# THE PARADOX OF CONTROL

*Don't forget the desktop when you manage application development, says Graham Oakes.*

**Graham Oakes: there is a lot of value tied up in spreadsheets**

"Build the right product. Build it the right way." Sounds good as an approach to application development… After all, application lifecycle management is an emerging trend among the tools vendors, and they're promising a lot. If we use their tools to control all the phases of application development (requirements, design, change and configuration management, etc), then we'll reduce our development costs, improve time-to-market, raise quality, ensure better compliance.

Of course it's easy to be sceptical about this. Anyone more than about 12 years old has heard these promises before. But for the moment let's posit that these tools work as advertised. Does this mean our application control problems are solved? I think not. All the tools in the world won't help if all the applications aren't in them.

When we say 'software development', most of us think about teams of Java programmers sweating over a hot pizza. We conveniently sweep a host of Access databases and Excel spreadsheets into the category of End User Computing. It's not 'real' software development, so it doesn't need our support. However, there's a lot of value tied up in those spreadsheets – budgets, transaction records, pricing tools and more.

What we've done is build a fortress around our 'real' software while relegating this spreadsheet value to a lawless zone where people with limited experience of development do the best they can to manage it. By trying to enforce controls, we've pushed these key assets into an uncontrolled domain.

So does this mean we should impose rigid controls over all spreadsheet development? No. The users wouldn't accept it, and we couldn't afford it if we tried. But we do need to find some way to:

- Identify critical spreadsheets and databases.
- Review and test them appropriately.
- Manage changes to them.

There are some interesting differences between end user computing and conventional software development. Requirements definition is a different challenge: the users often know their domain and requirements pretty intimately. Instead, areas like testing and change control come to the fore.

There's also an unanswered question as to when a user-developed application becomes critical enough to pull into a more controlled environment. My guess is that some spreadsheets are really application prototypes. Someone starts out developing a neat application and evolves it rapidly to a point where it's stable enough to roll out to additional users. Over time it stabilises further and gets rolled out more widely. At some point along this curve, tighter controls might be both justified and feasible.

So, by all means buy the tools. They won't solve all the problems of software development, but they may help. However as you do this, look outside the fortress – the end user hordes are massing and you need to manage them too!

● *EAI, Web Services & Software Development 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.*