Developing quality software “just in time” has been one of the most significant challenges of the 1990s. Software tools and new software methodologies can together play a key role in achieving a higher level of software quality and productivity. Software quality can be greatly improved by selecting a correct development tool to assist in each phase of the development process, from requirements analysis to final testing and integration. Selecting an inappropriate tool, on the other hand, can actually hinder software development.
The Impact of Tools on Software Productivity, pp. 29-37.

Tilmann Bruckhaus, Nasim H. Madhavji, Ingrid Jansen, and John Henshaw

To stay competitive, software organizations must continuously improve product quality and customer satisfaction, as well as lower software development costs and shorten delivery time. One way to do this is to adopt appropriate software tools. However, to make effective use of specific tools you should first understand how a tool will affect these critical variables in your project.

Because we don't know how to analyze a tool's impact on specific projects, we generally adopt them based on an intuitive understanding of their expected impact. In many cases, the actual results of this practice are disappointing. The problem is aggravated because tool adoption often brings considerable costs.

The authors did a case study on the impact of tool insertion in ongoing software projects. The result of their study was a method that organizations can use to assess the impact of tool insertion on software productivity.

Alan W. Brown and Kurt C. Waldau

Software development organizations continually make decisions on how to select, apply, and introduce software technology. Companies decide on some technologies explicitly after examining alternatives in detail; other technologies are selected with little study of the decision's potential impact. In both cases, the organization attempts to understand and balance competing concerns regarding the new technology. These concerns include acquisition costs, the technology's effect on quality and time to market, and the training and support services it will require.

Attempts to calculate the return on investment of software technology have generally failed. The foremost reason is the difficulty in establishing cause and effect when assessing new software technologies' impact on an organization. The authors' experimental framework can help companies evaluate a new software technology by examining its features in relation to its peers and competitors through a systematic approach that includes modeling and experiments.

Performance Testing a Large Finance Application, pp. 50-53.  
David Grossman, M. Catherine McCabe, Christopher Staton, Brett Bailey, Ophir Frieder, and David Roberts

Developers have been performance testing large database applications for more than 20 years. Up to now, they have focused primarily on system-level testing of operating systems and database management systems. However, just because a machine, its operating system, and its database system are fast does not mean that an application using the database system will perform well. Design flaws in the application or in the tuning parameters specific to the application's environment often result in serious performance problems.

The authors present a case study that shows how a simple prototype can be used to verify, before production, that a system will perform at an acceptable level under realistic conditions. Their approach hinges upon the use of a simple prototype to verify that the system performance falls within acceptable bounds.

John P. Hadepo, Stephen J. Aud, Taghi M. Khoshgoftaar, Edward B. Allen, and Jean Mayrand

As software becomes more and more sophisticated, industry has begun to place a premium on software reliability. Consequently, software reliability is a strategic business weapon in an increasingly competitive marketplace. In response to these concerns, BNR, Nortel, and Bell Canada recently developed Emerald, a design support system that promotes telecommunication software reliability. Emerald efficiently integrates software measurements, quality models, and delivery of results to the desktop of software developers.

Emerald not only improves software reliability, but also facilitates the accurate correction of field problems. The authors' experiences developing Emerald taught them valuable lessons about the implementation and adoption of this type of software tool, which they present here.

Alan Chmura and David Sharon

Migrating legacy systems and developing new systems for client/server environments has dominated the software development tool market in the '90s. In the last two years, the Internet's surging popularity has led many established tool developers to provide Web functionality for their legacy migration and client/server tools. The authors provide a sampling of what's available in the many new and upgraded tools available to help you create client/server and Internet applications, along with tips for picking those best suited to your needs.
KEY CRITERIA

What we need, then, is an increased awareness of how useful available tools are and how they can be applied to different applications and the different phases of a software project. We must develop simple, usable, and yet comprehensive criteria for the evaluation of software tools. Case studies and experience from users and experts must be presented to as wide an audience as possible. In pursuit of that goal, this special issue attempts to provide

- information on the most up-to-date and innovative software development tools and methodologies available today,
- criteria for qualitative and quantitative evaluation of tools and methodologies, and
- case studies on the application of the evaluation criteria.

In assessing the quality of a software tool, you could start with Standards such as ISO 9126. However, as Alan Brown argues, evaluating a specific CASE tool is different from evaluating a CASE environment provided by a suite of tools. For example, the “integration” of the component tools in terms of their cohesiveness, interface compatibility, and shared conceptual notions is appropriate only in the case of multiple tools.

We go one step further and state that it is not feasible to design a single set of criteria that can be used to evaluate all software tools and environments. The current approach taken by the software community is to provide evaluation guidelines whereby different evaluation criteria may be developed to suit different software projects. For example, the ISO 9126 standard proposes the following characteristics to assess a software product’s quality:

- **Functionality.** Does the software product exhibit required functions to satisfy particular needs?
- **Usability.** How much effort is needed to use the software?
- **Reliability.** How capable is the software product in maintaining its level of performance?
- **Efficiency.** What resources are needed to maintain the required level of performance?
- **Maintainability.** What effort is needed to make specific changes to the product?
- **Portability.** What effort is needed to transfer the product from one operating environment to another?

These characteristics can be used as guidelines to develop more specific evaluation criteria for software tools.

APPLYING CRITERIA

It is not enough, however, simply to design evaluation criteria; they must also be applied. A metric is only useful if it compares different tools, either qualitatively or quantitatively, for a specific characteristic. Applying an evaluation metric to a tool may itself require other tools. Alternatively, customers can be surveyed to evaluate the impact of software tools in their work environments. For example, one study evaluated against the ISO 9000 standard 17 different CASE tools used by German software industry. The evaluation is based on a carefully conducted survey of the industry in which customers were asked to name a best tool and a worst tool for each of several well-chosen criteria.

The impact of a tool on the quality or productivity of a project depends not only on the characteristics of the other tools used but also on the characteristics of the project itself. Elsewhere in this issue, Tillman Bruckhaus presents some interesting case studies showing the impact of project size on overall productivity. His study also shows that productivity is affected by the development process used. For example, the use of a sophisticated tool in a project that relies on a simple development process may actually decrease the productivity. This is intuitively appealing since a development process that does not need sophisticated tools can be hampered by the steep learning curve they require.

The evaluation of tools and methodologies for their effect on developing quality software and improving the productivity of the development process is finally receiv-

It is not feasible to design a single set of criteria that can be used to evaluate all software tools and environments.

ACKNOWLEDGMENTS

We thank Al Davis and the anonymous referees who helped with the review process, without whose help this special issue would not have been possible.
1. Misunderstanding key user problems and needs
2. Discovering missing or wrong requirements late in development
3. Not communicating requirement priorities and status to the team
4. Underestimating the cost of changing requirements

These mistakes cause schedule delays, missed expectations and even project cancellations. You owe it to your team to learn about Requisite, the leading groupware tool for requirements management. Requisite makes your projects easier to build, easier to test, and easier to manage. And because it integrates with Microsoft Word, it won't change the way you work.

Effective requirements management is where we can often achieve the greatest leverage in application development, and Requisite does an excellent job. 

Ed Yourdon, SOFTWARE AUTHOR AND CONSULTANT, REQUISITE, INC. BOARD MEMBER

REFERENCE