GUEST EDITOR'S INTRODUCTION

The RAD Fad:

Is Timing Really Everything?

In our haste to speed development, we should consider if we are sprinting in the right direction. Is faster time-to-market equally important to every developer? Does a faster cycle time necessarily improve the chance for success? And how does the answer change from one year to the next?

DAVID N. CARD, Software Productivity Solutions

The software industry has experienced a series of fads. During the '70s, productivity was the fashionable concern, and a plethora of methods and tools were developed and marketed as productivity aids. During the '80s, quality took center stage, with a similar effect. Time-to-market, rapid development, and related themes may preoccupy the software community by the end of the '90s. This special issue provides articles, opinions, and news that will help the software practitioner understand the nature and potential of rapid application development.

James Martin coined the term "rapid application development" to apply to a specific
ARTICLE SUMMARIES: RAPID-APPLICATION DEVELOPMENT

  Neil C. Olsen
  The literature of software engineering rarely focuses on time as the dominant factor in software projects. But service velocity — the rate at which software can be deployed to market or customized — has a more significant influence on project decisions than quality, predictability, risk, cost, or productivity. Even in the noncommercial software market, where profit considerations may not dominate, the rate at which work can be serviced affects all other measurable elements more profoundly than any other factor.

  I have been a software engineer for more than 20 years, and schedules, market windows, and release dates have dominated my work process, my technical designs, and my life. This is generally true of my colleagues as well. As an engineer, I also feel the need to codify my observations and subjective experience with mathematical models so that I can understand my predicament, not merely testify to it.

  In this article, I make a case for improving the rate of development and deployment of software applications from the perspective of a business sponsor. My goals are to increase profit, beat the competition, establish a reputation, build market share, and increase shareholder value. My challenge is to select the best software technologies, staff, information-engineering processes, and organizational structures to achieve rapid deployment and satisfy customer demands.

* Internationalizing Software with Concurrent Engineering, pp. 39-46.
  Farshad Rafii and Sam Perkins
  For the software industry, the simultaneous launch of new products in multiple global markets has become a competitive necessity. The growing importance of overseas markets, the diverse demands of multinational customers, and the narrow profitability window for products with increasingly shorter life cycles compel software vendors to release internationalized versions of major products within months, if not weeks, of the original.

  Concurrent engineering promises to enable simultaneous launch by dramatically reducing the time and costs involved in internationalizing a base software version and customizing it for local markets. Yet concurrent engineering also has potential risks, especially when you apply it to products that represent quantum leaps in technology or when you must disperse development teams around the globe.

  We present a framework that outlines the range of options available to managers. These options vary along the dimensions of location and timing. We assert that the most suitable choice depends on two critical factors: potential market attractiveness and the development project's inherent technological uncertainty. When concurrent, distributed development does make sense, managers must focus on key factors such as project leadership, organizational structure, and measurement systems, because these play the greatest role in determining the successful outcome of projects.

  To support our recommendations, we report on the early stages of a research effort based on three detailed case studies conducted at a large hardware and software manufacturer that was concerned with significant delays in the internationalization of its software products.

* Large-Scale Industrial Reuse to Reduce Cycle Time and Cost, pp. 47-53.
  Emmanuel Henry and Benoit Faller
  All the changes in an organization's management, methods, and tools necessary to encourage reuse take time, delaying and sometimes wiping out the return on investment. We think that long-term reuse strategies must be based on short-term reuse successes. For this reason, our company — Matra Cap Systemes — founded its reuse strategy on pragmatic, opportunistic reuse-based projects.

  Here we report the results from two large industrial projects, in which project-based and cross-organizational reuse improved time-to-market, productivity, and quality (as measured through error rates). These positive results can be attributed to both reuse and the iterative nature of the development process. One project confirmed the interest of cross-organizational reuse. As a matter of fact, we record more and more projects that are reusing large parts of existing systems, although they come from different departments in the company.

  We now face another interesting consequence of our opportunistic reuse policy: several versions of similar modules, whose maintenance could be expensive if managed by different groups. For this reason, we are in the process of centralizing the maintenance of common code. We believe that our new interal-products policy will make a priori reuse easier in the long run.

methodology and toolset. However, the use of RAD has come to imply a wide range of technologies that attempt to reduce development time. Here, I use the term in a generic sense. RAD technologies include time-box scheduling, joint-application-development workshops, application generators, rapid prototyping, and many more. These technologies approach the problem in one of three ways: performing activities concurrently, avoiding some activities entirely, or delivering capability incrementally. Simply "working faster" is wishful thinking, not a meaningful solution.

  It is important to distinguish between time-to-market and software-development cycle time. Time-to-market is the time elapsed from project initiation to the first product's delivery to customers. The product may be more than just software. Moreover, the time to market is sure to be affected by factors other than the software process. The software-development cycle time is the amount of time it takes to produce a fixed quantity of software. This is different from the common tactic of negotiating the amount of capability to be delivered to fit within a fixed schedule. In this...
issue, Neil Olsen provides a high-level model of cycle time, and the article by Farshad Rafii and Sam Perkins illustrates the difference between cycle time and time-to-market. Rafii describes a business that, although it reduced its time-to-market through concurrent development, did not significantly change the cycle times of the individual software processes.

Cycle-time reductions are most often obtained by eliminating or automating activities such as rework, coding, and documentation. Incremental development approaches may be adopted to allow more concurrency within a software process. Incremental and prototyping approaches may reduce time-to-market by getting some partial capability into the customer's hands early. In this issue, Emmanuel Henry and Benoit Faller show how software reuse (avoiding code development) significantly reduces cycle time.

PASSING FAD? Is RAD just a passing fad? To answer that question, it helps to examine earlier fads. The productivity and quality initiatives of the '70s and '80s did contribute to some successes, but they have also led to visible failures. These failures are usually blamed on poor technology, poor implementation of the technology, and poor management. Indeed, many software engineers are already pointing out the importance of effectively managing RAD.2

However, a more fundamental problem is the potential mismatch between the technological fad and the needs of business. Businesses must consider whether or not there is a match at both the strategic and tactical levels. Does an organization really need the benefits the new fad promises? Of course it is always good and desirable to reduce cost, increase quality, and shorten time-to-market — but are they equally important to everyone's success?

At the strategic level, a business should consider the importance of time-to-market to its success. Figure 1 shows two factors that shape markets: the number of consumers and the number of suppliers. These two factors define four markets, each of which requires a long-term business strategy:

- **Capability.** Most markets begin with relatively few producers and consumers. In this type of market, capability is the most important business factor — offering something the competition doesn't. Early adopters of new products are often willing to pay more and accept lower quality than the normal customer.

- **Cost.** In a market with relatively few consumers and many suppliers, consumers dictate the capability, quality, and schedule they desire. The suppliers' only possible business strategy is to meet those requirements at the lowest cost.

- **Time-to-market.** In a market in which only a few producers compete for many customers, the business that gets its product out first will capture a large part of the market. Generally, the competitors are well-known to each other.

- **Quality.** In a fully mature market with many suppliers and consumers, quality is a primary determinant of success. It is difficult to get to market first because there are too many competitors to track and stay ahead of. It is difficult to have the lowest cost, because someone else can always run a sale. Because these markets are mature, the potential capabilities of the products are well-understood — truly revolutionary advances are unlikely, and when they occur they often create a new market.

CUSTOMIZE YOUR INVESTMENTS. To get the best return, businesses should invest in technology that will give them a competitive advantage within their market. Markets change, so business strategies must change to keep pace. It is common for time- and cost-sensitive markets to change to quality-sensitive markets as more consumers and suppliers enter. An organization that invests in technology that supports a business strategy for a market as it existed a few years ago probably won't gain a competitive advantage. Of course, focusing on only the primary strategy and ignoring all others can also lead to disaster. Ed Yourdon suggests that the current fad has made quality an inflexible ideal, when careful tradeoffs of capability, cost, schedule, and quality should be considered.1

Obviously, the simple model in Figure 1 does not capture all the market factors that affect business strategy, but it illustrates the point that there are markets in which time-to-market is critical and others in which it is less so. Nevertheless, a recognition that there are significant markets in which productivity and quality are not paramount motivates the current interest in RAD. This segment of the software market has not previously been the focus of attention by researchers, methodologists, and tool vendors.

At the tactical level, the decision to invest in a specific RAD technology must be based on a detailed under-
standing of the business's processes. "Rapid" techniques do not always reduce cycle time or time to market. For example, a survey of 39 rapid-pro-totyping projects did not report any instance of schedule reduction. Achieving a significant schedule reduction requires first understanding where delays are occurring and then applying technology to address those specific problems, not just adopting a generic set of "accelerated" methods and tools. The delays in delivering software to market may have nothing to do with software. For example, the requirements-definition and acquisition phases of many government software projects often exceed the development phase. What this discussion shows is that RAD responds to a real need of businesses competing in time-sensitive markets, but that its adoption by others will probably have a limited effect. The RAD fad will help some businesses, but many others will be disappointed. Nevertheless, RAD technology does offer new help to those businesses competing in a type of market that has previously been largely ignored — the market in which getting there first is everything.

REFERENCES

Meet the challenges of fast and reliable software development -

T. LAUER
PORTING TO WIN32
A Guide to Making Applications Ready for the 32-bit Future of Windows
This book provides a comprehensive guide to upgrading applications from 16-bit to 32-bit programs designed to run under Windows NT or Windows 95. Readers are assumed to have a solid grounding in Windows programming, but this may be their first introduction to the Win32 API layer. The author begins by introducing the new Win32 systems and describes the six main areas of difficulties in porting Win16 applications to Win32 environments. He describes methods for planning the transition of large applications and goes into great detail regarding the design of C and C++ programs for portability. Although the book shows mainly C examples, it is also suitable for Pascal and Modula-2 developers. The final chapter discusses in depth the usage of the new 32-bit development tools and the process of creating Win32 applications.

R. HALL and D. FORBITH both, Civic in Perspective.
INTERACTIVE 3-D GRAPHICS IN WINDOWS
This is a hands-on book which uses a component software approach to help Visual C++ programmers quickly and easily develop Windows integrated, interactive 3-D graphics applications. The book includes JOEY, a 3-D user interface toolkit which addresses interaction issues not dealt with in the Microsoft User Interface Style Guide. JOEY provides a 3-D user interface, 3-D tools, C++ linking and Embedding, and C++ automation within the MFC framework so that the application programmer can focus on application functionality. Using this book and JOEY, an experienced Visual C++ programmer can create an interactive 3-D application in a few hours. This combined book and software package provides a complete graphics package suitable for developing applications interactively.

R. KEHOE, Intel Corporation, Santa Clara, CA, and A. SHAH-JARVIS, Stratcom, San Jose, CA
ISO 9000-3
A Tool for Software Product and Process Improvement
ISO 9000-3 is the internationally recognized standard of quality assurance for software and its use is rapidly becoming the norm throughout the software industry. As yet, few software organizations meet the process quality required to meet the standard, but increasingly the industry will seek to ensure that all software can be certified under the standard. This book includes templates forms which are useful to document the design development and testing as well as plenty of practical guidance for managers responsible for ensuring that software meets the standard.

A.M. CHRISTIE, Carnegie Mellon University, Pittsburgh, PA
SOFTWARE PROCESS AUTOMATION
The Technology and Its Adoption
Process automation provides a means to integrate people in a software development organization with the development process and the tools supporting that development. This new technology may significantly improve software quality and development productivity. However, there is little practical experience in its day-to-day use. This book is for those who wish to explore the technology or are considering its adoption. Special emphasis is on the process modeling language ProNet which is commercially available and numerous examples, tables, and technical appendices are provided.

ORDER/PRODUCT INFORMATION:
- Call 800-SPRINGER (800-777-4646) (7:00 am to 5:00 pm IST 7:00 am to 5:00 pm IST) for the book, and for all in-print titles. For information on all Springer titles, call 800-SPRINGER (800-777-4646)
- Write to Springer-Verlag New York, Inc., 175 Fifth Avenue, New York, NY 10010

Reader Service Number 10