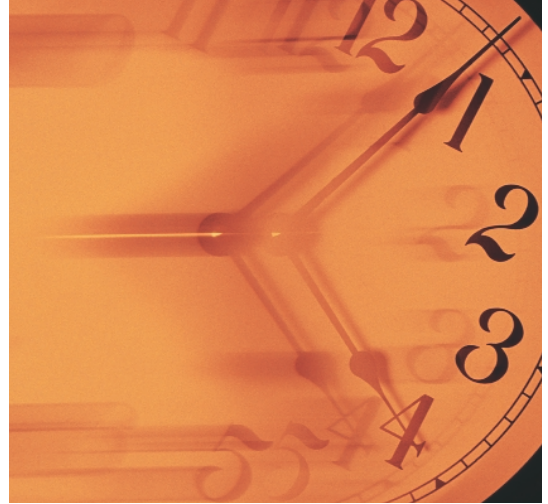


A leaner version of traditional capacity planning can help you hone system performance without inflating schedules or delaying time to market.

Neil J. Gunther



Hit-and-Run Tactics Enable Guerrilla Capacity Planning

We so-called performance experts tend to regurgitate certain performance clichés to one another and to anyone else who will listen. Here are two such clichés:

- Acme Corp. lost a \$40 million sale because its new application cannot meet service-level targets under heavy load. How much money must the company lose before its management embraces capacity planning?
- Company XYZ spent \$1 million buying performance management tools, but it won't spend \$10,000 to train its staff to use the capacity planning functionality. They just continue to produce and consume endless strip charts without regard for what that data might imply about their company's future.

Several years ago, I stopped mindlessly reiterating stories like these and took a hard look at the real situation: People were not just failing to gravitate toward capacity planning, they were actually avoiding it at all costs. I devised guerrilla capacity planning—tactical methods for quickly assessing capacity and performance—to solve the problems managers had with traditional capacity planning.

WHY MANAGERS RESIST CAPACITY PLANNING

In coming to grips with this

new awareness, I discovered some unusual reasons managers had for avoiding capacity planning. Two of the most compelling have to do with misperceptions about risk.

As long as it fails on time

Some executives and managers believe they don't need to do anything about capacity planning, because it will always be someone else that loses millions of dollars due to poor performance. But this is a false perception about risk. Such faulty perceptions often subvert effective risk management. The "Risk Management versus Risk Perception" sidebar explains why this view is so deep seated.

Companies primarily employ managers to control schedules. To emphasize this fact to my students, I tell them that managers will even let a project fail—as long as it fails on time! Many of my students are managers, and none has disagreed with me yet. In other words, managers often resist capacity planning because they are suspicious that it will interfere with project planning. Under such scheduling pressures, the focus is on functionality first.

Unfortunately, senior managers and marketing often over-prescribe new functionality because they see it as a competitive differentiator. All the development time, therefore, tends to go toward implementing the new functionality. In the clamor to get the new functionality to market as fast as possible, the overall application often fails to meet performance expectations. So, the application arrives on schedule, but it fails to meet all the requirements (especially performance requirements); it fails on

Inside

**Risk Management
Versus Risk
Perception
Guerrilla Tools
Guerrilla Resources**

time! As this article tries to explain, application development doesn't have to be like this.

Performance homunculus

A homunculus is the sensate representation of the human body. Reflecting this sensory weighting, the homunculus' hands and mouth are huge, while its torso and head appear relatively small, as Figure 1 shows. This is because humans receive vastly more sensory information via fingers and tongues than via the skin on our torsos, for example.

Just as the homunculus emphasizes the disproportionate weighting of our senses to our bodies, so too performance management (which draws on many skills and disciplines) should receive a disproportionately large part of system management investment. This is not, however, conventional wisdom.

Performance management, most often, is seen as just another system management activity. In other words, the resources required to carry out performance management are roughly the same as those required for managing the distribution of software, backup and recovery, charge back, and security. But this is another misconception like the one involving risk. Almost without exception, it's possible to accommodate most of the latter activities by purchasing appropriate COTS (commercial off-the-shelf) packages, which require little support in terms of human infrastructure.

Because performance management in general (and capacity planning in particular) is like a homunculus, it requires disproportionately more resources, more human infrastructure, and more training than almost any other system management activity. Nonetheless, the plethora of COTS performance tools on the market continues to provide false assurance that performance management is just another limb on the body of system management, rather than the huge helping hands of the homunculus that it could be.

WHY THE CAPACITY PLANNING PROCESS MUST CHANGE

Given these observations, I began to think about explaining why managers were avoiding capacity planning in today's computer environments (Neil Gunther, "Shooting the RAPPIDs: Swift Performance Techniques for Turbulent Times," *CMG 97 Proc.*, Computer Measurement Group, Turnersville, N.J., 1997, pp. 602-613). I realized that

Risk Management Versus Risk Perception

Consider the poor fellow driving to the airport with white knuckles because he just heard a radio news report about a plane crash, and now he's fretting about the safety of his upcoming flight. What's wrong with this picture? Statistics tell us that he has a greater risk—by a factor of 30 or more—of being killed on the freeways than the airways.



Our traveler has also heard these same statistics. So, why doesn't he remind himself of this important fact and look forward to his flight, in spite of the air disaster that day? Try doing this sometime—it doesn't work. It's a psychological issue, not one of rational thought.

On the freeway, our intrepid driver feels he is in control because his hands are firmly on the steering wheel. But on the aircraft, he is just another fearful passenger strapped into his seat. This fear registers at a deep, personal level of (false) insecurity. On the freeway, he remains oblivious to the possibility of complete obliteration by another careless driver.

This illustrates the essential difference between risk perception and risk management. Companies pay managers to be in control. Therefore, a manager's perception is that bad things will not happen to his project because doing so would be tantamount to admitting that he is not really in control, a catch-22 situation.

Figure 1. Performance homunculus.



Capacity planning is more like the hands of the homunculus than simply part of the torso (systems management).

it's the capacity planning process itself that must change.

Software is (often) the performance bottleneck

Capacity planning has long been accepted as a necessary

evil for mainframe and data network procurement (Frank Huebner, "Performance and Capacity Evaluations of IP Networks and Systems," *IT Professional*, Nov.-Dec. 2001, pp. 38-43). The motivation is simple: Hardware components have always been expensive, and budgets have always been limited. But today, hardware—even mainframe hardware—has become relatively cheap.

In launching a new application and system, however, IT professionals must consider a less-obvious caution: Bottlenecks are more likely to arise in the application software than in the hardware. So throwing more hardware at a performance problem might not necessarily help.

In this sense, capacity planning has not gone away. Time is money, even if you have all the hardware in the world. The new emphasis is on software scalability, and that

emphasis affects the way you do capacity planning. Today's demands on time and scheduling constraints no longer accommodate the traditional detailed approach to capacity planning based on procurement for the monolithic mainframe. Moreover, today's computer systems employ distributed computing architectures. Distributed computing comprises many software pieces running in many hardware places.

Complex distributed-computing environment

Several factors make the more complex environment of distributed computing less amenable to traditional mainframe capacity planning methods:

- Third-party or in-house applications have little or no instrumentation.
- There's no such thing as Unix; there's AIX, HP-UX, Solaris, BSDI, FreeBSD, RH Linux, Debian Linux, the MacOS, and so on.
- Scripts built on one form of Unix are almost guaranteed not to work on another variant.
- There's not even one type of Windows operating system anymore.
- Multiple COTS applications run on multiple vendor platforms.
- Today's systems lack a common performance database like RMF (Resource Measurement Facility) or SMF (System Management Facility), which are available for MVS (multiple virtual storage) mainframes.
- Most commercial tools have mainframe roots and are thus server-centric in their data collection capabilities. Monitoring network and application data does require additional tools.
- There's no convenient way to comprehend resource consumption across multiple tiers.

Think of traditional capacity planning as the 800-pound gorilla. That gorilla needs to go on a diet.

Little standardization or instrumentation

The Universal Measurement Architecture (UMA) standard from the Open Group (<http://www.opengroup.org/pubs/catalog/c427.htm>) could have helped surmount some of these difficulties. This standard offers a means to normalize collection and management tools for performance data (especially Unix data).

But vendors and other competing parties had no strong incentive to adopt UMA. To provide performance data in UMA format, platform vendors would incur a cost burden without an obvious return on investment. Tool vendors thought a standard format made it easier for new competitors to enter the market, threatening their revenue streams. Similarly, ARM (Application Resource Measurement), a standard primarily promoted by Hewlett-

Packard and Tivoli, has not really caught on either.

So, today's IT professionals are building more complex architectures with less instrumentation available to manage them. I'm glad Boeing doesn't build aircraft this way!

GETTING STARTED WITH CAPACITY PLANNING

By now it should be clear that capacity planning is as important as ever, but capacity planners today must surmount two new hurdles that traditional planners did not:

- We're trying to do more with less. Distributed systems are ubiquitous and more complex than monolithic mainframes, but we have fewer standardized tools to manage performance and plan for growth.
- We have to do more in less time. Whatever performance management and capacity planning we do, it must be done in less time so as not to inflate product/project schedules.

Taken together, these constraints seem impossible to meet. Is it any wonder that managers give up and ignore capacity planning? But, as you will see, there is a way out.

First, it's important to realize that performance management has three major levels: monitoring, analysis, and planning.

Managers usually pay the most attention to performance monitoring, because it is generally the easiest area to address. If you want to manage performance and capacity, you must measure it. Naturally, this is the activity that the majority of commercial tool vendors target. (As a manager, if you spend \$250,000 on tools, you inevitably feel like you must have accomplished something.)

If you can't afford these prices, you can turn to your Unix or Windows NT system administrator. They are usually very good at writing scripts to collect all sorts of performance data and presenting it to you in your Web browser.

But data collection just generates data. The next level is analysis: uncovering the information hidden in the data. These days, unfortunately, the usual motivation for performance analysis is to firefight an unforeseen performance problem that is delaying a release schedule or reducing deployed functionality. This is performance analysis *after* the horse has bolted.

With a little more investment in infrastructure, managers can plan ahead, minimizing the outbreak of those unforeseen fires. But, as was explained earlier, managers usually skip this third level of planning for fear of inflating project schedules. How can managers cut this Gordian knot?

GUERRILLA CAPACITY PLANNING

In my view, managers need a more opportunistic approach to capacity planning; enter guerrilla capacity planning.

The notion of planning tactically might seem contradictory. At the risk of mixing metaphors, think of traditional capacity planning as the 800-pound gorilla. That gorilla needs to go on a diet to produce a leaner approach to capacity planning that is adapted to today's business environment. By lean, I don't mean skinny. "Skinny" means remaining stuck in the rut of simply monitoring everything that moves with the false hope that capacity issues will never arise, and thus avoids the planning level altogether. Monitoring requires that someone watch the "meter needles" wiggle. Inherent in this approach is the notion that action isn't necessary unless the meter redlines. But performance meters only convey the system's current state. Such a purely reactive approach does not provide any means for forecasting what lies ahead. You can't forecast the weather by listening to leaves rustle.

The irony is that a lot of predictive information is likely contained in the collected monitoring data. But, like panning for gold, it takes additional processing to reveal the hidden gems about the future.

It's not a model railway

The goal of capacity planning is prediction and that requires a consistent framework in which to couch any assumptions. Capacity planners call that framework a *model*. The word "model," however, is one of the most overloaded terms in English; it can mean everything from a fashion model to a model railway set.

For example, the best model train set is usually the one

Guerrilla Resources

Books

- *Computer Performance Analysis with Mathematica*, A. Allen, Academic Press, San Diego, 1994: Contains a very readable overview of capacity planning issues and techniques written by a master of the subject. Also, it is the only book I know of that shows you how to apply Mathematica to solve performance problems.
- *Guerrilla Capacity Planning: Hit and Run Tactics for Sizing UNIX, Windows and Web Applications*, N.J. Gunther: This book is in preparation; see <http://www.perfdynamics.com> for information.
- *Sun Blueprints: Capacity Planning for Internet Services*, A. Cockcroft and W. Walker, Prentice Hall, Upper Saddle River, N.J., 2000: One of these authors has adopted and extended many of the concepts I present in this article. The book also outlines guerrilla-style management processes and scenario planning; some chapters are Sun/Solaris specific.
- *The Web Testing Handbook*, Steven Splaine and Stefan P. Jaskiel, STQE Publishing, Orange Park, Fla., 2001: This book covers the interface between functional testing and performance measurement. Chapter 8 covers application scalability and, in particular, the "retrograde" throughput effect discussed in example 1 of this article.

Classes

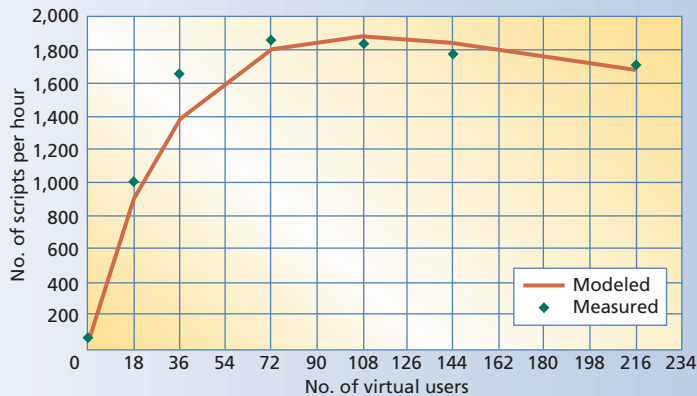
- **Guerrilla Planning** (<http://www.perfdynamics.com/PitV/guerrilla.html>): I teach a five-day class on these techniques.
- **Guerrilla Tools** (<http://www.perfdynamics.com/PitV/gtk.html>): This five-day class is a lab in which students participate in the construction of performance and capacity models.



that includes not just a scale model of the locomotive, not just a model of an engineer driving the scaled locomotive, but the one that includes the pupil painted on the eyeball of the engineer driving the scaled locomotive. In other words, the more detail the better.

This is precisely what a capacity planning model is not. For capacity planning, the goal is to *discard* as much detail as possible while still retaining the essence of the system's performance characteristics. This goal tends to argue against the construction and use of detailed simulation models, in favor of spreadsheets or even automated forecasting. The skill lies in finding the correct balance. Linear trend models can be too simple in many cases, while event-based simulation models can be overkill. To paraphrase Einstein: Keep the model as simple as possible, but no simpler.

Figure 2. Load test measurements and capacity model.



No compass required

Traditional capacity planning has required relatively high precision because each significant digit of the calculation had many thousands of dollars attached to it. In today's lower-cost climate, however, managers often just want a sense of direction rather than the actual compass bearing. In this sense, the precision of traditional capacity predictions has become less important than their accuracy. Managers often see little virtue in spending two months debugging and verifying a full-blown simulation if the accuracy of a simpler spreadsheet model would suffice.

At a level of performance data collection, there is little support for high-precision measurements in open systems. Take Unix, for example: It's basically an experiment that escaped from the lab around 1975 and has been mutating ever since. What little performance instrumentation existed back then in the Unix kernel was for the benefit of early developers, not the grand purpose of capacity planning in today's distributed systems. Nonetheless, every capacity planning tool in existence relies primarily on those same kernel counters with little modification. And since the PC revolution of the 1980s fueled the move to distributed systems, performance management has become ad hoc, at best.

Guerrilla capacity planning, on the other hand, tries to facilitate rapid forecasting of capacity requirements based on available performance data without inflating schedules.

Consistent investment in human infrastructure is also important. People must have the training and know-how to use the methods and tools at hand. Another key point is to keep ahead in the procurement cycle. This was never truer than it is for exponentially growing Web sites. If you don't accurately forecast capacity, user traffic can disappear to competitor sites. Or latent and unanticipated demand can consume newly procured capacity the instant you install new servers. Let's look at two brief examples of guerrilla capacity planning in action.

Example 1: Sizing by numbers

As a first example of how to use guerrilla capacity planning, here is a tactical method for quantitatively determining application scalability. It's noteworthy that scalability—particularly application scalability—is a perennial hot button that involves notions of performance and planning, yet few people can quantify the concept.

Scalability has to do with the laws of diminishing returns so a single number cannot represent it. Scalability is a function. Figure 2 shows an example of actual load test throughput plotted as number of scripts S executed per hour on the y-axis as a function of number of virtual users N on the x-axis.

Superimposed on this data is the corresponding scaling function predicted by a simple capacity model that does not involve queueing theory or simulations. This means that sizing server capacity can be relatively quick using a spreadsheet.

The formula for the simple capacity model is

$$S(\alpha, \beta, N) = N / \{1 + \alpha [(N - 1) + \beta N (N - 1)]\}$$

It involves just two parameters (N.J. Gunther, *The Practical Performance Analyst*, iUniverse Inc., Lincoln, Neb., 2000). You identify α with contention delays, for example, time spent waiting on a database lock. β is associated with additional delays due to pair-wise coherency mismatches, such as time to fetch a cache miss. These delays can be in hardware, software, or (most likely) a combination of both.

You can easily enter this function in a Microsoft Excel spreadsheet. If the number of virtual users resides in column N , and regression parameters α and β reside in cells A1 and B1, the equation becomes the Excel cell formula

$$=Nn/(1+A1*((Nn-1)+B1*Nn*(Nn-1)))$$

where Nn is the value in the cell at column N and row n . You can now determine scalability parameters α and β using the linear-regression tools built into Excel (see, for example, D. Levine, M. Berenson, and D. Stephan, *Statistics for Managers Using Microsoft EXCEL*, Prentice-Hall, Upper Saddle River, N.J., 1999). In summary, the basic steps of this guerrilla technique are as follows:

- Measure throughput as a function of load N using tools like LoadRunner.
- Collect a sparse data sample; having at least four load points is usually sufficient.
- Calculate α and β by performing a regression fit of Excel against transformed versions of N and S .
- Use those values to predict the complete application scalability function using the Excel formula discussed earlier.

You can find a more detailed account of this procedure at <http://www.teamquest.com/html/gunther/fitting.shtml>.

An essential feature of this simple model is that it can predict *retrograde throughputs* (where the amount of completed computation decreases as the system load increases) like that in Figure 2. This effect is not easily modeled using conventional performance modeling tools based on queueing theory without specialized load-dependent servers.

What are the benefits of this guerrilla sizing methodology? Primarily, it avoids the need for more complicated queueing theory or simulation models. However, the most significant benefit is not the model's technical merits but the fact that it creates a framework against which to check the consistency of load measurements. If the data does not fit the model in the first equation, there is very likely a problem with the measurement process that may be worth more detailed investigation. Moreover, because each of the model's terms has a real physical interpretation, engineers from disparate groups quickly recognize which parts of the application or platform need further tuning to improve scalability. In this way, the spreadsheet model is a capacity planning tool that forecasts scalability without inflating the release schedule.

There are many other tools that can help in such a "guerrilla" analysis; the "Guerrilla Tools" sidebar lists a few.

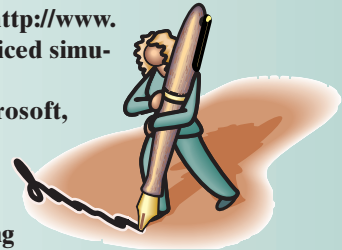
Example 2: Estimating capacity doubling time

You can also apply the spreadsheet model to Web site traffic analysis, where the rapid increase in traffic growth demands a more tactical approach to capacity planning. The growth of sites like those of eBay, AOL, and Amazon might not be as extreme as it was last year (A.C. Lear, "Managing e-Commerce Reliability, eBay Style," *IT Professional*, Mar.-Apr. 2000, pp. 80, 77-79), but server demand will still grow. As already noted, managers of these Web sites know they need capacity; it's the planning part that is culturally unfamiliar.

I have devised a useful metric for high-growth Web sites, the *capacity doubling period* (N.J. Gunther, "Performance and Scalability Models for a Hypergrowth e-Commerce Web Site," *Performance Engineering: State of the Art and Current Trends, Lecture Notes in Computer Science 2047*, Reiner Dumke and Claus Rautenstrauch, eds., Springer-Verlag, Heidelberg, Germany, 2001). This period is simply the time until the amount of consumed processing capacity is twice that now being consumed. In some cases, this

Guerrilla Tools

- **CSIM** (Mesquite Software, <http://www.mesquite.com>): Moderately priced simulation package.
- **Excel** (Microsoft, <http://www.microsoft.com>): Spreadsheet in Microsoft Office suite.
- **MathCad** (MathSoft Engineering & Education Inc., <http://www.mathsoft.com>): Commercial symbolic-computation program.
- **Mathematica** (Wolfram Research, <http://www.wolfram.com>): Commercial, general-purpose program for symbolic computation.
- **Minitab statistical package** (<http://www.minitab.com>): Next level above Excel.
- **PDQ** (Performance Dynamics Consulting, <http://www.perfdynamics.com/Tools/PDQ.html>): An open-source performance analyzer.
- **R** (<http://www.r-project.org>): Public-domain version of S+ statistical-analysis product.
- **SPE *ED** (Performance Engineering Services, <http://www.perfeng.com/sped.htm>): Tool for software performance engineering (SPE).
- **TeamQuest performance software** (<http://www.teamquest.com>): Commercial capacity-modeling tools.



period can be as short as six months. That's about 10 times faster than typical data processing centers and four times faster than the original version of Moore's law.

Such exponential demand for server capacity can lead to a new definition of bankruptcy—if you have to purchase a lot of cheap servers, pretty soon you're talking real money. Such costs force the need to plan for capacity well in advance of the procurement cycle.

Once again, you can determine the capacity doubling period by using elementary tools, like spreadsheets. If, for example, you measure processor utilization U at regularly scheduled intervals, you can estimate long-term consumption by assuming an exponential trend

$$U_{\text{future}} = U_{\text{now}} e^{\lambda W}$$

where growth rate λ is determined by using the Add Trendline facility in Excel, and W is the number of weeks over which to fit the data. Given this relationship, the doubling time is

$$T_{\text{double}} = (\ln 2) / \lambda$$

I chose an exponential-growth model because it is the

CAPACITY PLANNING

simplest function that captures the notion of compounded growth. It also reflects superlinear revenue growth models. If you decide to use statistical models, you might want to consider using more robust tools like Minitab or R, mentioned in the “Guerrilla Tools” sidebar.

The next task is to translate these trends into procurement requirements. Because trend lines pertain only to measurements from the current system configuration, you need a way to extrapolate to other possible configurations. For this purpose, I used the scalability function discussed in Example 1.

As I hope these examples demonstrate, guerrilla capacity planning provides an approach to assessing application scalability that matches management’s requirement to keep a tight rein on project schedules.

In many situations where managers tend to avoid traditional capacity planning, the guerrilla approach can provide a simple framework to bring disparate groups together and reveal unanticipated performance issues. Once revealed, these issues are addressable within the context of existing schedules. In this way, guerrilla capacity planning helps keep projects on schedule and minimizes revisions. Think of it as a way of managing hidden time

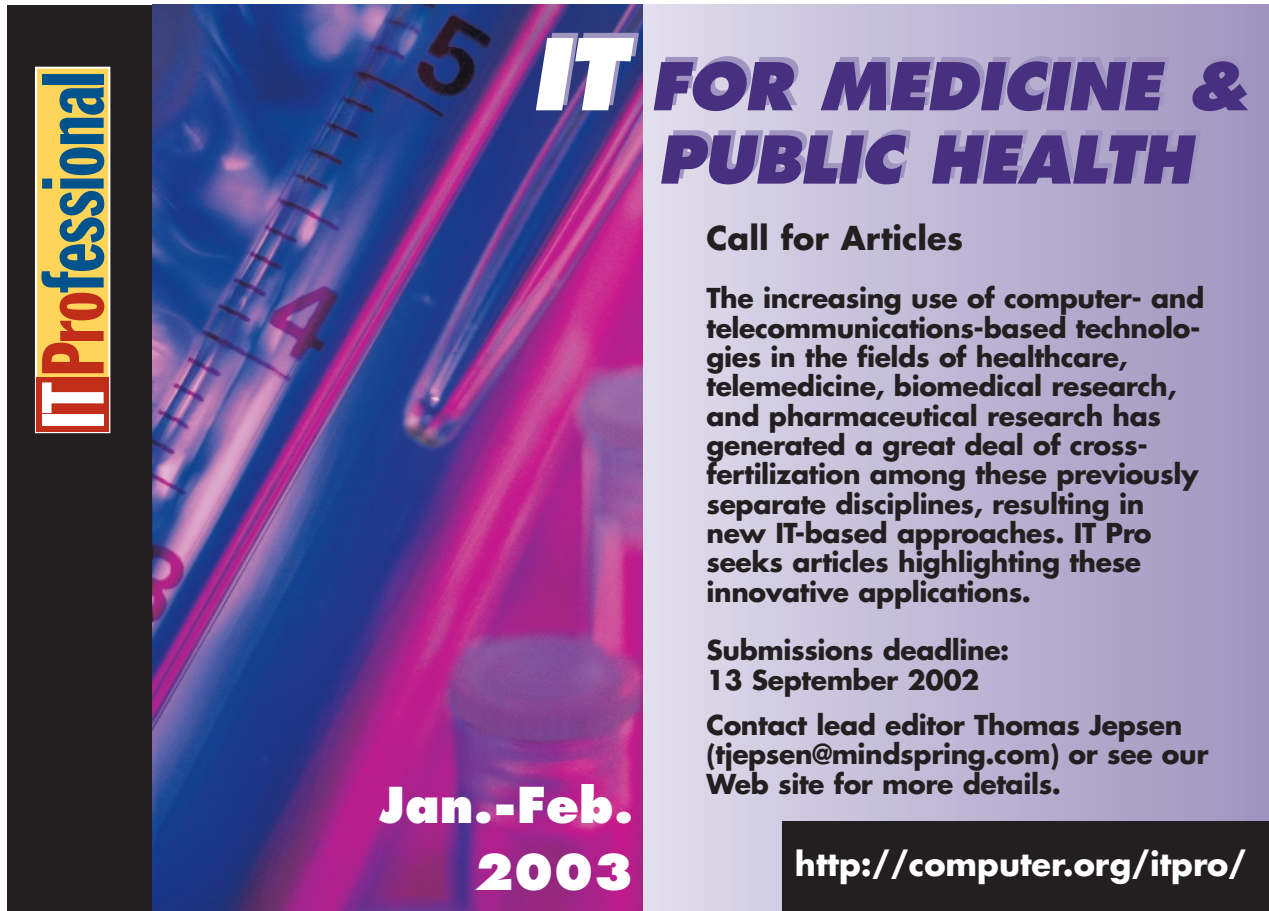
sinks. It’s also a way of replacing risk perceptions with risk management. Sometimes, the biggest hurdle preventing the introduction of guerrilla capacity planning is simply getting started.

Another way of defining guerrilla capacity planning is to see it from the viewpoint of those of us who do it:

- Management plans; we review.
- Management decides; we measure.
- Management dithers; we propose.
- Management revises; we plan.

Aspects of capacity planning that I have not discussed here include floor space, power, cooling, disk storage, tape storage, and so on. You can address many of these issues with spreadsheet models similar to those presented here. I encourage you to consider adopting guerrilla capacity planning in your organization. ■

Neil J. Gunther is chief scientist at Performance Dynamics Consulting in Castro Valley, Calif.; <http://www.perfdynamics.com>. Contact him at njgunther@perfdynamics.com.



IT Professional

IT FOR MEDICINE & PUBLIC HEALTH

Call for Articles

The increasing use of computer- and telecommunications-based technologies in the fields of healthcare, telemedicine, biomedical research, and pharmaceutical research has generated a great deal of cross-fertilization among these previously separate disciplines, resulting in new IT-based approaches. IT Pro seeks articles highlighting these innovative applications.

Submissions deadline:
13 September 2002

Contact lead editor Thomas Jepsen (tjepsen@mindspring.com) or see our Web site for more details.

Jan.-Feb. 2003

<http://computer.org/itpro/>