Editors: Dave Thomas and Andy Hunt ■ The Pragmatic Programmers
dave@pragmaticprogrammer.com ■ andy@pragmaticprogrammer.com

Practice

Dave Thomas and Andy Hunt

Andy and I have never been completely comfortable with the name of this column—we don't really believe in the concept of "construction" as being a separate activity in the development process. Having something called "construction" makes it easy for some in the industry to Balkanize the concept—"Oh, that's just a construction detail." Even the name con-

struction has connotations that make it seem somewhat manual compared to the more cerebral "analysis" and "design" phases.

Let's use a construction metaphor to see why that's dangerous.

Building houses?

For "historical reasons," Dave lives in a Dallas suburb, one of those communities where new houses spring up like watercress on a Chia Pet. Somewhere behind each of these houses is a set of plans, and behind the plans is an architect. However, it's a fairly safe bet that none of these architects have come out to the neighborhoods they designed and hefted a brick or eyed a joist during construction. Why? Because they didn't need to. The properties of these materials are well understood and documented. If the architect needs to know the maximum span of a floor member, there's a convenient table with all the information he or she needs. The average household brick holds few surprises. The details of construction have become abstract—the designer no longer needs to deal with the physical materials.

But ask an architect to design a building using some radically new material, and the situation changes. They'll ask for samples, so they can hold it and get a feel for it. They'll ask engineers to determine the properties and might even conduct tests to see how the material behaves in real life. And during construction, they'll visit the site, watching how well theory maps into practice. They'll talk with the workers, examine the construction, and make many, many minor adjustments as small issues come to light. They'll be involved.

Many IT folks wish that software development were like building suburbs in Dallas. They wish that their raw materials were totally understood, with their properties tabulated. These folks dream of repeatable processes that they could invoke as if they were recipes. Andy and I feel that way—it would be nice to be able to assemble complex systems by aggregating well-understood components according to a verified set of rules.

But, wishing aside, we're not there yet. Software construction isn't like building houses in the burbs.

0740-7459/04/$20.00 © 2004 IEEE

This is our last Software Construction column—we're focusing our energies on getting our new Pragmatic Bookshelf publishing business launched. We'd like to thank the folks at *IEEE Software* for giving us this opportunity and thank you for reading these columns over the last three years. Do keep in touch with us at www.pragmaticprogrammer.com. —Dave & Andy

Get your hands dirty

So, if we're going to insist on using a construction metaphor, we'll have to accept that we're working more like the second group of architects. We don't fully understand our materials and our processes, so when we design and specify, we'll have to get more involved with the details. It's just the way it is.

What does this mean in practice? It means that, like the architect, we can't work from theory and abstractions—we have to get our hands dirty. Until we get to the point where code modules are as well understood as bricks and joists, senior industry people will have to stay involved at the construction end of things. Designers need to have coding experience in the types of systems they're designing. Architects should have worked on projects constructing similar applications. Educators and consultants should have genuine and productive work experience before telling their students how projects should be run.

But prior experience alone isn't enough. Just like an architect who visits the site to check that her ideas are translating well into the real world, these senior software people can't just throw another idea over the wall and leave it to others to find a way to implement them. Designers need to work with their teams to see how well the designs actually work in practice. Does something that seemed easy at a high level have nasty implications when you actually code it? Does a structure that seemed elegant and decoupled on paper turn out to be a bowl of spaghetti when you add in all the exception handling and error recovery?

Analysts need to do the same. Does what looked like a neat and tidy hierarchy during requirements gathering turn out to be not quite as structured when the real world intervenes? You won't know unless you're down there working with the team as it churns through the specification.

Equally, educators and consultants have a duty to discover if the techniques and practices they tout actually work. Often the best strategies from five years ago are less applicable today as technology and techniques move ever onward.

Back in the '80s, Patricia Benner addressed the problems of the nursing profession in her landmark book *From Novice to Expert: Excellence and Power in Clinical Nursing Practice* (Prentice Hall, 2001). Using the Dreyfus Model of Skills Acquisition, she turned around the training and development of an entire profession. One of her recommendations was to keep experts in practice. That's a lesson we can take to heart.

A practical challenge

A while back, we submitted a Construction column called "How to Produce Better Software." The first page was blank, apart from the title. The second page was blank too. In the middle of the third page, in nine-point type, was the word "practice." The idea didn't make it into print, which is a shame, because practice is really at the core of producing better software. It's likely that we can all do with more time to practice the important things, to experiment with new ideas and technologies. But time pressures are relentless, and this practice time is hard won.

So, here's a challenge. If you're an expert leading others in the field, ask yourself if you have all the current experience you need to direct that work. How much time do you spend working with the raw materials, getting the feel of them? How much time do you spend on projects to see how well your ideas translate into reality? How much feedback do you carry forward into your next project? Could you benefit from more practice down at the sharp end? If so, maybe now would be a good time to get that practice. If you're a designer who doesn't currently get the opportunity to code your designs, suggest that you want to do some programming on the next project. If you're an analyst who doesn't get to work with development teams once you finish your specification, ask to spend some time with them, working to map your ideas into working code. If you're a consultant or educator who doesn't get to spend time actually working on teams, offer to work on a project so you can see how well your ideas work in practice.

Maybe we should have asked that this department be renamed "Practice" when we took it on. Ultimately, that's what it's about.

Dave Thomas and Andy Hunt are partners in The Pragmatic Programmers and authors of the Jolt Productivity Award-winning *The Pragmatic Starter Kit* book series. Contact them via www.PragmaticProgrammer.com.

Classified Advertising

SUBMISSION DETAILS: Rates are $110.00 per column inch ($300 minimum). Eight lines per column inch and average five typeset words per line. Send copy at least one month prior to publication date to: Marian Anderson, *IEEE Software* Magazine, 10662 Los Vaqueros Circle, PO Box 3014, Los Alamitos, CA 90720-1314; (714) 821-8380; fax (714) 821-4010. *Email: manderson@computer.org.*

SOFTWARE CONFIGURATION ARCHITECT – Must have MS in Computer Science or related, 2 yrs exp in the job offered, and authorization to work in the U.S. on a permanent basis. Apply best J2EE practices including developing JSPs and EJBs. Map and build complete software configuration strategy. Correlate SCM process with building and deploying enterprise application on Windows/UNIX servers. 40-hrs/wk, 8a-5p, M to F. Qualified applicants send resumes to Humanizing Technologies, Attn: K. Fernung, 6325 Digital Way, Ste. 330, Indpls., IN 46278.
www.geo.oregonstate.edu/igert