

TOO MANY TOOLS

Graham Oakes yearns for a simpler approach to writing code.



Graham Oakes: how do you choose the right tool for the job?

One of the defining characteristics of humanity is the tools we use. Stone age. Bronze age. Iron age. We've even defined our history in terms of our tools.

So did people debate the benefits of 'best of breed' versus 'integrated' tools while they warmed themselves in their caves? I can just see the hunter with his single, multi-purpose spear arguing with the Bowman who had a special arrow for every prey.

These debates continue within IT today. Many programmers would much rather discuss editors, compilers, development environments and languages than actually write code. And I've lived through far too many debates about the pros and cons of monolithic ERP systems versus the pain of integrating multiple best of breed applications.

Nowadays many of the debates are dressed up in the language of service oriented architecture – 'My approach is more service oriented than yours' – but the underlying questions are the same. How do you choose the right tool for the job? And how do you make best use of the tools you've chosen?

The debate often arises because these questions point in different directions. Powerful, specialist development tools will do the best job, but you need more of them and they're often harder to use. At the extreme, you end up with a dedicated tool for every job, but you never use any one tool often enough to truly master it. So you end up doing a poor job anyway.

On the other hand, having a single development tool means you can invest in training and practice and infrastructure to support it. So you'll swing that hammer well as you drive in the screw.

Maybe it's a sign of age, but I'm increasingly leaning towards simplicity. I like to start with a small set of tools and learn how to use them well. I may occasionally use a 'good enough' tool rather than one that's perfect for the job, but that's all part of the learning – next time, I'll recognise the need to add a new tool to my set. (And if there isn't a next time, then it probably wasn't worthwhile finding a specialist tool for this one-off job anyway.)

As I achieve mastery of my existing tools, then I gradually accrue new ones for more specialist functions.

This also seems to work for the organisations I'm helping. In many places, the problem isn't that their software tools won't do the job. It's that they've never really put in place the required governance and skills to truly master their tools.

How different would it be if we'd defined our history in terms of skills, not tools? Hunter age. Farmer age. Machinist age. Blogger age...

● *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.*