MONTAGUE INSTITUTE REVIEW

ISSN 1554-303X

January 2006

Get Ready for End-User Development

The Montague Institute Review *is published by the Montague Institute and edited by Jean Graef.*

Within the last few years, some corporate IT managers have started to ask an intriguing question: how do we support end-user developers? These are employees who modify or create software to become more effective in their primary jobs as research scientists, financial analysts, sales, and other business roles.

The phenomenon is not new; it first took off in the 1980's with the introduction of microcomputers and electronic spreadsheets, but recently it has reached a tipping point. Instead of being tolerated or disparaged by IT staff (who sometimes call it "shadow computing"), it's emerging as a viable development option. In this article, we look at end-user prototyping and development — what is it, what are its pros and cons, and what opportunities does it hold for information professionals?

What is end-user development?

End-user development can be thought of as amateur computing in a business setting. It rarely shows up in a job description and is often done in the employee's spare time — on the lunch break, at the end of the day, or at home. It requires skill, creativity, persistence, and sometimes special tools, but it can deliver quick results and can be tailored to local requirements. There are several reasons why end-user development is gaining visibility and acceptance now:

1. *Users are more adept*. Today's users are more knowledgeable about computer technology. Many have experimented with macros, databases, templates, and Web publishing tools. They can get instant help through Internet discussion groups, and they have access to a growing variety of easy-to-use, inexpensive software tools.

2. *Users are more vocal.* Users are increasingly reluctant to abandon their home grown solutions even when they're offered another alternative with more features. Many have no qualms about voicing their dissatisfaction with products that they deem to be overly complex and inflexible.

3. *Limitations of traditional approaches.* When CFOs and CIOs look for reasons why IT projects fail or get into trouble, lack of user involvement is usually high on the list. For certain projects where user requirements are hard to define in advance, traditional development approaches may look too risky.

4. *Adoption of collaborative software tools.* The spread of collaborative software has the effect of transferring responsibility for security, Web site maintenance, and other technical functions from IT staff to business units. When that happens, IT faces the need for a new kind of support and training.

5. *The need for speed and agility*. As more economic value is based on intangible assets and the pace of change quickens, it's necessary to give knowledge workers the means to customize their own tools. The traditional alternative — interviewing users to discover their needs and then creating a product to meet them — is too slow.

The fact is that there are now too many end-user developers to ignore, they're too important to the company, and their numbers are expected to grow.

Tools and tasks

Unlike most employees, who use only a small fraction of the capabilities available on their desktops, end-user developers are adept at advanced features, such as macros, style sheets, indexing, export/import, scripting, and Web publishing. They are also likely to be good at data conversion — e.g. turning word processed text into rows and columns or records and fields. Their favorite tools are spreadsheets, desktop databases (e.g. Access, Filemaker), scripts, visualization tools, and "plugins" (small programs that extend the functionality of commercial products).

Typical examples of end-user development projects include the following:

• *Automating repetitive tasks*. Macros are popular with end-user developers because they can be created by "recording" a series of keystrokes instead of writing commands in an arcane language. The recording process, available in Microsoft Office, creates a series of commands that can be assigned to a button or key combination. Using Word macros, we created a button that would automatically reformat a text file into columns and rows that could then be imported into a database.

• *Personal knowledge management*. There are many products and Web sites that can be used to manage Web browser bookmarks, create a personal electronic library, or share information with a team. We have found that creating a custom database application (a knowledge base) is the most versatile way to accomplish these tasks — and to make the data available to business systems such as sales and billing.

• *Customizing commercial software*. Many packaged programs have free or low cost add-ons that extend their functionality. They're called plugins, Web parts, or APIs (application program interfaces). We customized a Dreamweaver plugin to create a data entry form that makes it easier for authors to enter Dublin Core metadata in Web pages. We used a database plugin to create scripts that could add, rename, and delete files on the user's hard drive.

• *Sharing data among applications*. Getting data out of one application and into another is a perennial problem. It's possible, for example, to export URLs from one Web browser to another but not to a database. We jumpstarted our corporate taxonomy by exporting index terms from a page layout program, removing page numbers, and importing them into our knowledge base.

• *Prototyping new applications.* Instead of writing a thick specifications document or creating screen mock-ups in Powerpoint, some business units are using modeling software to create working prototypes that look and behave like the real thing. The results can be reviewed and modified by a variety of stakeholders and can cut months off development time.

Pros and cons

Like other development strategies, end-user computing has its pros and cons. It's an alternative to, not a replacement for, other methods such as the traditional systems development process (the "waterfall method"), packaged software, outsourcing, and open source. As CIO's and enterprise executives become more aware of end-user development's benefits, they are coming up with ways to accentuate the positives and minimize the negatives.

For knowledge workers and their managers, the pros are speed and customization. The cons the time taken from the employee's "real" job and the potential for errors.

For the CIO, the pros include lower project failure rates and rework costs, less time spent on customization and maintenance, and higher user satisfaction. The cons include the difficulty of reusing code in other applications and the potential for service disruptions due to lack of quality assurance.

For the CFO and other enterprise executives, pros include lower overall IT-related costs, greater agility in responding to change, and higher customer adoption rates. For example, suppose a new Web-based application is projected to save $20 per customer but only 28% of them use it. If end-user prototyping can raise the adoption rate to 80%, the savings are much greater.

End-user development is showing up now on the executive radar screen in part due to the proliferation of enterprise software targeted to the issue. At $250,000 or more, these products require high level approval, and vendors are becoming adept at providing cost/benefit data to justify the investment. Their efforts, as well as those of computer science academics, focus on the hidden costs and intangible benefits of software development such as reduced risk, faster time to market, the costs of customization and rework, and missed revenue opportunities.

Differences between end-user and traditional computing

One reason that end-user development is faster and cheaper is that it's not encumbered by the practices required in a mission-critical, enterprise-wide environment running on expensive equipment. Instead of writing a formal statement of requirements, the end-user developer makes a few sketches on a notepad and jumps right into programming. Documentation is much simpler, and there's not as much pre-release testing. The developer simply starts using the application and fixes it as he goes along. There's less time spent on optimizing performance. If the application slows down, he simply replaces the computer with a more powerful model.

"Lite" IT practices

The longer a business unit engages in end-user development and the more people that use its products, the greater the need for documentation, maintenance, and quality assurance. Otherwise, the time required for training, bug fixing, and reinventing the wheel will eat up any productivity gains. To deal with this issue, we have gradually evolved a "lite" version of standard IT practices, such as:

• *Documentation*. Most developers don't like to write documentation, but it's well worth it in terms of time saved in teaching users, fixing bugs and creating enhancements, and working with outside contractors. The kind of documentation that we find most useful consists of previous software versions, step-by-step how-to instructions aimed at nontechnical users, and comments embedded in the software itself.

• *Electronic support system*. To save time, the developer needs at his fingertips a searchable database of relevant discussion groups, contact information for experts, a history of tech support calls, links to documentation and how-to articles, and product information (date purchased, serial numbers, warranty information, etc).

• *Data clean-up and capture.* end-user developers become sensitized to the need for clean, consistent data as well as codes that can be used as a "hook" into other systems. With nearly every significant new feature, we've had to correct historic data errors and/or capture new kinds data. For example, to get records in our knowledge base to sort correctly by date, we had to convert all publication dates into the MM/DD/YYYY format. To make it easier to reconcile accounting data from two different financial institutions, we had to capture a certain numeric code that appears on both statements.

• *Technical environment.* Most end-user developers learn the hard way to isolate their work so that it doesn't crash production systems or corrupt files. We designate certain computers as test machines. When possible, we standardize hardware and software to minimize the need for reprogramming. If it's necessary to have different operating systems, we stick to development tools that work on all of them.

Since end-user developers by definition have other jobs to do, it's important for them to resist the urge to add nonessential features and spend a lot of time tinkering with layouts. We've learned to get feedback from colleagues early and often and let new features evolve as our business processes require it.

Role of information professionals

So what should IT be doing to support end-user development? From our experience, it's mostly about better communication, better information, and more productive interdisciplinary partnerships.

Commercial software vendors, especially those with customizable products, show the way with special programs for developers. Most provide encouragement and sometimes funding for birds-of-a-feather meetings and discussion groups, short tutorials, and a searchable knowledge base of tips, plugins, demonstrations, sample code, and frequently asked questions. A similar service within corporations that is geared to do-it-yourself rather than professional programmers, focused on problems instead of specific products, and including a section on application integration, could increase the productivity of end-user developers and help leverage their work across the enterprise.

Prototyping gives end-users a richer language for communicating their needs, not only with IT staff but also with managers, contractors, and colleagues in other departments. The ability to see a working model makes it easier to get funding, reduces the cost of professional computing services, and can help streamline information flows across organizational boundaries. Even a minimal amount of documentation and promotion is a big time saver. By "promotion" we mean participating in communities of practice, entering a project description in a corporate database, or posting sample code on a Web site.

End-user development is not new, but because of a better access to help, easier to use programs, more comprehensive standards, and a more competitive business environment, it is becoming more widespread. Formerly operating under the corporate radar screen, it is becoming recognized as a viable IT strategy. It's valued by enterprise managers for its ability to reduce software implementation costs, shorten time to market, and increase adoption rates. It's valued by CIO's as a way to reduce project delays, budget overruns, and rework costs, and it's valued by users because it's fast and can be tailored to local conditions.

To capitalize on these benefits, knowledge base editors and other information professionals have three roles to play — helping IT create a more open, flexible, and user-friendly infrastructure, assisting end-user developers in adapting proven IT tools and techniques to their fast-paced and time-constrained environment, and acting as brokers to spread developers' products and expertise thoughout the enterprise.

READING LIST

• *Professional end user developers and software development knowledge.* Written by a professor of computer science at the Open University in the UK, this article discusses how to support "professional" end user developers. Drawing on data from two field studies — one with a financial consultancy and the other with a scientific research organization — the author describes their working environment and discusses the pros and cons of various strategies to help them. Her preference is communities of practice.

• *Managing software development*. Powerpoint presentation by Tom Malone, a professor at the MIT Sloan School and co-founder of the MIT Center for Coordination Science. Concise overview of six software development models, including end user development and prototyping.

• *Making it on their own*. CIO magazine overview of the topic, including quotes from business managers and discussion of various software products for end user modeling and prototyping.

• *"Evaluating the costs and benefits of end-user development"* in Proceedings of the First Workshop on End-user Software Development. Describes a cost/benefit model for end user development that compares perceived software benefits prior to implementation with actual benefits and costs after implementation. Discusses an example of a content management system in a university setting.

• *The business case justification for simulation software*. This white paper by iRise Corporation, a vendor of prototyping software, gives examples of cost/benefit calculations for a number of business scenarios. This link takes you to registration page, but the paper itself is free.

• *Companies look for ideas in all the wrong places.* An MIT professor argues for software that lets "lead users" build and modify products as they see the need for them.

• *Understanding information needs in technical work settings* Although not strictly focused on end user development, Vicki O'Day's work provides insights on how professionals work and why they might find the developer role appealing. ❑