September 2005 $9.95 www.StickyMinds.com

BETTER SOFTWARE

REQUIREMENTS MATH
How do your job skills add up?
PAGE 52

EXTREME PMO?
Yes, it is possible
PAGE 22

The Print Companion to StickyMinds.com

REQUIREMENTS

WHAT GOES UP MUST COME DOWN
A Balanced Approach to Writing Requirements
PAGE 26

IF THE SHOE DOESN'T FIT...
Agile Requirements for Stepsister Projects
PAGE 34

BRING YOUR REQUIREMENTS DISCUSSIONS
Down to Earth
PAGE 40

Staying on the Critical Path

by Michael Bolton

In my last column, I wrote about the importance of critical thinking. Critical thinking skills are central to James Bach's philosophy of Rapid Software Testing. In our courses, presentations, and conversations, people often ask if critical thinking skills can be taught. I'm pretty sure that they can, but I'm almost positive that they can be inspired, practiced, and cultivated. Mind you, incuriosity can be taught easily too; that happens whenever a teacher or manager provides a "right" answer, deems the matter closed, and permits no further discussion—or worse, no further thought. I think a good start to promoting critical thinking skills is to encourage people to ask questions. That's important for testing, too; when we stop asking questions, we build tunnels for our vision, we ignore context, and we start to believe in "best practices" as though there were such things.

James developed the Heuristic Test Strategy Model as a way to ask questions that help testers quickly and effectively frame their approach for testing the product. In this installment, we will look at the *project environment*, the first of four components of the model. The model identifies eight categories that we can ask about to obtain useful information. Rapid Testers love lists, and we love mnemonics to help us remember those lists on the spot. So think "CIDTESTD" (Mother Approved!) as you ask questions.

...when we stop asking questions, we build tunnels for our vision, we ignore context, and we start to believe in "best practices" as though there were such things.

Customer. Who are the customers of the product? As I mentioned last time, it's a good idea to think expansively about this. Consider not only those who use the software but also the people who are their customers. Remember that someone other than the end-user might purchase the product. Another important question related to customers—who are the customers of the *testing effort*? To whom do we report? Who else might be interested in our work? Do we have access to all of these people?

Information. What are the sources of information about the product and the domain in which it will operate? Do testers already have background information—knowledge and inferences and implied specifications from which to derive test ideas? Do we have access to project planning documents, and if so, in what kind of shape are they? Who on the project could provide us with useful information?

Developer Relations. Who is responsible for the code? Where are they—in one location or all over the place? How do they get along with the rest of the project community? With each other?

Test Team. With whom are we working? Do we know yet? Is there a diversity of skills in place? Is someone going to have to do special research or training, or will we need to recruit outside help?

Equipment and Tools. Is there a test lab set up, or do we have to make do with our own computers? Are there separate development, testing, and production environments available? Does our testing require peripherals or special hardware of some kind? Are our favorite testing tools available now, or are we going to have to collect and install them?

Schedule. When can we expect code to test and documentation to read? What hard deadlines or milestones are in place? What is the rate of change on the project? How much time do we have for test planning, preparation, and execution? What do we have to do *right now*?

Test Items. What is the product that we're testing? Do we have access to it? Is it relatively stable, or is it changing a lot? Will changes require new tests, or will we focus on retesting? What hooks or features exist that might make the product easier to test? And what is the future direction of the product?

Deliverables. What do our clients want to see from the test process? How are we going to record and report progress? How formalized is the reporting process, and to whom do we report? Are there standards that we could—or must—follow? We've found it helpful to ask of each document: Is it a *tool* that supports our work or a *product* that we are required to show to others? Don't work too hard preparing documents that people aren't going to read or use; focus on the important stuff.

A more detailed and specific list, including questions and subcategories, is available in the document on James Bach's Web site www.satisfice.com. If you have categories or questions that you'd like to add to the list, by all means do so, especially if it's a question that you find yourself asking on several projects. But the general process is the main thing: With practice—and by carrying the mnemonic "CIDTESTD" (Mother Approved!) around in your head—you can gather a remarkable amount of important information about any project in a very short time.

As a two-sided critical thinking exercise, try this alone, or with a colleague or two:

Use the list to get three minutes' worth of new observations about your product's context.

If you're working on your own, pause for a few minutes, then revisit the categories trying to think of alternatives to your answers. If you're working with colleagues, compare notes and then brainstorm alternatives together. Don't stop until you've come up with at least three new observations in each category.

Then try to imagine aspects of the project environment that the "CIDTESTD" model itself misses, aspects that would change your approach to testing. (And if you come up with something, please let us know so that we can start a conversation about it.)

As an example, at one point, I suggested to James that "budget" might be an important addition to the model and asked why it was missing. He welcomed the question and noted that budget had been on the list for several years. He said that he had dropped it when he found that, for him, budgetary issues were covered by other questions raised by the model and that asking further questions about it didn't help him to plan or execute tests more quickly or more effectively. He then asked me if I could come up with some circumstance in which asking questions about the budget might change my test strategy in some meaningful way that wasn't already covered by the other questions. Can you come up with such a circumstance?

The matter isn't yet resolved. It may be the case that I develop a model different from his, and that would be fine. Either way, neither a good model nor a good critical thinking exercise is necessarily conclusive. Its outcome might be to prompt new questions or to cause us to consider new information that could prompt us to change our assumptions.

I believe that good testing requires us to do that constantly. As testers, we don't want to be fooled by an application that seems to be working correctly, and we don't want to be fooled into thinking that we're doing a good job. To be intellectually honest, I believe we must welcome criticism of our own work and thinking, and must be self-critical. We should be prepared to accept the strongest challenges to our beliefs so that we can continuously refine our tools, our practices, and our approaches. By questioning our project, our mission, our models, and our strategies, we reduce the chance that we will be surprised by a disastrous bug. **{end}**

Michael Bolton lives in Toronto and teaches James Bach's Rapid Software Testing course all over the world. Contact Michael at mb@developsense.com.

Don't Stop Now!

Log on to **StickyMinds.com** and join Michael Bolton and your peers in a conversation about this topic. At the end of the digital column, add your views or just read what others have to say.

LogiGear. TestArchitect™ 2

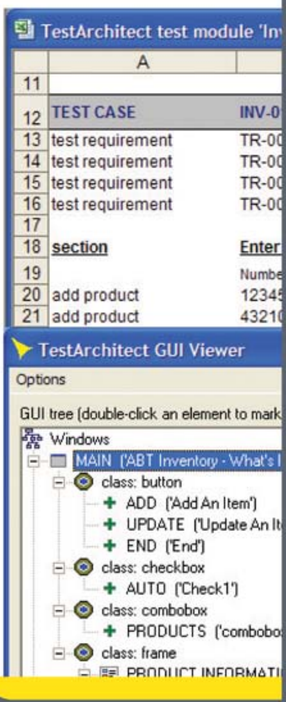
Test earlier. Test faster.
Automate smarter

Global management
- shared repository
- distributed teams

Automation that is
- maintainable,
- scaleable and
- truly reusable.

Contact us for free downloads of
- evaluation copy
- whitepaper

*Logi*Gear® Corp.
Tel 1 800 322 0333
Fax 1 650 572 2822
sales@logigear.com
www.logigear.com


TestArchitect test module 'Inv

	A	
11		
12	TEST CASE	INV-0
13	test requirement	TR-00
14	test requirement	TR-00
15	test requirement	TR-00
16	test requirement	TR-00
17		
18	section	Enter
19		Numbe
20	add product	12345
21	add product	43210

TestArchitect GUI Viewer
Options
GUI tree (double-click an element to mark
Windows
MAIN ('ABT Inventory - What's I
class: button
ADD ('Add An Item')
UPDATE ('Update An It
END ('End')
class: checkbox
AUTO ('Check1')
class: combobox
PRODUCTS ('combobo
class: frame
PRODUCT INFORMATI