morton's Butterfly Model

Another variation on the V Model is the little known Butterfly Model [12] by Stephen Morton, which shares many features of the W Model.

The butterfly metaphor is based on the idea that clusters of testing are required throughout the development to tease out information about the requirements, design and build. These micro-iterations can be structured into the familiar testing stages, and the early static testing stages envisaged by the W Model.

In this model these micro-iterations explicitly shape the development during the progression down the development leg. In essence, each micro-iteration can be represented by a butterfly; the left wing for test analysis, the right wing for specification and design, and the body is test execution, the muscle that links and drives the test, which might consist of more than one piece of analysis and design, hence the segmented wings. Sadly, this model does not seem to have been fully fleshed out, and in spite of its attractions it has almost vanished from sight.



Conclusion - the role of education

The beauty of the W and Butterfly Models is that they fully recognise the flaws of the V Model, but they can be overlaid on the V. That allows imaginative test managers to smuggle a more helpful and relevant testing model into projects committed to the V Model without giving the impression that they are doing anything radical or disruptive.

The V Model is so vague that a test manager could argue with a straight face that the essential features of the W or Butterfly are actually features of the V Model as the test manager believes it must be applied in practice. I would regard this as constructive diplomacy rather than spinning a line!

I present the W and Butterfly Models as interesting possibilities, but what really matters is that test managers understand the need to force planned iteration into the project schedule, and to hook testing activities into the project plan so that "testing early" becomes meaningful rather than a comforting and irrelevant platitude. It's possible for test managers to do any of this provided they understand the flaws of the V Model and how to improve matters. This takes us onto the matter of education.

The V model was the only "model" testers learned about when they took the old ISEB Foundation Certificate. Relatively few testers went on to take the Practitioner Certificate, which gave the more inquisitive students at least the chance to learn about other models. The Practitioner Certificate was difficult to pass and took too much time and money. For most testers it was not a high priority because there was no pressing need to pass it in order to obtain good jobs and contracts.

As a result of this, and the pressures of project management and procurement, the V model is unchallenged as the state of the art for testing in the minds of many testers, project managers and business managers.

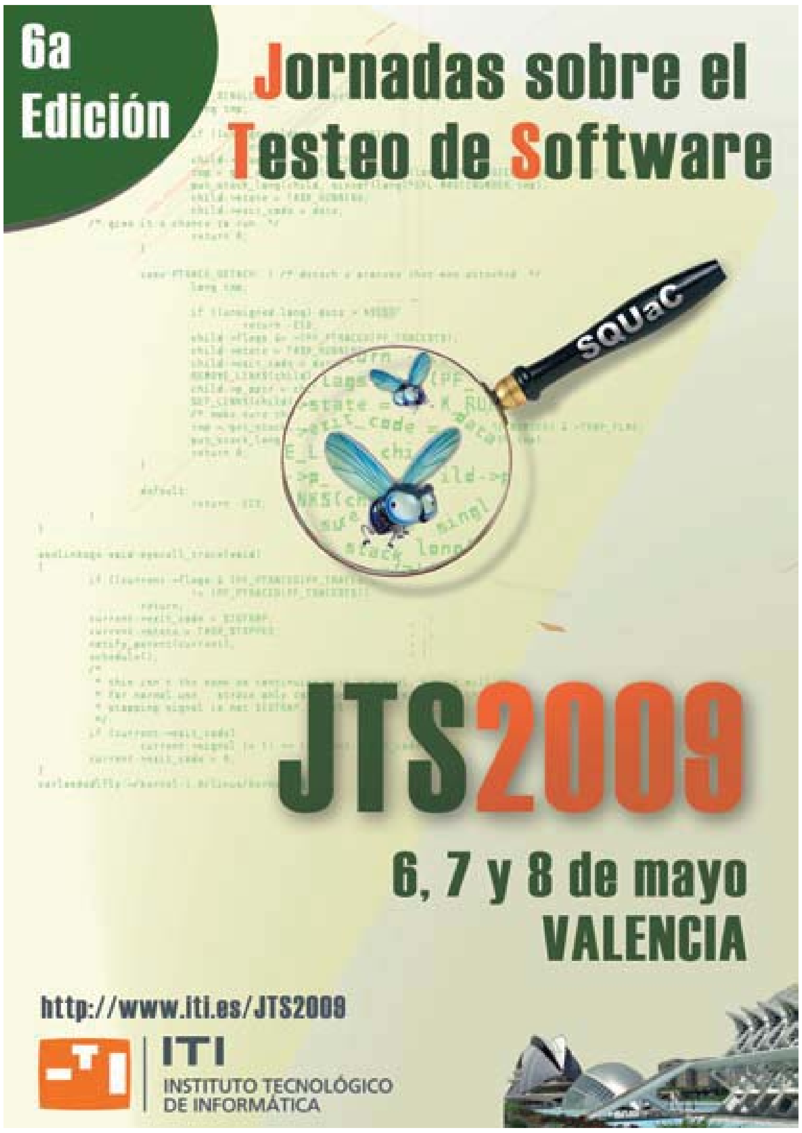
Perhaps the introduction of the new ISEB Intermediate Certificate in Software Testing will encourage more testers to think about the different effects that different development lifecycle models have on testing. The intermediate qualification will make it easier and more attractive for testers to aim for the new Practitioner Certificate in Test Management which will introduce them to test process improvement, and possibly encourage them to find out about more sophisticated testing models like TMap.

The V model will not disappear just because practitioners become aware of its problems. However, a keen understanding of its limitations will give them a chance to anticipate these problems and produce higher quality applications.

Testers must seek to improve their knowledge and skills through formal qualifications. However, they also need to be aware of the work of the testing experts such as Rex Black, Cem Kamer, Paul Gerrard, Erik van Veenendaal and Brian Marick. Go and look for their work. And read this magazine of course!

References

1. Arnouts, D. (2008). "*Test management in different Software development life cycle models*". "Testing Experience" magazine, issue 2, June 2008.
2. IABG. "*Das V-Modell*". http://v-modell.iabg.de/index.php?option=com_docman&task=cat_view&Itemid=&gid=15&orderby=dmdate_published&ascdesc=DESC . Accessed 9th August 2008. This is in German, but there are links to English pages and a downloadable version of the full documentation in English.
3. US Dept of Transportation, Federal Highway Administration. "*Systems Engineering Guidebook for ITS*" . http://www.fhwa.dot.gov/cadiv/segb/index.htm . Accessed 9th August 2008.
4. BCS ISEB Foundation Certificate in Software Testing – Syllabus. http://www.bcs.org/server.php?show=nav.7181 . Accessed 9th August 2008.
5. Wikipedia. "*V-Model*". http://en.wikipedia.org/wiki/V-Modell . Accessed 9th August 2008.
6. Wikipedia. "*V-Model (software development)*". http://en.wikipedia.org/wiki/V-Model_(software_development) . Accessed 9th August 2008.
7. Royce, W. (1970). "*Managing the Development of Large Software Systems*", IEEE Wescon, August 1970.
8. National Audit Office. (2003)."*Review of System Development – Overview*". www.nao.org.uk/intosai/edp/Audit%20Guides/Review%20of%20systems%20development%20-%20overview.pdf . Accessed 26th July 2008.
9. Black, R. (2008). "*ISTQB Certification: Why You Need It and How to Get It*" . "Testing Experience" magazine, issue 1, March 2008.
10. Black, R. (2002). "*Managing the Testing Process*", p415. Wiley 2002 .
11. Herzlich, P. (1993). "*The Politics of Testing*". Proceedings of 1st EuroSTAR conference, London, Oct. 25-28, 1993.
12. Morton, S. (2001). "*The Butterfly Model for Test Development*". Sticky Minds website. http://stickyminds.com/getfile.asp?ot=XML&id=2618&fn=XUS441382file1%2Edoc . Accessed 9th August 2008.



6a Edición
Jornadas sobre el Testeo de Software
SQUaC
JTS2009
6, 7 y 8 de mayo
VALENCIA
http://www.iti.es/JTS2009
ITI
INSTITUTO TECNOLÓGICO DE INFORMÁTICA

Biography

James lives in Perth in Scotland . He is currently working as a consultant through his own company, Claro Testing Ltd (www.clarotesting.com).
James has 24 years commercial IT experience, mainly in financial services, with the General Accident insurance company and IBM (working with a range of blue-chip clients) throughout the UK and also in Finland.
His experience covers test management (the full life-cycle from setting the strategy, writing the test plans, supervising execution, through to implementation), information security management, project management, IT audit and systems analysis.
In 2007 he successfully completed an MSc in IT. His specialism was "Integrating usability testing with formal software testing models". He is particularly interested in the improvement of testing processes and how the quality of applications can be improved by incorporating usability engineering and testing techniques.
James is a Chartered IT Professional and a professional member of the British Computer Society (MBCS). He holds the ISEB Practitioner Certificate in Software Testing and is a member of the Usability Professionals Association.