Carnegie Mellon
Software Engineering Institute

news @ sei interactive

Volume 2 | Issue 3 | Spring 2000

In This Issue:

http://interactive.sei.cmu.edu

news@sei interactive

From the Director

Stephen E. Cross

The 13th annual Software Engineering Process Group (SEPG) Conference held in Seattle, WA in March again demonstrated the extent to which the Capability Maturity Model® for Software (SW-CMM®) has become a standard for assessing and improving software processes. At the SEPG, 2134 attendees—the most ever to attend an SEPG Conference—met to discuss and share their experiences in applying the SW-CMM, which was created at the SEI by SEI Fellow Watts S. Humphrey in 1987.

At the SEI, we are proud to celebrate Watts Humphrey's profound impact on the field of software engineering. Every day we see numerous examples of the broad influence of the concepts that Watts pioneered.

The Boeing Company presented Watts and the SEI Software Process Program with awards at the conference for leadership and innovation in software process improvement. In February, Watts attended an inauguration ceremony in Chennai, India, for a new software institute bearing his name, the Watts S. Humphrey Software Quality Institute (see http://www.watts-sqi.org). The March 1, 2000, issue of Business Week published a Newsmaker Q&A interview of Humphrey, entitled "The Guru of Zero-Defect Software Speaks Out" (http://www.businessweek.com/bwdaily/dn?ash/mar2000/nf00301b.htm). Watts was also recently chosen as one of the 10 people who have made the most significant contributions to the software industry by the managing editor of *CrossTalk* magazine, in an article published in the December 1999 issue.

Since coming to Carnegie Mellon, Watts has pursued a mission of radically improving the way software is developed, world-wide. He continues to successfully pursue that mission in ever-expanding ways and in ever-expanding circles. We are proud of Watts's accomplishments, and we celebrate his impact.

In this issue, we offer an example of this impact. "Putting the Team Software Process^{SM} into Practice" presents TSP results and SEI plans for encouraging organizations to adopt TSP. Also in this issue, "Component-Based Systems" describes a new book by SEI staff members. In "Countering the Threat of Internet Denial of Service Attacks," we describe the SEI's efforts to help organizations recognize and recover from the kinds of attacks that recently affected several large e-commerce sites. Finally, "Probing Product Line Practices" describes a diagnostic tool for examining an organization's readiness to adopt the product line approach.

Thanks for reading, and please keep letting us know how well we are meeting your needs by sending your comments and suggestions to news-editor@sei.cmu.edu.

Putting the Team Software Process into Practice

Bob Lang

Effective teamwork is essential for most software projects. SEI Fellow Watts Humphrey notes that "the scale of modern systems is such, and the demand for short schedules so great, that there is simply not time for one person to do the entire job. Software development is thus a team activity, and the effectiveness of this teamwork largely determines the quality of the engineer's work."[1]

The Team Software Process^{SM} (TSP^{SM}) builds on the foundation of the Capability Maturity Model® (CMM®) and the discipline of the Personal Software Process^{SM} (PSP^{SM}) to help organizations form and manage high performance teams. The CMM defines five levels of process maturity that organizations use as a basis for appraisal and improvement. The PSP provides guidance on how individual engineers can continually improve their performance. The TSP helps PSP-trained engineers work as effective team members as part of a high performance team. Optimum team performance is obtained by enlisting all team members in the project, engaging them in project planning and quality management, and motivating team members to perform to the best of their ability.

TSP Results

Previously in *news@sei* (volume 2, number 1), we reported on the results of a TSP pilot project at Hill Air Force Base, a CMM level 5 organization. The product that the project produced, consisting of more than 25,000 lines of code, was delivered a month ahead of schedule and at almost exactly the planned cost. During system and operational tests, the development team found only one high-priority defect and reduced testing time from 22% to 2.7% of the project schedule. Since then, four organizations have presented data at SEI conferences and have shown similar results.

Each organization reported on effort estimation error, schedule deviation, defects found in system test, defects found in the field, and time to system test 1000 lines of code. A summary of the range of results is shown in Figure 1. Jim Over of the SEI says, "after the introduction of TSP, there was a graphic difference. The performance of the worst case organization using TSP was as good as the best case performance before; and the range is a lot tighter." The four organizations represent four different CMM levels: 1, 2, 3, and 5. "It's clear," says Over, "that the performance of these teams using TSP is consistent with what you'd expect to see at level 5, regardless of what level the organization is."

[1] 1 Humphrey, Watts, S. Leading and Coaching Disciplined Teams: The Team Software Process. Reading, MA: Addison-Wesley, forthcoming.

Putting TSP into Practice

These data indicate that many more organizations could also benefit from TSP. To help more organizations adopt TSP, the SEI will be offering a TSP training course and authorization program. The course will train participants to become TSP launch coaches. The authorization program is the process whereby individuals trained as launch coaches become experienced and authorized to lead TSP launches. The authorization program also provides organizations that have authorized launch coaches access to the TSP materials: the TSP process, supporting forms, the launch workshop materials, and support tools to manage the data that is part of TSP. The 4 1/2-day class will include a mix of lectures and role play exercises that simulate the launch process. Attendees must be authorized PSP instructors. A course schedule will be announced in the fourth quarter of 2000.

Becoming an authorized launch coach involves taking the SEI training course, observing a TSP launch, and being observed leading a TSP launch. The principal role of the launch coach is to show teams how to use the TSP to do quality work. For example, it is the responsibility of the launch coach to ensure that

- the team has adequate resources, training, and support
- the team has realistic goals and a detailed plan to meet the goals
- team morale is good and the team is committed to doing the job

Launching a TSP Project

The TSP launch is a three- to four-day workshop integrated into each of the four typical phases of a project (requirements; high-level design; implementation; and integration and test). The workshop accelerates team building; the team establishes a common understanding of the work and the approach that the team will use; team members make a plan that they can commit to and obtain management support for their plan.

But note that the launch is more than the process of building a plan. Humphrey writes, "While the launch process ostensibly produces the team plan, the principal launch objective is to establish the conditions needed for the team to gel... . If the team produces all the required launch products but has not become a cohesive unit, the launch will not have been successful.... . Teambuilding is a non-trivial process and few people have the time or the skill to devise the specific teambuilding steps needed to build teams in a complex and high-pressure engineering environment. This is what the TSP is designed to do."

For more information contact-

Customer Relations
Phone
412 / 268-5800
Email
customer-relations@sei.cmu.edu
World Wide Web
http://www.sei.cmu.edu/tsp/

Component-Based Systems

Bill Thomas

The market regime, with its new component-based software development processes, is overthrowing the custom-system regime, with its ideal of the "software factory" in which "software processes are analogous to manufacturing processes, programmers are analogous to assembly-line workers, and the ultimate product is lines of code." That is the thesis of the upcoming book, *Component-Based Software Engineering*, part of the Addison-Wesley SEI Series in Software Engineering, by SEI technical staff members Kurt Wallnau, Scott Hissam, and Robert Seacord.

The "software factory" made sense in the era of large-scale, custom-built systems, but is no longer feasible now that the demand for new software far outstrips its supply, jobs for computer and data processing workers go unfilled, and the number of graduates in computer science fields declines. The response is to build systems using commercial software components, and cede control to the new "market regime," and a new set of software development processes. "Components represent a kind of revolution, and revolutions are inherently cyclical," Wallnau says. "We must overthrow our 'traditional' ideas of software process-based on the factory metaphor-with a 'new' process regime."

The market regime consists of component producers and consumers, each behaving, in the aggregate, according to the laws of the marketplace, the authors write. "The market regime decides how features are distributed across components, which features become available when, and how long the features are supported. The marketplace also determines the

What is a Component-Based System?

Software components (or simply "components") are software implementations that have two fundamental properties: they are distributed in binary form, and they are sold or licensed by software vendors. These kinds of components are sometimes referred to as commercial off-the-shelf (COTS) components. Examples of components are Web browsers and Web servers, relational and object-oriented database management systems, object request brokers, spreadsheets, enterprise resource management systems, and operating systems. There are different engineering issues involved in the use of each of these types of components. Some are shrink-wrapped and used as-is, while others must be adapted using vendor-supplied tools.

Component-based systems are built substantially from COTS components. Building systems from multiple COTS components introduces novel engineering problems rooted in the heterogeneity and independent evolution of these components.

—*Excerpted from* Component-Based Software Engineering interfaces components support and which standards are adopted. Ultimately, the market regime decides which components thrive, and which are destined for the bit bucket."

Accommodating Changes

To succeed in the new era, software developers must come to terms with the following realities of the market regime:

- The market drives the system architecture, and designers have less control. Today the designer must accommodate marketplace instability. Components come and go and often do not integrate well. If integration can be achieved at all, it often breaks down when new versions of components are introduced.
- Systems must be designed to accommodate new components. These components may be desirable for their innovative features, or necessary in response to environmental pressures, such as a new operating system release. The downside is that new components may not be compatible with each other, or with older components.
- Requirements analysis, design, implementation, and testing must be compressed and co-mingled. Often the only way to know whether a component, or an ensemble of components, will work is to try it. As a result, previously discrete activities are often indistinguishable. To the uninitiated, it may appear that developers bypass requirements analysis and design and head straight for implementation. In fact, the goal is to achieve "just-in-time competency" through highly focused prototyping experiments that are designed to answer precisely stated questions.
- Systems comprise component suppliers, not simply components. This element of the new market regime resembles a traditional manufacturing supply chain, "with suppliers of parts linked to suppliers of subassemblies of parts to suppliers of larger subassemblies," the authors write. But unlike a supply chain for, say, automobile manufacturing, the top-level enterprise does not control the chain and set standards. In component-based software systems, those standards are established by the marketplace.
- Designers must strive to remove sources of misfit among components. This misfit occurs on two levels: misfit between the user's needs and the available components, and misfit among components that cannot be seamlessly integrated. On the first level, the designer must be a mediator and negotiator to close the gaps between user needs and component capabilities. On the second level, the designer must be a "hacker," with the programming and system skills to bridge, wrap, or otherwise cobble together misfitting components.
- Technology competence is a critical design asset. "The once lowly software engineer," the authors write, "consigned to the bleak and unrewarding prospects of the IT [information technology] equivalent to the assembly line, has emerged as a dominant player in the design and construction of modern, component-based information systems." Engineers who have mastered either a technology, such as Web technology or distributed object technology, or a particular vendor's component, must be hired or developed, and then encouraged to sustain their competence.

The Goal: A New Process Regime

After describing these engineering challenges, the authors present a "toolkit" of engineering techniques that can be used to overcome the loss of control inherent in component-based software engineering. The authors' overarching message is that new processes that are predictable and repeatable must be established for component-based software engineering, just as they were established and matured under the custom system regime.

For more information, contact-

Kurt Wallnau
Email
kcw@sei.cmu.edu

or

Customer Relations
Phone
412 / 268-5800
Email
customer-relations@sei.cmu.edu
World Wide Web
http://www.sei.cmu.edu/tsp/

Countering the Threat of Internet Denial of Service Attacks

Katherine Fithen

The work of the SEI's CERT® Coordination Center (CERT/CC) became a focal point of worldwide media attention in the wake of recent denial of service attacks on some of the Internet's largest and most recognizable electronic commerce sites. These attacks, known as distributed denial of service (DDoS) attacks, are not a new phenomenon. The CERT/CC has been tracking DDoS attacks since 1998, and has published guidance about how to deal with them. The recent DDoS attacks nevertheless caused considerable damage, underscoring the importance of the SEI's continuing efforts to disseminate information about preventing, detecting, and recovering from attack.

The Threat of DDoS Attacks

Distributed systems based on the client/server model have become increasingly common. In a DDoS attack, an intruder compromises several machines individually to install software that will be used to launch a coordinated attack on a target machine. In this type of attack, the compromised machines become agents used by the intruder to carry out the attack. For the victim, the impact can be extensive. In a denial of service attack using distributed technology, the attacked system observes simultaneous attacks from all the nodes at once—flooding the network normally used to communicate and trace the attacks and preventing any legitimate traffic from traversing the network.

For so-called "e-commerce" sites, which operate solely on the Web, even a short term loss of service can be costly. Government and Department of Defense sites are at risk as well because they depend increasingly on networked systems and commercially supported networking products.

Coordinated attacks across national boundaries have occurred. The tools and attacks demonstrate that a network that optimizes its technology for speed and reliability at the expense of security may experience neither speed nor reliability, as intruders abuse the network or deny its services. The intruder technology is evolving, and future tools may be more difficult to defeat.

Countering the DDoS Threat

The CERT/CC constantly monitors trends and watches for new attack techniques and tools. The CERT/CC saw distributed denial of service tools as early as 1998. By fall 1999, it was evident that steps needed to be taken to deal with the increasingly sophisticated intruder tools that were being developed. On November 2 - 4, 1999, the CERT/CC invited 30 experts from around the world to address the problem of network attack tools that use distributed systems in increasingly sophisticated ways. During the resulting Distributed-Systems Intruder Tools (DSIT) Workshop, participants discussed a large number of approaches to preventing, detecting, and responding to distributed attacks.

The workshop effectively provided a venue for experts around the world to share experiences, gain a common understanding, and creatively brainstorm possible responses and solutions to this category of attack before the dissemination of the attack tools-and the attacks themselves-became widespread. The outcome of the workshop was a paper, Results of the Distributed-Systems Intruder Tools Workshop.1 This paper explains the threat posed by these intruder tools and provides guidance to managers, network administrators, Internet service providers, and incident response teams for safeguarding systems from this type of malicious activity:

For managers, planning and coordination before an attack are critical to ensuring adequate response when the attack is in progress. Since the attack methodology is complex and there is no single-point solution or "silver bullet," management must realize that resolution and restoration of systems may be time-consuming. Management needs to be aware that systems may be subject at any time to distributed attacks that are extremely difficult to trace or defend against.

Incident response teams (IRTs) should first make sure they follow the standard operating procedures that they have established. Examples of such procedures can be found the in SEI document, Handbook for Computer Security Incident Response Teams (CSIRTs).2 The best step IRTs can take is to raise awareness within their constituencies.

The report also provides guidance to network administrators and Internet service providers on actions they can take immediately, in the near term, and in the long term. These actions are focused on protecting systems from attack, detecting attacks, and reacting to attacks.

The CERT/CC continues to collaborate with the participants who attended the workshop and with an additional group of security experts to address the ongoing problem.

Looking Forward

The tremendous interconnectedness and interdependency among computer systems on the Internet is not likely to disappear anytime soon. As a result, the security of each system on the Internet depends on the security of all other systems on the network. Recent attacks such as the latest DDoS attacks clearly demonstrate this interdependency. Any computer system can be a victim of a DDoS attack, and there is little system owners can do beyond depending upon others to protect their systems from being used as a launch site in a DDoS attack. To address these and other security problems on the Internet, the entire Internet community must continue to work together.

For more information contact-

Customer Relations
Phone
412 / 268-5800
Email
customer-relations@sei.cmu.edu
World Wide Web
http://www.sei.cmu.edu/tsp/

Internet Denial of Service Attacks and the Federal Response

Katherine T. Fithen

Editor's Note: This column is adapted from testimony by Katherine Fithen, director of the CERT® Coordination Center at the SEI, before the Subcommittee on Crime of the House Committee on the Judiciary and the Subcommittee on Criminal Justice Oversight, of the Senate Committee on the Judiciary, February 29, 2000. This testimony is also available at <http://www.cert.org/congressional_testimony/Fithen_testimony_Feb29.html>.

Introduction

This article describes distributed denial-of-service attacks (DDoS), the role that the CERT® Coordination Center (CERT/CC) has played, and continues to play, in countering these attacks, and what the future holds. The CERT/CC is part of the Networked Systems Survivability Program of the SEI, and was established in 1988, after an Internet "worm" stopped 10% of the computers connected to the Internet. Since then, the CERT/CC staff have handled well over 30,000 incidents and analyzed more than 1,600 computer vulnerabilities. This article is based on that first-hand experience. (For more on the CERT/CC please see <http://www.cert.org>, as well as a feature on the CERT/CC and network security in *SEI Interactive* at <http://interactive.sei.cmu.edu/Features/1998/December/Introduction/intro_dec98.htm.)

Contents:

Distributed Denial-of-Service Tools

Distributed systems based on the client/server model have become increasingly common. In recent months, there has been an increase in the development and use of distributed network sniffers, scanners, and denial-of-service tools. Attacks using these tools can involve a large number of sites simultaneously and be focused to attack one or more victim hosts or networks.

Damaged systems include those used in the attack as well as the targeted victim. For the victim, the impact can be extensive. For example, in a denial-of-service attack using distributed technology, the attacked system observes simultaneous attacks from all the nodes at once—flooding the network normally used to communicate and trace the attacks and preventing any legitimate traffic from traversing the network.

There are indications that the processes for discovering vulnerable sites, compromising them, installing daemons (programs used in the attack), and concealing the intrusion are largely automated, with each step being performed in “batch” mode against many machines in one “session.” Attack daemons have been discovered on a variety of operating systems with varying levels of security and system management.

It is critical to plan and coordinate before an attack to ensure an adequate response when an attack actually happens. Since the attack methodology is complex and there is no single-point solution or “silver bullet,” resolution and restoration of systems may be time-consuming. The bottom line is that an organization’s systems may be subject at any time to distributed attacks that are extremely difficult to trace or defend against. Only partial solutions are available.

Although an organization may be able to “harden” its own systems to help prevent having its systems used as part of a distributed attack, there is essentially nothing a site can do with currently available technology to prevent becoming a victim of, for example, a coordinated network flood. The impact upon the site and its operations is dictated by the (in)security of other sites and the ability of a remote attacker to implant the tools and, subsequently, to control and direct multiple systems worldwide to launch an attack. The result may be reduced or unavailable network connectivity for extended periods of time, possibly days or even weeks depending upon the number of sites attacking and the number of possible attack networks that could be activated in parallel or sequentially.

Coordinated attacks across national boundaries have occurred. The tools and attacks demonstrate that a network that optimizes its technology for speed and reliability at the expense of security may experience neither speed nor reliability, as intruders abuse the network or deny its services. The intruder technology is evolving, and future tools may be more difficult to defeat.

Here are key points to note about distributed-system denial-of-service (DDoS) tools:

- Intruders compromise systems through other means and install DDoS tools.
- The DDoS tools often are equipped with a variety of different attack types.

- Computers that are compromised with DDoS tools are aggregated into networks.
- These networks act in unison to attack a single victim. Any computer on the Internet can be a victim.
- The networks can be activated remotely at a later date by a “master” computer.
- Communication between the master computer and the networks can be encrypted and obfuscated to make it very difficult to locate the master.
- Once activated, the tools typically proceed on their own. No further communication is necessary on the part of the intruder—and it is not possible to discover the master by tracing an ongoing attack. However, there may be evidence on one or more of the machines in the DDoS network regarding the true location of the master.
- Attacks from the network to the victim typically employ techniques designed to obfuscate the true location of the machines in the DDoS network. This makes it difficult to recognize the traffic (and thus block it), to trace the traffic back from the victim to the nodes in the network, and to analyze an attack while it is in progress.
- There are no proactive technical steps an organization can take to prevent it from becoming a victim. Everyone’s security is intertwined. However, by preparing a response in advance, sites can significantly diminish the impact. For information on preparing to respond to these attacks, see the report on the results of a workshop that the CERT Coordination Center organized in November 1999 to address the imminent threat posed by the tools:

  http://www.cert.org/reports/dsit_workshop-final.html

- The tools are rapidly evolving but have not reached their full potential by any means.
- The magnitude of the attacks can overwhelm even the largest networks.
- Intruders are building networks of machines used in these attacks ranging in size from tens to hundreds of machines. It is likely that some networks are much larger.
- The individual nodes in the network can be automatically updated by the master machines, enabling rapid evolution of tools on an existing base of compromised machines.
- A variety of tools are available to detect DDoS tools. Each of these tools has weaknesses, and none is a general-purpose solution. Some of these tools can be found at

  http://www.fbi.gov/nipc/trinoo.htm

  http://staff.washington.edu/dittrich/misc/stacheldraht.analysis

  http://www.iss.net/cgi-bin/dbt-
display.exe/db_data/press_rel/release/122899199.plt

http://www.sans.org/y2k/stacheldraht.htm

- Currently, there is a nearly inexhaustible supply of computers with well-known vulnerabilities that intruders can compromise and install DDoS tools. Additionally, many networks are configured in a way that facilitates the obfuscation techniques used by intruders to conceal their identities. Information about how to configure networks properly is available at

  http://info.internet.isi.edu:80/in-notes/rfc/files/rfc2267.txt

- An archive of DDoS tools can be found at

  http://packetstorm.securify.com/distributed/

- The CERT/CC published advisories and other documents about this topic:

  http://www.cert.org/advisories/CA-2000-01.html

  http://www.cert.org/advisories/CA-99-17-denial-of-service-tools.html

  http://www.cert.org/tech_tips/denial_of_service.html

Role of the CERT/CC in Distributed Denial-of-Service Attacks

The CERT Coordination Center constantly monitors trends and watches for new attack techniques and tools. We began seeing distributed denial-of-service tools in early 1998. Denial-of-service attacks are not new. (See, for example, CERT advisories CA-96.21 <http://www.cert.org/advisories/CA-96.01.UDP_service_denial.html> on TCP “syn” flooding and CA-98.01 <http://www.cert.org/advisories/CA-98.01.smurf.html>on “smurf” attacks, as well as a “tech tip” <http://www.cert.org/tech_tips/denial_of_service.html> on denial-of-service attacks, which the CERT/CC wrote for system administrators in 1997.)

By fall 1999, it was evident that steps needed to be taken to deal with increasingly sophisticated intruder tools before they—and attacks using them—became widespread. On November 2-4, 1999, the CERT/CC invited 30 experts from around the world to address the problem of network attack tools that use distributed systems in increasingly sophisticated ways. During the Distributed-Systems Intruder Tools (DSIT) Workshop, participants discussed a large number of approaches to preventing, detecting, and responding to distributed attacks. The CERT/CC invited personnel who could contribute technically to the solutions regardless of their position in their home organization or their “political” stature in the community. Thus, the workshop effectively provided a venue for experts around the world to share experiences, gain a common understanding, and creatively brainstorm possible responses and solutions to this category of attack before the dissemination of the attack tools and the attacks themselves became widespread. A paper, *Results of the Distributed-Systems Intruder Tools Workshop* <http://www.cert.org/reports/dsit_workshop-final.html>, is available on the CERT Web site <http://www.cert.org>. This paper explains the threat posed by these intruder tools and provides suggestions for safeguarding systems from this type of malicious activity.

The CERT/CC continues to collaborate with the participants who attended the workshop and with an additional group of security experts to address the ongoing problem. Earlier this month, Rich Pethia of the CERT/CC, Alan Paller of the SANS Institute, and Gene Spafford of Purdue University, prepared a Consensus Roadmap for Defeating Distributed Denial of Service Attacks for the Partnership for Critical Infrastructure Security. The most current version can be found on the SANS Institute Web site <http://www.sans.org>.

Internet Trends and Factors Affecting Security

The recent attacks against e-commerce sites demonstrate the opportunities that attackers now have because of several Internet trends and related factors:

- The Internet is becoming increasingly complex and dynamic, but among those people connected to the Internet there is a lack of adequate knowledge about the network and about security. The rush to the Internet, coupled with a lack of understanding, is leading to the exposure of sensitive data and risk to safety-critical systems. Misconfigured or outdated operating systems, mail programs, and Web sites result in vulnerabilities that intruders can exploit. Just one naive user with an easy-to-guess password increases an organization’s risk.
- When vendors release patches or upgrades to solve security problems, organizations’ systems often are not upgraded. The job may be too time-consuming, too complex, or just at too low a priority for the system administration staff to handle. With increased complexity comes the introduction of more vulnerabilities, so solutions do not solve problems for the long term and system maintenance is never-ending. Because managers do not fully understand the risks, they neither give security a high enough priority nor assign adequate resources.
- Attack technology is developing in an open-source environment and is evolving rapidly. Technology producers, system administrators, and users are improving their ability to react to emerging problems, but they are behind and significant damage to systems and infrastructure can occur before effective defenses can be implemented. As long as defensive strategies are reactionary, this situation will worsen.
- Currently, there are tens of thousands—perhaps even millions—of systems with weak security connected to the Internet. Attackers are compromising these machines and building attack networks (and will continue to do so). Attack technology takes advantage of the power of the Internet to exploit its own weaknesses and overcome defenses.

- Increasingly complex software is being written by programmers who have no training in writing secure code and who are working in organizations that sacrifice the safety of their clients for speed to market. This complex software is then being deployed in security-critical environments and applications, to the detriment of all users.
- User demand for new software features instead of safety, coupled with industry response to that demand, has resulted in software that is increasingly supportive of subversion, computer viruses, data theft, and other malicious acts.
- Because of the scope and variety of the Internet, changing any particular piece of technology usually cannot eliminate newly emerging problems; broad community action is required. While point solutions can help dampen the effects of attacks, robust solutions will come only with concentrated effort over several years.
- The explosion in the use of the Internet is straining our scarce technical talent. The average level of system-administrator technical competence has decreased dramatically in the last five years as non-technical people are pressed into service as system administrators. Additionally, there has been little organized support of higher education programs that can train and produce new scientists and educators with meaningful experience and expertise in this emerging discipline.
- The evolution of attack technology and the deployment of attack tools transcend geography and national boundaries. Solutions must be international in scope.
- The difficulty of criminal investigation of cyber crime coupled with the complexity of international law means that successful apprehension and prosecution of computer criminals is unlikely, and thus little deterrent value is realized.
- The number of directly connected homes, schools, libraries and other venues without trained system administration and security staff is rapidly increasing. These “always-on, rarely protected” systems allow attackers to continue to add new systems to their arsenal of captured weapons.

Near-Term Actions to Reduce Risk and Dampen the Effects of Attacks

The problem of distributed denial-of-service attacks is complex, and there are no easy answers. The *Results of the Distributed-System Intruder Tools Workshop* paper <http://www.cert.org/reports/dsit_workshop-final.html> contains specific steps that can be taken by managers, system administrators, Internet service providers, and computer security incident response teams. The *Consensus Roadmap for Defeating Distributed Denial of Service Attacks* <http://www.sans.org/ddos_roadmap.htm>, contains additional recommendations. The recommendations below, divided into four key problem areas, can be found in the *Consensus Roadmap*.

Solutions to mitigate “spoofing”

Attackers often hide the identity of machines used to carry out an attack by falsifying the source address of the network communication. This makes it more difficult to identify the sources of attack traffic and sometimes shifts attention onto innocent third parties. Limiting the ability of an attacker to spoof IP source addresses will not stop attacks, but will dramatically shorten the time needed to trace an attack back to its origins.

- User organizations and Internet service providers can
    - ensure that traffic exiting an organization’s site, or entering an ISP’s network from a site, carries a source address consistent with the set of addresses for that site, and
    - ensure that no traffic from “unroutable addresses” listed in RFC 1918 are sent from their sites. This activity is often called egress filtering.
- ISPs can provide backup to pick up spoofed traffic that is not caught by user filters. ISPs may also be able to stop spoofing by accepting traffic (and passing it along) only if it comes from authorized sources. This activity is often called ingress filtering.
- Dial-up users are the source of some attacks. Putting an end to spoofing by these users is also an important step.
- ISPs, universities, libraries, and others that serve dial-up users should ensure that proper filters are in place to prevent dial-up connections from using spoofed addresses.
- Network equipment vendors should ensure that “no-IP-spoofing” is a user setting, and the default setting, on their dial-up equipment.

Solutions to help stop broadcast amplification

In a common attack, the malicious user generates packets with a source address of the site he or she wishes to attack (site A, using spoofing) and then sends a series of network packets to an organization with many computers (site B), using an address that broadcasts the packets to every machine at site B. Unless precautions have been taken, every machine at site B will respond to the packets and send data to the organization (site A) that was the target of the attack. The target will be flooded and people at site A may blame the people at site B. Attacks of this type often are referred to as Smurf attacks. In addition, the echo and chargen services can be used to create oscillation attacks similar in effect to Smurf attacks. Solutions include the following:

- Unless an organization is aware of a legitimate need to support broadcast or multicast traffic within its environment, the forwarding of directed broadcasts should be turned off. Even when broadcast applications are legitimate, an organization should block certain types of traffic sent to "broadcast" addresses (e.g., ICMP Echo Reply) messages so that its systems cannot be used to effect Smurf attacks.
- Network hardware vendors should ensure that routers can turn off the forwarding of IP-directed broadcast packets as described in RFC 2644 and that this is the default configuration of every router.

- Users should turn off echo and chargen services unless they have a specific need for those services. (This is good advice, in general, for all network services: they should be disabled unless they are known to be needed.)

Solutions that encourage appropriate responses to attacks

Many organizations do not respond to complaints of attacks originating from their sites or to attacks against their sites, or they respond in a haphazard manner. This makes containment and eradication of attacks difficult. Further, many organizations fail to share information about attacks, giving the attacker community the advantage of better intelligence sharing.

- User organizations should establish incident-response policies and teams with clearly defined responsibilities and procedures.
- ISPs should establish methods of responding quickly if they discover that their systems were used for attacks on other organizations. They must also have enough staffing to support these efforts.
- User organizations should encourage system administrators to participate in industry-wide early warning systems, where their corporate identities can be protected (if necessary), to counter rapid dissemination of information within the attack community.
- Attacks and system flaws should be reported to appropriate authorities (e.g., vendors, response teams) so that the information can be applied to defenses for other users.

Solutions to protect unprotected computers

Many computers are vulnerable to takeover for distributed denial-of-service attacks because of inadequate implementation of well-known “best practices.” When those computers are used in attacks, the result can be major costs, headaches, and embarrassment for the owners of the computers being attacked. Furthermore, once a computer has been compromised, its data may be copied, altered, or destroyed, its programs changed, and the system disabled. Solutions include the following:

- User organizations should check their systems periodically to determine whether they have had malicious software installed, including DDoS Trojan horse programs. If such software is found, the system should be restored to a known good state.
- User organizations should reduce the vulnerability of their systems by installing firewalls with rule sets that tightly limit transmission across the site’s periphery (e.g., deny traffic, both incoming and outgoing, unless given specific instructions to allow it).
- All machines, routers, and other Internet-accessible equipment should be periodically checked to verify that all recommended security patches have been installed.

- The security community should maintain and publicize lists of the "Top 20 Exploited Vulnerabilities" and "Top 20 Attacks," which include the current most-often-exploited vulnerabilities. This would help system administrators set priorities.
- Users should turn off services that are not required and limit access to vulnerable management services (e.g., RPC-based services).
- Users and vendors should cooperate to create "system-hardening" scripts that can be used by less sophisticated users to close known holes and tighten settings to make their systems more secure. Users should employ these tools when they are available.
- System software vendors should ship systems where security defaults are set to the highest level of security rather than the lowest level of security. These "secure out-of-the-box" configurations will greatly aid novice users and system administrators. They will furthermore save critically scarce time for even the most experienced security professionals.
- System administrators should deploy "best practice" tools including firewalls (as described above), intrusion-detection systems, virus-detection software, and software to detect unauthorized changes to files. Use of software to detect unauthorized changes may also be helpful in restoring compromised systems to normal function.
- System and network administrators should be given time and support for training and enhancement of their skills. System administrators and auditors should be periodically certified to verify that their security knowledge and skills are current.

Prognosis for the Future

In spite of preparation and protective measures that can be taken now, the problem of conquering DDoS attacks requires a long-term effort to define and implement effective solutions. The Consensus Roadmap for Defeating Distributed Denial of Service Attacks identifies the following actions that should be considered:

- Establish load and traffic volume monitoring at ISPs to provide early warning of attacks.
- Accelerate the adoption of the IPsec components of Internet Protocol Version 6 and the Secure Domain Name System.
- Increase the emphasis on security in the research and development of Internet II.
- Support the development of tools that automatically generate router access control lists for firewall and router policy.
- Encourage the development of software and hardware that is engineered for safety with possibly vulnerable settings and services turned off, and encourage vendors to automate security updating for their clients.
- Sponsor research in network protocols and infrastructure to implement real-time flow analysis and flow control.

- Encourage wider adoption of routers and switches that can perform sophisticated filtering with minimal performance degradation.
- Sponsor continuing topological studies of the Internet to understand the nature of “choke points.”
- Test deployment and continue research in anomaly-based and other forms of intrusion detection.
- Support community-wide consensus of uniform security policies to protect systems and to outline security responsibilities of network operators, Internet service providers, and Internet users.
- Encourage development and deployment of a secure communications infrastructure that can be used by network operators and Internet service providers to enable real-time collaboration when dealing with attacks.
- Sponsor research and development leading to safer operating systems that are also easier to maintain and manage.
- Sponsor research into survivable systems that are better able to resist, recognize, and recover from attacks while still providing critical functionality.
- Sponsor research into better forensic tools and methods to trace and apprehend malicious users without forcing the adoption of privacy-invading monitoring.
- Provide meaningful infrastructure support for centers of excellence in information security education and research to produce a new generation of leaders in the field.
- Consider changes in government procurement policy to emphasize security and safety, rather than simply cost, when acquiring information systems, and to hold managers accountable for poor security.

Conclusion

We have discussed for many years the tremendous interconnectedness and interdependency among computer systems on the Internet. As a result, the security of each system on the Internet depends on the security of all other systems on the network. The distributed denial-of-service attacks clearly demonstrate this interdependency. Any computer system can be a victim of a DDoS attack, and there is little that system owners can do beyond depending upon others to protect their systems from being used as a launch site in a DDoS attack. To address the problem of distributed denial-of-service attacks and other security problems on the Internet, we must continue to work together. We must pool our expertise, take steps to protect ourselves, and help others protect themselves.

About the Author

Katherine T. Fithen is manager of the CERT® Coordination Center. She has been a member of the CERT/CC since March 1992. The CERT Coordination Center provides technical assistance to Internet sites that have computer security issues or concerns, or that have experienced a computer security compromise. Prior to joining the CERT/CC, she was a user consultant for PREPnet, the regional network service provider for the state of Pennsylvania. Fithen has earned a bachelor's degree in retail management, a master's degree in personnel management, and a master's degree in information science.

Results of the Distributed-Systems Intruder Tools Workshop

Held November 2-4, 1999, in Pittsburgh, Pennsylvania

Following is the executive summary from the paper published by participants of the Distributed-Systems Intruder Tools Workshop, which was held a few months before a series of distributed denial-of-service attacks interrupted service on several major Web sites. We encourage readers to read the complete paper, which is available at <http://www.cert.org/reports/dsit_workshop-final.html>.

On November 2-4, 1999, the CERT® Coordination Center invited 30 experts from around the world to address a category of network attack tools that use distributed systems. Several tools are in use now, and the technology is maturing. As a result, a single, simple command from an attacker could result in tens of thousands of concurrent attacks on one or a set of targets. The attacker can use unprotected Internet nodes around the world to coordinate the attacks. Each attacking node has limited information on who is initiating the attack and from where; and no node need have a list of all attacking systems. Damaged systems include those used in the attack as well as the targeted victim. For the victim, the impact can be extensive. For example, in a denial-of-service attack using distributed technology, the attacked system observes simultaneous attacks from all the nodes at once—flooding the network normally used to communicate and trace the attacks and preventing any legitimate traffic from traversing the network.

Distributed intruder technology is not entirely new; however, it is maturing to the point that even unsophisticated intruders could do serious damage. The Distributed-Systems Intruder Tools Workshop provided a venue for experts around the world to share experiences, gain a common understanding, and creatively brainstorm possible responses and solutions before the dissemination of the maturing attack tools—and attacks themselves—become widespread.

One consideration is the approach typically taken by the intruder community. There is (loosely) organized development in the intruder community, with only a few months elapsing between “beta” software and active use in attacks. Moreover, intruders take an open-source approach to development. One can draw parallels with open system development: there are many developers and a large, reusable code base. Intruder tools become increasingly sophisticated and also become increasingly user friendly and widely available. As a result, even unsophisticated intruders can use them.

There has already been some public discussion in the intruder community about distributed attack tools while development continues. In their development, intruders are using currently available technology to develop new technology. For example, they are building on previous scanning technology and automated intrusion tools to create more powerful intrusion tools. One concern of workshop participants is that in a relatively short time, it may be possible for unsophisticated intruders to gain control of and use systems distributed across significant portions of the Internet for their attacks.

This paper is one outcome of the Distributed-Systems Intruder Tools Workshop. In it, workshop participants examine the use of distributed-system intruder tools and note that current experiences have highlighted the need for better forensic techniques and training, the importance of close cooperation, and a concern for the rapid evolution of intruder tools. They provide information about protecting systems from attack by the tools, detecting the use of the tools, and responding to attacks. The paper includes suggestions for specific groups in the Internet community: managers, system administrators, Internet service providers (ISPs), and incident response teams (IRTs).

The suggestions address actions each group should take immediately, along with actions for the short term and long term. They also remind readers that the security of any network on the Internet depends on the security of every other network. The widely varying implementation of security measures is what often makes a distributed attack successful.

The workshop participants hope that the information offered here will help reduce the impact of distributed attack tools on the Internet as those tools mature.

The complete report is available at

<http://www.cert.org/reports/dsit_workshop-final.html>

Probing Product Line Practices

Claire Dixon

Increasingly, organizations are moving toward the product line approach to software development and reaping major benefits. The strategy of deriving an entire set of products, to satisfy a particular market need, from a common set of building blocks is an idea whose time has come. However, the product line approach also involves unique risks and costs. A company's awareness and preparedness can mean the difference between a sustained success and a waning effort. The SEI has developed the Product Line Technical Probe (PLTP)—a diagnostic tool for examining an organization's readiness to adopt the product line approach, or to proceed effectively with its existing product line effort.

The PLTP is based on the principles described in A Framework for Software Product Line Practice–Version 2.0.1 Released by the SEI in 1999, the framework is an evolving Web-based document that describes the essential concepts, practice areas, and activities involved in successful development or acquisition of a software product line. It provides organizations with the ingredients of an integrated business and technical approach—the "product line practice approach"—that allows them to produce and maintain similar systems of predictable quality, at lower costs, and in significantly shorter time. The PLTP examines how well an organization measures up against the Product Line Practice (PLP) Framework.

The SEI has spent more than three years creating the PLP Framework. Its contents were developed from a combination of in-depth studies of organizations that build product lines; direct collaborations with industry and Department of Defense organizations on product line efforts; and workshops involving participants from the product line commercial leaders. Larry Jones of the SEI feels that the probe is the natural outcome of the framework development, and is valuable to organizations that are trying to adopt its practices. "When you have a model of best practices, it makes sense to compare yourself to that model to see how you might improve."

The PLTP is intended to benefit two groups of organizations: 1) those that are considering adopting a product line approach and 2) those that have already initiated a product line effort. Organizations that are candidates for the PLTP might be asking the questions, "Will this endeavor prove profitable to our company?" "This seems like it will be worth doing but how ready are we?" or "We've hit an impasse in our implementation; how do we proceed to assure a positive outcome?"

The result of the probe is a set of findings that portray an organization's strengths and challenges relative to its product line endeavors. These findings can then be used as a basis for improvement planning, with the goal of increasing the organization's capability for product line success.

The Probe Process

Preparation

During the preparation phase of the probe, the PLTP team communicates extensively with the organization executive who has commissioned the probe—the "organization sponsor." Team members discuss the organization's experience with the product line approach, its goals regarding the approach, and its objectives and expectations for the PLTP.

Data Gathering

The data gathering phase follows, featuring structured interviews with small groups of organization members. Open communication is crucial; groups are made up of peers, so that none report to each other in the organization. All comments are non-attributable. The PLTP team poses to each peer group a set of questions, based on the practice areas described in the SEI PLP Framework. These practice areas are loosely organized into three categories: software engineering, technical management, and organizational management. While the practice areas under consideration may be required for any software system, the product line context imposes its own set of constraints that must be specifically addressed. The scope of the question set is determined during the preparation phase. Different questions from this set may be posed based upon the job responsibilities of particular interview groups. The team may probe more deeply into certain areas to learn about specific strengths and challenges. The team may also request documentation to support the interview data-gathering process.

Data Reduction and Analysis

The PLTP team consolidates and compares the data gathered against practice areas documented in the framework, and classifies findings as strengths and challenges based upon that comparison. Findings may also include information that lies outside of the scope of the framework, if the team determines that the information affects the organization's product line readiness.

Results

The team presents its findings to an audience designated as appropriate by the organizational sponsor. A written report follows. Depending on its arrangement with the organization, the SEI may assist the organization in developing an action plan to address the findings.

The PLTP method is based on years of SEI experience in evaluating organizations' software engineering practices. It gives a penetrating view of how the approach fares at all levels of infrastructure. It detects where technical management is strained, where organizational weakness may develop, and where engineering competencies are exceptionally strong. This perspective provides a map of where reinforcement or revision is required. It is a picture not generally available to decision makers through their typical information-gathering methods. Without the benefit of the probe, a company may not even be aware of a deficit, or may misdiagnose its cause.

The extraordinary potential of a product line approach multiplies the probe's value. The rewards of establishing a whole line of software from a common engineering basis can be tremendous. One company was able to reduce its system build time from a year to one week by adopting a product line approach. Another organization increased its production six-fold over three years. All this is possible, but having the necessary insight and guidance to determine the correct approach is essential. The probe can help provide that insight.

For more information contact-

Customer Relations
Phone
412 / 268-5800
Email
customer-relations@sei.cmu.edu
World Wide Web
http://www.sei.cmu.edu/tsp/

or

Linda Northrop
Email
lmn@sei.cmu.edu
World Wide Web
http://www.sei.cmu/plp/

Quality Attribute Workshops

Mario Barbacci

In large software systems, the achievement of qualities such as performance, availability, security, and modifiability is dependent not only on code-level practices (e.g., language choice, detailed design, algorithms, data structures, and testing), but also on the overall software architecture. The quality attributes of large systems can be highly constrained by a system’s software architecture. Thus, it is in our best interest to try to determine at the time a system’s software architecture is specified whether the system will have the desired qualities.

In previous columns we have written about various components of an emerging "software architecture practice" by drawing analogies to architectural engineering and describing approaches to software architecture representation, quality attributes, and the use of scenarios in architecture evaluations. In this column, I describe one way for combining several of these components into a process that allows early insight into a system’s architecture, including its quality-attribute sensitivities, tradeoffs, and risks.

Making Attribute Goals Concrete

Quality attributes are interdependent. For example, performance affects modifiability, availability affects safety, security affects performance, and everything affects cost. Therefore, achieving one quality attribute can affect the other attributes [Boehm 78]. These side effects reflect dependencies among attributes and can be defined by parameters shared among attribute models. If we can identify these parameters, the results from one analysis can feed into the others.

Quality attributes, such as modifiability, performance, and security, are not definitive enough by themselves either for design or for evaluation. They must be made more concrete. Using modifiability as an example, if a system can be easily adapted to have different user interfaces but is dependent on a particular operating system, is it modifiable? The answer is that it depends on what modifications are expected to the system over its lifetime. That is, the abstract quality of modifiability must be made concrete. The same observation is true for other attributes.

For the past two years the SEI has been developing the Architecture Tradeoff Analysis Method^{SM} (ATAM^{SM}). ATAM is based on a set of attribute-specific measures of a system—some analytic, based on formal models (e.g., performance and availability), and some qualitative, based on formal inspections (e.g., modifiability, safety, and security). We now have a stable process for carrying out ATAM analyses. The process includes a set of steps and a set of reusable architectural styles and analytic models, called attribute-based architectural styles (ABASs), that we use during an ATAM analysis. The ATAM process will be covered in a future issue of *The Architect*, so we will not dwell on the method here.

The effectiveness of ATAM depends on having a concrete, well-defined architecture to be analyzed. However, some ATAM benefits can be achieved even if an architecture is not fully defined. Under some circumstances, an organization might wish to identify potential architecture risks while developing a system's architecture. With the sponsorship of the U.S. Coast Guard, we are testing the concept of a "Quality Attribute Workshop" in which architects, developers, users, maintainers, and other system stakeholders, such as people involved in installation, deployment, logistics, planning, and acquisition, carry out several ATAM steps, but focus on system requirements and quality attributes, rather than on the architecture. The objective of the workshop is to identify sensitivities, tradeoffs, and risks and use these as early warnings to the architecture developers.

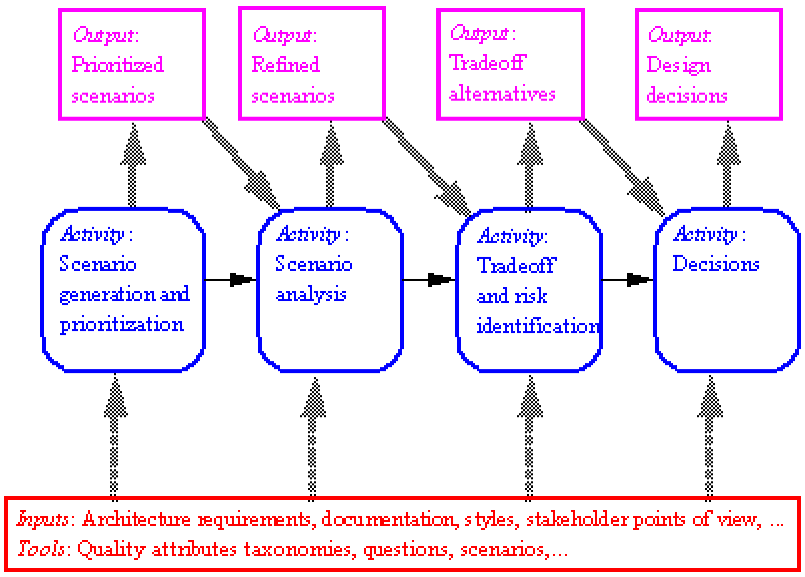
The workshop is intended as a forum for the discussion of quality attributes and their evaluation. The workshop does not aim at an absolute measure of "architecture quality;" rather the purpose is to identify scenarios from the point of view of a diverse group of stakeholders (e.g., the architect, developers, users, sponsors) and to identify risks (e.g., inadequate performance, successful denial-of-service attacks) and possible mitigation strategies (e.g., replication, prototyping, simulation).

Roadmap Activities

Figure 1 illustrates the Quality Attribute Roadmap, the process we use during the workshops to discover and document quality attribute risks, sensitivity points, and tradeoffs in the architecture, where

- *risks* are architecture decisions that might create future problems for some quality attribute requirement
- *sensitivity points* are architecture parameters for which a slight change makes a significant difference in some quality attribute
- *tradeoffs* are architecture parameters affecting more than one quality attribute

Figure 1: Quality Attribute Roadmap


Output: Prioritized scenarios
Output: Refined scenarios
Output: Tradeoff alternatives
Output: Design decisions
Activity: Scenario generation and prioritization
Activity: Scenario analysis
Activity: Tradeoff and risk identification
Activity: Decisions
Inputs: Architecture requirements, documentation, styles, stakeholder points of view, ...
Tools: Quality attributes taxonomies, questions, scenarios,...

During the workshop we conduct several activities aimed at generating various outputs or products:

- Scenario generation takes place during a facilitated brainstorming process; stakeholders propose scenarios that test the effectiveness of a candidate or conceptual architecture to achieve specific quality attributes within a specific Deepwater mission and geographic context. For prioritization, each stakeholder is assigned a number of votes that she can allocate as desired.
- During scenario analysis, for each of the high-priority scenarios, the stakeholders choose an appropriate architectural style or architectural fragment as an artifact for analysis, and apply the scenario to the artifact. The purpose of the analysis is to identify important architecture decisions and sensitivity points. As a result of this activity, the stakeholders might decide to conduct additional, more detailed or formal analyses of the scenarios or artifacts, but these activities take place offline, not during the workshop.
- During tradeoff and risk identification, the stakeholders use the results of the analysis activity to identify and document risks—i.e., potential future problems that might impact cost, schedule, or quality attributes of the system. Scenarios to consider include:
    - a single scenario that involves two attributes explicitly or implicitly
    - multiple scenarios about different attributes sharing common factors (e.g., resources, protocols)
    - multiple contradictory scenarios

We use various sources as inputs for the activities, including:

- architecture documentation
- stakeholder points of view
- architecture styles

The stakeholders generate, prioritize, and analyze the scenarios, and identify tradeoffs and risks from their points of view, depending on the role they play in the development of the system, and their expertise on specific quality attributes. As an additional input source, we try to identify known architectural styles because they can expedite the process. Architecture styles are abstractions such as "client/server," "publish/subscribe," "shared memory," "layered," and "pipe and filter," which can be used as drivers for the analysis because they provide "canned" scenarios, known tradeoffs, and likely risks. The results of the analysis would depend on which architecture styles are used.

Finally, there is a collection of tools and techniques that we use to perform a quality attribute analysis:

- scenarios
- quality attribute tables
- questions

These sometimes have different labels, such as “screening questions” or “exploratory scenarios.” It is important to be precise in our use of terms to ensure that (a) we share the same understanding, and (b) we can decide what tools to use and when to use them. Thus, prior to the workshops, the participants receive a handbook describing the activities and the tools to be used and as a reminder, they are taken through a short presentation at the beginning of the meeting. The rest of this article details some of the tools and the experiences we have had with the workshops.

Scenarios

Scenarios are used to exercise the architecture against current and future situations:

- Use-case scenarios reflect the normal state or operation of the system. If the system is yet to be built, these would be about the initial release.
- Growth scenarios are anticipated changes to the system. These can be about the execution environment (e.g., double the message traffic) or about the development environment (e.g., change message format shown on operator console).
- Exploratory scenarios are extreme changes to the system. These changes are not necessarily anticipated or even desirable situations. Exploratory scenarios are used to explore the boundaries of the architecture (e.g., message traffic grows 100 times, operating system is replaced).

The distinction between growth and exploratory scenarios is system- or situation-dependent. Anticipated growth in a business application might be a disaster in a deep space probe (e.g., 20% growth in message storage per year).

Table 1 shows several representative performance scenarios.

Table 1: Performance Scenarios

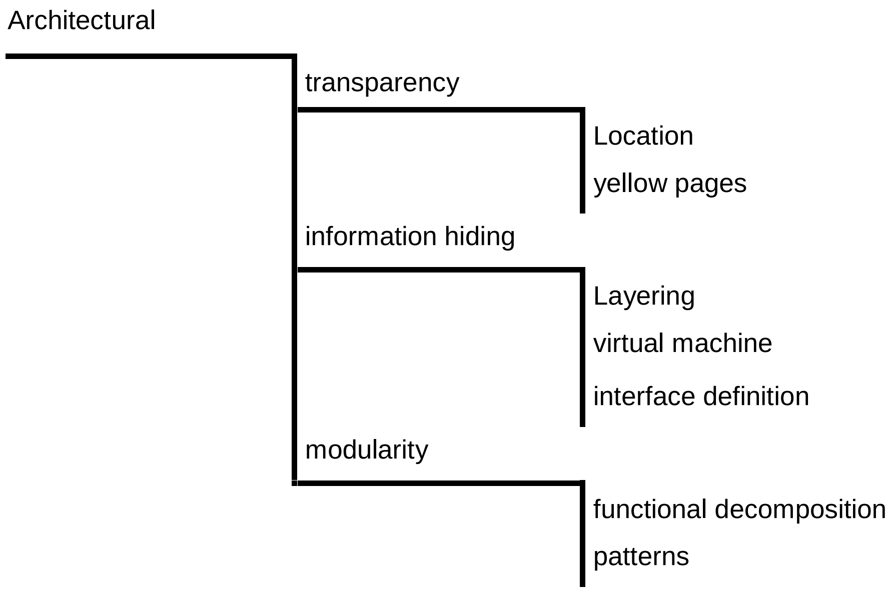
Scenario	**Type**
The communications network is overloaded. Missions are reassigned to reduce traffic.	use case
The LAN is overloaded. Tasks are reassigned to reduce traffic.	use case
The process-to-processor allocation is changed to balance the load.	use case
Data throughput is doubled.	growth
The number of users doubles.	growth
Real-time video data is needed by central office.	exploratory
Important, but not mission-critical, traffic doubles in volume.	growth

There are no clear rules other than stakeholder consensus that some scenarios are likely (desirable or otherwise) and other scenarios are unlikely (but could happen and, if they occurred, it would be useful to understand the consequences).

Quality Attribute Tables

The handbook used in the workshop describes various quality attributes, characterized by stimuli, responses, and architectural decisions that link them. Stimuli and responses are the activities (operational or developmental) that exercise the system and the observable effects, respectively. For example, a stimuli for the "modifiability" attribute could be "change the operating system," and the responses could include "effort to implement" and "number of subsystems affected." The architecture decision in this case might be "use a virtual machine approach." Each attribute is described by stimulus/response/mechanism tables, where the level of detail is appropriate to the state of development and the available documentation. See Figure 2 for an illustration of a table of architecture mechanisms for the modifiability attribute.

Figure 2: Modifiability Architecture Mechanisms



The attribute tables are used only to suggest stimuli, responses, and mechanisms that might be of interest. They are just a reminder of the kinds of issues we want the participants to take into consideration when generating and analyzing scenarios. We could expect that, depending on the interests of the stakeholders, quality attribute tables might be added, removed, refined, or pruned, as the participants see fit.

Questions

We use various types of questions to collect and analyze information about current and future system drivers and architectural solutions.

- Screening questions are used to quickly narrow or focus the scope of the evaluation. They identify what is important to the stakeholders.
    - Screening questions are qualitative; the answers are not necessarily precise or quantifiable. The emphasis is on expediency.
    - Screening questions can be driven by a quality attribute deemed important to some stakeholders. Sometimes the attribute is clear and explicit (e.g., "the service must be continuous" identifies availability and security as the quality attributes of concern). Sometimes the attribute is implied (e.g., "life-cycle cost must be minimal" suggests modifiability and interoperability as the relevant quality attributes).

    - Screening questions can also be driven by a subsystem or a service deemed important to achieve a quality attribute. For example, once an important attribute is identified by the stakeholders, screening questions can be used to narrow or focus on subsets of the architecture that are relevant to achieving the attribute (e.g., the user authentication subsystem, the message filtering and distribution subsystem).

- Elicitation questions are used to gather information to be analyzed later. They identify how a quality attribute or a service is achieved by the system.

    - Elicitation questions collect information about decisions made; the emphasis is on extracting quantifiable data.

    - Elicitation questions can be driven by an attribute model. We ask for quality attribute-specific information when the answer is a parameter of an attribute model (e.g., message arrival rates are parameters in a model of throughput, repair rates are parameters in a Markov model of availability). These elicitation questions are guided by stimulus/response branches of the quality attribute tables.

    - Elicitation questions can also be driven by architecture styles. We ask for architectural information when the answer is important to determine the "quality" of a particular architecture style choice (e.g., individual latencies are required to compute the performance of a pipe-and-filter architecture). These elicitation questions are guided by the architecture mechanism branch of the quality attribute tables.

- Analysis questions are used to conduct analysis using attribute models and information collected by elicitation questions. Analysis questions refine the information gathered by elicitation.

There is an implied ordering in the questions (i.e., screening > elicitation > analysis) although questioning can be carried out in breadth-first or depth-first order:

- Breadth-first questioning first identifies all important attributes and subsets of the architecture. Then, for each one, questioning elicits all the information that will be used later for analysis.

- Depth-first questioning dives deeply into an important attribute or subset of the architecture before other attributes or subsets of the architecture are considered.

Either order can be used, and the decision might be opportunistic. During a discovery or early analysis exercise, breadth-first might be more appropriate; during an evaluation or detailed analysis exercise, depth-first might be more appropriate.

For each quality attribute of interest, attribute-specific example questions serve as seeds for additional questions about stimuli, response, or architecture mechanisms. Table 2 provides an example of a list of specific questions for security.

Table 2: Security Questions

	Question	**Type**
Requirements	What are the trusted entities in the system and how do they communicate?	Screen
Stimulus/ Response	Which essential services could be significantly affected by an attack?	Analysis
	Are there attacks or events that could affect service across the entire integrated system?	Analysis
	Is there a single point from which the entire system is controlled?	Analysis
	For which kind of attacks will recovery be the most difficult?	Analysis
Resistance/ Recovery/ Recognition	How is user authentication and authorization information maintained for employees?	Elicitation
	How is access managed for those people who are outside the network?	Elicitation
	What sensitive information must be protected?	Elicitation
	What approach is used to protect that data?	Elicitation
	Which user actions are logged?	Elicitation
	What kind of monitoring and access controls exist at network boundaries?	Elicitation
	What information is permitted through or filtered out?	Analysis

The questions are not meant to be exhaustive; rather they are meant to serve as starting points and as examples for stakeholders to generate additional questions about the quality attribute requirements and the system.

Scenarios and questions contain the explicit or implied attribute stimulus/response/mechanisms that are deemed important. Scenarios and questions might raise doubts or concerns regarding some aspect of the architecture about which we might have to elicit further information to conduct a more detailed analysis. They serve to identify potential risks, such as what risks can arise from decisions made (e.g., choice of middleware), as well as decisions not yet made (e.g., message encoding).

The generation of scenarios can alternate with the generation of questions. For example, screening questions can identify regions of stimuli/responses as sources of use-case scenarios, which in turn might suggest questions about architecture mechanisms involved in the scenario. For example, a screening question might identify message throughput as important; a scenario about message throughput would identify the components involved in the message path. The capacity or speed of some components might be seen as questionable, prompting further elicitation questions (e.g., time required to process a message or choice of queuing policy).

Experience with Quality Attribute Workshops

We have conducted a handful of workshops, and the process is still evolving. As indicated earlier, the intent of the workshops is to encourage an organization to generate and analyze scenarios about a hypothetical system, not necessarily something under development. However, we need something to analyze the scenario against! For example, if a scenario suggests that message throughput is important, we need a sketch of the components and connections that implement the subsystem that processes the messages. Because no such decisions are expected to have been made at the time of the workshop, when we analyze a scenario, the architect can suggest a reasonable or likely candidate architecture for the purposes of the exercise. The stakeholders are not bound to that solution and are not "graded" on the effectiveness of a choice made on the spur of the moment. However, the scenarios, questions, and attribute tables remain with the organization, and they can repeat the exercise using alternative subsystem architectures.

Further Reading (these references must be further developed before publication)

To learn more about the U.S. Coast Guard Deepwater project and its novel approach acquisition, consult
[Deepwater] United States Coast Guard Deepwater Project
[Freedberg 00] Freedberg, S. Jr. Coast Guard uses new model to procure new fleet. Daily Briefing Column, GovExec.com April 24, 2000

References

[ATAM] The Architecture Tradeoff Analysis (ATA) Method

[Boehm 78] Boehm, B.W., Brown, J.R., Kaspar, H., Lipow, M., MacLeod, G.J. & Merritt, M.J. Characteristics of Software Quality. New York, NY: Elsevier North-Holland Publishing Company, Inc., 1978.
[Workshop Handbook] Quality Attribute Workshop Participants Handbook
[Workshop Slides] Quality Attribute Workshop Slides

About the Author

Mario Barbacci is a senior member of the technical staff at the SEI. He was one of the founders of the SEI, where he has served in several technical and managerial positions, including project leader (Distributed Systems), program director (Real-Time Distributed Systems, Product Attribute Engineering), and associate director (Technology Exploration Department). Before coming to the SEI, he was a member of the faculty in the School of Computer Science at Carnegie Mellon.

Barbacci is a fellow of the Institute of Electrical and Electronic Engineers (IEEE), a member of the Association for Computing Machinery (ACM), and a member of Sigma Xi. He was the founding chairman of the International Federation for Information Processing (IFIP) Working Group 10.2 (Computer Descriptions and Tools) and has served as vice president for technical activities of the IEEE Computer Society and chair of the Joint IEEE Computer Society/ACM Steering Committee for the Establishment of Software Engineering as a Profession. He was the 1996 president of the IEEE Computer Society. He is the 1998-1999 IEEE Division V Director.

Barbacci is the recipient of several IEEE Computer Society Outstanding Contribution Certificates, the ACM Recognition of Service Award, and the IFIP Silver Core Award. Barbacci received bachelor's and engineer's degrees in electrical engineering from the Universidad Nacional de Ingenieria, Lima, Peru, and a doctorate in computer science from Carnegie Mellon.

Just-In-Time Component Competence

Kurt Wallnau

One of the most positive developments in software process thinking has been "personalization" in the Personal Software Process^{SM} (PSP^{SM}). PSP is concerned with building competence in a fundamental skill (programming) at the most fundamental level (the individual software engineer). An investment in PSP is, in effect, an investment in personal competence. Evidence indicates that this investment pays rich dividends to the investors—the organizations that pay for the training and the developers who complete it. [For more on the Personal Software Process and Team Software Process (TSP^{SM}), see <http://www.sei.cmu.edu/tsp/>, as well as the *SEI Interactive* feature on PSP/TSP <http://interactive.sei.cmu.edu/Features/1999/June/Introduction /Intro.jun99.htm>.]

As useful as PSP is, those of us experienced in building systems from commercial software components understand that the role of computer programming (in the traditional sense) is becoming less pronounced as our dependence on commercial software components becomes more pronounced. In the new world of components, software engineers are more likely to be preoccupied trying to discover *what components do*, determining how to *structure a design problem* to make best use of components, selecting how the components should be *assembled*, and *diagnosing* why the assembly is behaving strangely. Answers to these kinds of questions are precursors to a traditional programming activity, and getting the "wrong" answers inevitably leads to the "wrong" kind of programs, resulting in wasted time and effort, and, ultimately, failure.

The software engineer must possess knowledge, and sometimes very deep knowledge, about the components being used in a system in order to answer the above kinds of questions—indeed, even to know which questions to ask. Obtaining this knowledge, which I will call "component competence," requires investment. But unlike an investment in PSP, which deals with the timeless competence of computer programming (the essentials of which have not changed in 40 years), component competence must be sustained in the face of a fast-changing technology landscape. In this article, I will explain what component competence is, why it is essential to building component-based systems, and how it can be obtained “just in time” to make good engineering decisions.

Technology Competence as a Wasting Asset

Even the most unskilled chef understands (or at least believes) that the quality of a stew depends upon the quality of the ingredients used to make the stew. The same, of course, is true for component-based systems—that is, systems composed substantially from components. We can expect that the properties of the components we use will influence the properties of the system that we build, and perhaps will also influence the development process itself. For example, if all of our components are resource hogs, the final system will likely also be a resource hog. Similarly, if all of the components are “buggy” then we can be sure that the development process will be skewed toward a lot of debugging and repair work.

Which component properties most influence the development process? When we think about properties of components, we most often think about things like performance, usability, functionality, and so forth. These and other similar properties are certainly important, but they are not the properties that define the fundamental challenge of building a component-based system. Instead, I have in mind three properties that apply to most software components: complexity, idiosyncrasy, and instability. These properties are a consequence of the way the component *market* works rather than the way the components work:

What Are Commercial Software Components?

Commercial software components are software implementations, distributed in executable form, that provide one or more services. Examples of commercial software components include Microsoft Word, Netscape Communicator, Oracle relational database, SAP, and so forth. Readers interested in a more nuanced discussion of the meaning of “commercial component” may wish to read David Carney’s excellent column “The Elusive Search for Categories” <http://interactive.sei.cmu.edu /Columns/COTS_Spot/1999/December/COTS.dec99.htm>, which was *The COTS Spot* column in December 1999.

- **Complexity**. Components tend to be complex because they implement many features, or because the features are difficult to implement properly, or both. Complexity arises from the fact that a component vendor must convince consumers that it is better to buy the component than it is to build it. For a variety of reasons, some having to do with the limits of computer science (i.e., the limits of interface specification), the software industry is not good at masking this complexity. Of course, complexity is relative: something that is complex to me may be trivial to you. The old aphorism applies: *everything is easy, once you know the trick*. But learning the trick (i.e., obtaining component competence) takes time and effort, and therefore costs money.
- **Idiosyncrasy**. Software is as difficult for component vendors to write as it is for component consumers. Given the inherent complexity of components, producing high-quality implementations is an expensive and (from a business perspective) risky undertaking. It is understandable, then, that the last thing a component vendor wants to do is to develop a component that can be replaced by some other component. Instead, they strive to differentiate their products with unique and innovative features. The hope is that you will, in some way, come to depend upon these unique features so that the vendor can benefit from a continuous revenue stream as you purchase

software support and component upgrades. The downside of feature differentiation, though, is that component competence tends to be highly product-specific, and the all-important task of integrating components becomes more difficult because the unique features of one component tend not to match the features of other components.

- **Instability.** The software component industry is highly competitive. Consumers want the latest and greatest components on the market and are easily romanced by innovative features. It is said that imitation is the sincerest form of flattery, and this is certainly true in the component industry: successful features are copied by competitors. This forces the original vendor to seek new ways to differentiate its component, leading to a new round of innovation, and so forth in an endless cycle. A positive aspect of the hyper-competitive nature of the component market is that commercial software technology has capabilities today that could only be dreamed of a few years ago, and by futurists at that. On the negative side, however, most of us are straining to keep up with the market. The pace of innovation ensures that whatever component competence we obtain is sure to become stale within a surprisingly short time. If component competence is a key organizational asset, it is a wasting asset—that is, it wastes quite rapidly in the current hyper-competitive component marketplace.

These properties, taken together, pose a significant challenge to system designers and software engineers, especially as the number of components used in systems increases. Nowadays information systems of even modest scale will make use of a dozen or more commercial components. Knowing how any one component works can be a formidable challenge. Knowing how they all work, or the best ways to combine them, is more difficult still. More important, new component releases and the emergence of whole new categories of components happen much more quickly than the time it takes to build (or sometimes design) information systems. This means that component competence must often be obtained "just in time" to make key decisions, such as which components to buy and how to integrate them.

Using Toys to Obtain Component Competence

There is no doubt about it: a "hands-on" approach is required to obtain component competence. Components are simply too complex, their documentation too sketchy, and vendor literature too glossy to be exclusively relied upon.

There are two basic approaches to obtaining component competence—but the premise for both approaches is learning by doing. The first approach—*just do it*—is the more direct of the two. In this approach critical design and implementation decisions are made on the basis of available component competence. It is hoped that this competence is sufficient to avoid big mistakes, and that engineers will become more facile with the components as the project proceeds. Sometimes this works. Sometimes it doesn't. I can only say that a healthy proportion of the component-based project failures that we have encountered can be attributed to naïve assumptions about what components do and how they interact—

assumptions that could have, and should have, been verified before key design and implementation commitments
were made.

The second approach is more oblique. It begins with the building of *toys*. Before scoffing at this idea as "academic," it is important to reflect on the importance of toys in the learning process. Play is fundamental to the human condition. Philosophers and psychologists alike have long recognized *homo ludens* (human at play) as a natural state of being. Children (and even adults) play as an effective way of exploring *how things work* and *their place in the world* in an environment that is forgiving of mistakes. In our experience, engineers can most effectively learn about what components do, and how to combine them, through an analogous process of constructive play. Building toys allows engineers to explore possibilities without all of the complexities—and risks—inherent in a "live" design problem.

Key Concepts of Enterprise JavaBeans™

While I can't make you competent in EJB with a one- or two-paragraph description, I can tell you just enough about it for you to understand the examples in this article.

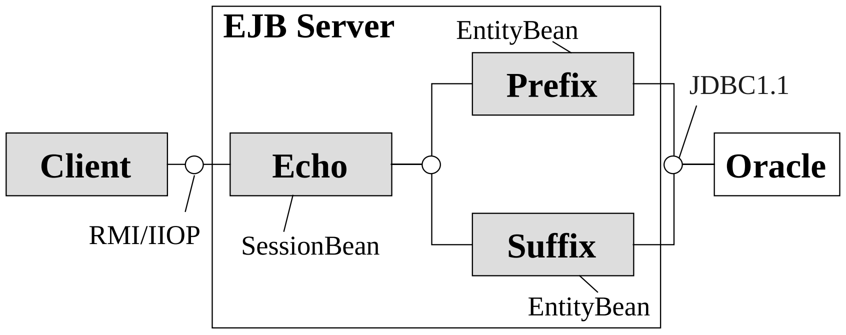
Developers write business logic as *enterprise beans*. Enterprise beans are components that are *deployed* into servers. EJB *servers* (EJB also has *containers* that execute in servers, but for our purposes we can lump container and server together) provide a runtime environment for enterprise beans, managing when they are created, activated, deactivated, cached, and deleted. EJB servers also provide a number of important services to beans, including transactions, naming, security, and thread management.

There are two major classes of enterprise bean: *session* bean, and *entity* bean. Session beans are used to export services to clients; each session bean can be connected to at most one client at a time. Entity beans are used to model business objects; they correspond to rows in a relational database table. The EJB server manages the flow of data between entity beans and relational databases. Many clients (most often these are session beans) can share a single entity bean.

Within the past year or so, a new component technology has emerged in the marketplace, called Enterprise JavaBeans™ (EJB). EJB is a Java-based approach for building "scalable, secure, distributed, transactional, interoperable enterprise systems." These are just a few of the claims made by EJB vendors. Do you believe—or *dis*believe—these claims? What is the *basis* for your beliefs? In our project at the SEI (COTS-Based Systems), we posed these questions to ourselves because some of our customers were beginning to nose around EJB. In order to gin up some competence quickly, we built a toy.

Our toy was a simple echo server: a client passes a string to the echo server and the server responds by sending back the string. To make things more interesting the server also attaches a prefix to the front of the string and a suffix to the end of the string it is presented. Our toy is illustrated in Figure 1. (Some EJB details, such as *home* and *remote* objects, are not shown.)

Figure 1: A Toy for Examining Enterprise JavaBeans



One thing that can be said for this toy is that it is simple. The motivation for its design is the desire to have *as little application functionality as possible* combined with the *greatest coverage of EJB features possible*. The boxes in gray depict the code we had to write, and there was precious little of that. On the other hand, because the application is so trivial we were able "play" with the toy to explore many different facets of EJB that would have been difficult to explore in a live project that has more complex functionality. Because there is so little application logic, almost all of the play is devoted to the EJB mechanisms themselves. For example:

- We stored the prefixes and suffixes on different versions of the Oracle database running on different platforms to see how well EJB supports heterogeneous (X/A) transactions.
- We deployed the echo session bean in one EJB server and the prefix and suffix entity beans in a different EJB server to see how well EJB supports transactions over distributed beans.
- We introduced different users and user roles, and granted different types of permissions to these users to see how well EJB supports security.
- We "ported" the toy from one vendor's EJB server to another to see how well EJB supports client and bean portability.

Because the purpose of this article is not to provide an exegesis on EJB, I will not bore you with the details of these and other playful excursions into the land of EJB. I will make two observations, however. First, each excursion required only a small investment in time—on the order of one to three days. Second, it is no idle boast to state that within a matter of two or three weeks we were able to have pointed and detailed discussions with the architect of one commercial EJB server on the limits of his product and the EJB specification. We were also able to intelligently discuss, and predict, enhancements that were planned for the EJB specification. Further, we were more familiar with the EJB specification and workings of EJB products than were researchers we had met who were building “formal models” of the EJB specification. That’s not at all bad for a few weeks’ worth of work!

From Toys to Engineering Design: Model Problems

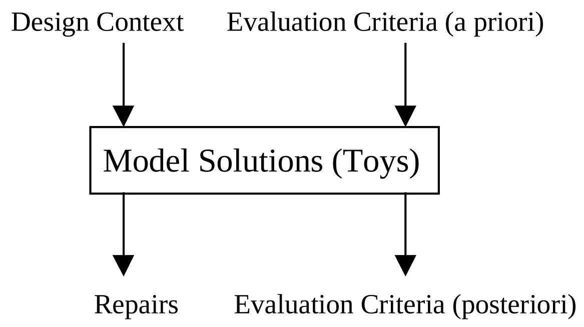
Still, the skeptical reader may observe (rightly) that system development houses have as their objective building *systems*, not an engineer’s component competence. Unless the building of toys can be seen as a clear means to this end, it might be better to resort to the more direct “just do it” approach. Fortunately, I can make this connection between toys and engineering design with *model problems*.

A model problem is a toy that has been “situated” into a real design problem by the addition of design *context*. Context includes any or all of the following:

- constraints on which components or component versions are used
- platform constraints (hardware, operating system, network topology, etc.)
- specific end-system requirements (functional requirements and quality attributes)
- implementation details (use cases, sequence charts)

A model problem is really a way of posing a question: what is the best way to use a component or an *ensemble* of components to achieve some end objective? There may be several model solutions, although in practice we usually have one particular solution in mind. To allow us to quickly focus in on the essence of the problem, and to explore alternative model solutions quickly, we try, to the best extent possible, to maintain the parsimonious simplicity of toys. Figure 2 below depicts the structure of a model problem.

Figure 2: Structure of a Model Problem


Design Context
Evaluation Criteria (a priori)
Model Solutions (Toys)
Repairs
Evaluation Criteria (posteriori)

As you can see, we have added the design context to our toys. Before we build our toy we also must be sure that we are focusing on the important questions and not just playing for the sake of playing. To this end we also must define evaluation criteria: how will we know that the proposed solution is acceptable? Sometimes the criteria will focus on feasibility—can the model solution be constructed at all? Other times the criteria will include such things as performance goals or other quality attributes.

You will also observe that the model solutions produce two kinds of output, both of which are the result of a learning process. One output is *a posteriori evaluation criteria*. In almost every situation where we have built model solutions, we have learned something unexpected that should have been part of the a priori evaluation criteria *had we known better*. The second output is what we refer to as *repairs*. It may be that the model solution does not quite satisfy the evaluation criteria. However, it is often the case that a small change (i.e., a "repair") to the design context or the toy itself could resolve the problem. For example, a system requirement might be relaxed, or an alternative component selected.

Because toys are, by design, kept ruthlessly simple, there is still a gap between the model solution and the end system. However, the evaluation criteria and design context should provide a sufficient basis for making predictions about the utility of a model solution. In any event, model solutions provide a foundation in experience that allows us, in the words of Winston Churchill, to "pass with relief from the tossing sea of Cause and Theory to the firm ground of Result and Fact." It is this grounding in result and fact, which is a consequence of our competence-building exercises, that permits us to reduce design risk.

Returning to our EJB illustration, it is now easy to see how the simple EJB toy can be "constrained" to serve as model problems in a design activity. In fact, in our work we generally skip the toy-building activity and head straight for model problems. Thus, each of the bulleted explorations we illustrated above (distributed transactions, bean portability, etc.) were themselves model problems. We learned, for example, at the time we built the model solutions (which was over a year ago) that

- EJB servers only supported JDBC1.1 and so did not support X/A distributed transactions, severely limiting what was meant by EJB support for distributed transactions.
- Distributing beans across different platforms required add-on products and was not supported by our selected EJB server "out of the box," further limiting distributed transactions.
- Although the security model was advertised to be based on public key infrastructure, in fact user identification and authentication was based on a simple password scheme.
- Enterprise beans developed for one EJB server were not portable to other EJB servers, and in fact EJB clients were, for the most part, not portable across servers either.

Each of these investigations (except perhaps for the last bullet) was driven by a particular set of design questions posed about an information system that our project had been designing. What we learned from our brief investigations could never have been discovered from documentation and vendor literature.

By now our EJB toy is quite dated, and the competence we obtained from building it is quite stale. However, these same toys could be built with today's versions of EJB, and could possibly be extended to explore new EJB features. We are confident that doing so would be a wise investment for any project considering using EJB if the engineers do not already have current experience with the technology.

Conclusions

The use of commercial components poses significant challenges to the engineering design process. Most notably, it requires the availability of rather deep competence in the components being used. Unfortunately, this competence, once obtained, wastes quite rapidly in the current hyper-competitive component marketplace. The solution is to find a way to develop this competence cheaply and effectively, and in the context of a particular design problem. We do so through the development of toys and model problems, and this has proven to be extremely effective in helping us make engineering decisions based upon observable fact rather than vendor literature.

In the next issue of SEI Interactive I will discuss how model problems fit within an iterative engineering design process. I will also describe how the "three Rs" of this process (**R**ealize model solutions, **R**eflect on their utility and risk, **R**epair the risks) can be used to reduce design risk for component-based systems.

About the Author

Kurt Wallnau is a senior member of the technical staff in the Dynamic Systems Program at the SEI, where he is co-lead of the COTS-Based Systems Initiative. Before that he was a project member in the SEI Computer-Aided Software Engineering (CASE) integration project. Prior to coming to the SEI, Wallnau was the Lockheed Martin system architect for the Air Force CARDS program, a project focused on "reusing" COTS software and standard architectures.

Removing Roadblocks to Cyber Defense

Richard D. Pethia

Editor's Note: This column is adapted from testimony by Richard Pethia, director of the CERT® Centers at the SEI, before the Senate Judiciary Committee on Technology, Terrorism, and Government Information, March 28, 2000. For the full text of Pethia's testimony, go to <http://www.cert.org/congressional_testimony/Pethia_testimony_Mar28-2000.html>.

This column will describe a number of issues that have impact on security on the Internet and outline some of the steps I believe are needed to effectively manage the increasing risk of damage from cyber attacks. My perspective comes from the work we do at the CERT® Centers, which includes the CERT® Coordination Center (CERT/CC) and the CERT® Analysis Center (CERT/AC) [For a description of these centers, see About the Author]. In 1999, the staff of the CERT Centers responded to more than 8,000 incidents. Since it was established in 1988, the CERT/CC staff has handled well over 24,000 incidents and analyzed more than 1,500 computer vulnerabilities.

An Ever-Changing Problem

The recently publicized rash of attacks on Internet e-commerce sites reminds us once again of the fragility of many sites on the Internet and of our ongoing need to improve our ability to assure the integrity, confidentiality, and availability of our data and systems operations. While it is important to react to crisis situations when they occur, it is just as important to recognize that cyber defense is a long-term problem. The Internet and other forms of communication systems will continue to grow and interconnect. More and more people and organizations will conduct business and become otherwise dependent on these networks. More and more of these organizations and individuals will lack the detailed technical knowledge and skill that is required to effectively protect systems today. More and more attackers will look for ways to take advantage of the assets of others or to cause disruption and damage for personal or political gain. The network and computer technology will evolve and the attack technology will evolve along with it. Many information assurance solutions that work today will not work tomorrow.

Managing the risks that come from this expanded use and dependence on information technology requires an evolving strategy that stays abreast of changes in technology, changes in the ways we use the technology, and changes in the way people attack us through our systems and networks. The strategy must also recognize that effective risk management in any network like the Internet is unlikely to come from any central authority, but can only be accomplished through the right decisions and actions being made at the end points: the organizations and individuals that build and use our interconnected information infrastructures. Consider this:

- We have distributed the development of the technology. Today’s networks are made up of thousands of products from hundreds of vendors.
- We have distributed the management of the technology. Management of information technology in today’s organizations is most likely distributed, and the trend toward increased collaborations and mergers will make that more likely in the future.
- We have distributed the use of the technology. The average computer user today has little in-depth technical skill and is properly focused on "getting the job done" rather than learning the nuances and idiosyncrasies of the technology.
- We must distribute the solution to the information assurance problem as well. The technology producers, organization and systems managers, and systems users are the only ones who can implement effective risk management programs.

In the long run, effective cyber defense will require:

- expanded research programs that lead to fundamental advances in computer security
- new information technology products with security mechanisms that are better matched to the knowledge, skills, and abilities of today’s system managers, administrators, and users
- a larger number of technical specialists who have the skills needed to secure large, complex systems
- improved abilities to investigate and prosecute cyber criminals
- increased and ongoing awareness and understanding of cyber-security issues, vulnerabilities, and threats by all stakeholders in cyberspace

I will focus on removing barriers to the last of these: building an ongoing awareness and understanding of cyber-security issues.

Building Awareness and Understanding

Information technology is evolving at an ever-increasing rate with thousands of new software products entering the market each month. Increasingly, cyber security depends not just on the security characteristics and vulnerabilities of basic networking and operating system software, but also on the characteristics and vulnerabilities of software used to implement large, distributed applications (e.g., the World Wide Web). In addition, attack technology is now being developed in an open source environment where a community of interest is evolving this technology at a rapid pace. Several significant new forms of attack have appeared in just the past year (for example, the Melissa virus, which exploits the widespread use of electronic mail to spread at network speeds, and distributed denial-of-service tools that harness the power of thousands of vulnerable systems to launch devastating attacks on major Internet sites). It is likely that attack technology will continue to evolve in this "public" forum and that the evolution will accelerate to match the pace of change in information technology. Once developed, this attack technology can be picked up and used by actors with significant resources to hone and advance the technology, making it a much more serious threat to national security and the effective operation of government and business.

The overall picture of vulnerability and threat is complex, but it must be understood to develop effective cyber-defense strategies. Building this understanding requires

- collection and analysis of information on the security characteristics and vulnerabilities of information technology
- collection and analysis of information on evolving attack technology
- collection and analysis of information on cyber attacks
- collection and analysis of information on cyber attackers
- collection and analysis of information on the effectiveness of defensive practices and technologies

Using this understanding to develop effective defense strategies requires

- providing technology producers and the rapidly growing community of system operators with information from the analysis activities
- convincing this community to act on this information to reduce serious vulnerabilities and implement effective security controls

The tasks described above are currently being conducted by a loose-knit network of cooperating organizations. Each organization focuses on its area of expertise and the needs of its customers or constituents. Each organization shares as much information as it can with others. Many varied organizations participate in this network, including federal, state, and local investigative organizations, security incident response teams, government labs and federally funded research and development centers, security researchers in universities and industry, technology producing organizations, security product and service vendors, system and network operators, and government agencies chartered to conduct security improvement efforts. The work of these organizations would be facilitated if the roadblocks described below were removed.

Roadblock: The Federal Debate Over Who’s in Charge

The ongoing federal debate over who’s in charge and whether or not the grand analysis center in the sky should be established is only detracting from the real work that is going on in the qualified organizations listed above. The Department of Defense must conduct data collection and analysis activities to operate and protect its networks. The FBI and NIPC must conduct data collection and analysis activities to carry out their missions of criminal investigation and infrastructure defense. GSA and NIST must conduct data collection and analysis activities to carry out their missions of dealing with incidents and improving security in the civilian agencies. University and industry researchers are among the best resources available to understand the evolution of information technology, attack technology and the interplay between them. The other organizations listed above must conduct data collection and analysis activities to meet the needs of their customers and sponsors. Attempts to replace these activities with one central data collection and analysis activity are misguided and seemingly miss the following realities:

- If you build it, they won't come. Sharing of sensitive security information is dependent on the trust relationship established between the information sender and receiver. These relationships are fragile, often take years to establish, and cannot be replaced by changing mandates or reassigning responsibilities.
- It is not possible to build an overall, comprehensive picture of activity on the networks. In spite of the strong desire to "see it all" so we can "understand it all," it is simply not possible to build a comprehensive view of activity on the networks. They are too big; they are growing too quickly; they lack the needed sensors; and they are literally being reconfigured and re-engineered on the fly. The challenge is not to pull all the data together, but to ensure that the right data is at the right place at the right time to allow local decision-makers to take effective action.
- All the talent needed to perform the analysis cannot be collected in one place. The detailed analysis work that must be done requires a combination of talents and skills and the best people that we can find. Organizations are not willing to give up their best people to other organizations, and the people are not willing to move. It is much more effective and efficient to move the data than to move the people. What is needed is an information-sharing network where data can be shared among organizations and analysis conducted at different sites for different reasons. The challenge is not to pull all data together, but to push it out to meet the varying needs of the various audiences.
- Centralization is not more efficient. Any central organization, unfamiliar with the operational needs of any particular network operator, technology developer, or researcher, will only be able to perform generic analysis tasks that yield high-level results. The detailed work must still be done to develop the detailed strategies and plans needed to build an effective cyber defense. Centralization is more likely to increase costs rather than decrease them. What is needed is increased collaboration among all players able to contribute to and draw from a growing body of data and knowledge.

Roadblock: Inadequate Resources for the Work That Must Be Done

The federal government has studied and debated the cyber-security problem for years. The newest flurry of activity began with the Presidential Commission on Critical Infrastructure Protection in 1996 and has led to the establishment of the National Infrastructure Protection Center and the creation of the National Plan for Information System Protection. However, many of the views being discussed and debated today are echoes of earlier studies and conclusions. The 1989 DARPA-funded study, *Computers at*

Risk[2], reached many of the same conclusions and recommended many of the same actions as the more recent studies. What has been missing is action and funding to take the steps needed to deal with this problem effectively. In spite of the nearly exponential growth of security incidents and security vulnerabilities over the last ten years, there has been little increase in budget to deal with these problems. Analysis centers must be resourced, information-sharing infrastructures must be established, and transition activities that move needed information and security solutions their eventual users must be staffed. We will make progress when we invest in making progress.

Roadblock: Lack of Protection for Sensitive and Company-Proprietary Data

Information sharing between the private sector and the federal government is impeded by the lack of protection from FOIA and other forms of disclosure. Organizations that are the victims of cyber attacks can contribute greatly to the understanding of cyber defense by providing detailed information regarding the security incidents they have suffered: losses, methods of attack, configurations of systems that were successfully attacked, processes used by the organization that were vulnerable, etc. Much of this information is extremely sensitive and could be used to damage the corporation if it became public. In addition, corporations often have more to lose from damaged reputations than from the attacks themselves. These organizations will not share security incident or loss information unless they have a high degree of confidence that this information will be protected from public disclosure. The federal government must take steps to protect the sensitive data as a precursor to information sharing. Only then will it be possible to form the trust relationships and begin data-sharing activities.

Roadblock: Lack of Information on Threats

Any effective risk management strategy requires an understanding of three things:

1. the value of the assets that must be protected and the consequences of loss of confidentiality or operational capability
2. the vulnerabilities that could be exploited to bring about the losses
3. the threats that exist—the actors who would exploit the vulnerabilities and some indication of the probability that they would do so

Today we are awash in information regarding vulnerabilities in our technologies and our networked systems. Computer security incident response teams warn their constituents of vulnerabilities that are being exploited. Internet news groups routinely publish descriptions of vulnerabilities and methods to exploit them. Technology vendors alert their customers to vulnerabilities in their products and provide software upgrades to

[2] Computers at Risk: Safe Computing in the Information Age, National Research Council. Washington, D.C.: National Academy Press, 1991.

correct them. Conferences and training courses abound that focus on corrections to vulnerabilities.

At the same time, system and network operators are becoming increasingly aware of the value of their information assets and of their growing dependence on the Internet and other communications infrastructures. The current emphasis on electronic commerce and use of the Internet as a powerful marketing and sales tool is sure to accelerate this understanding. With all this focus on value and vulnerability, why are so many organizations taking so little action to improve their cyber-security? Because they have little hard data that convinces them that there are real threats to their operations. We all know that we are vulnerable to many things. Our cars are vulnerable to certain forms of attack. Our homes and places of business are vulnerable to certain forms of attack. As individuals, we are vulnerable to certain forms of attack Yet we are not all driven to distraction by this sea of vulnerability. We first focus not on vulnerability but on threat. We act to correct vulnerabilities when we believe there is a significant probability that someone will take advantage of them. The same is true in cyberspace. Operational managers know that they cannot afford to eliminate every vulnerability in their operations. They need data to help them understand which ones are most critical, and which ones are likely to be exploited.

Our law enforcement and intelligence organizations must find ways to release threat data to the operational managers of information infrastructures to motivate these managers to take action and to help them understand how to set their priorities. In the absence of a smoking gun, it is unlikely that many organizations will have the motivation to invest in improved cyber defense.

About the Author

Richard D. Pethia is the director of the CERT® Centers at the Software Engineering Institute (SEI), Carnegie Mellon University. The CERT Centers include the CERT Coordination Center and the CERT Analysis Center. The SEI has operated the CERT/CC since 1988, and has provided a central response and coordination facility for global information security incident response and countermeasures for threats and vulnerabilities. The recently established CERT Analysis Center addresses the threat posed by rapidly evolving, technologically advanced forms of cyber attacks. Working with sponsors and associates, this center collects and analyzes information assurance data to develop detection and mitigation strategies that provide high-leverage solutions to information assurance problems, including countermeasures for new vulnerabilities and emerging threats. The CERT Analysis Center builds upon the work of the CERT Coordination Center.

Prior to becoming director of the CERT Centers, Pethia managed the SEI Networked Systems Survivability Program, focusing on improving both the practices and understanding of security and survivability issues relating to critical information infrastructures. Before coming to the SEI, Pethia was director of engineering at Decision Data Computer Co., a computer system manufacturer in Philadelphia. There he was responsible for engineering functions and resource management in support of new product development. Pethia also was manager of operating systems development for Modular Computer Corp. in Fort Lauderdale, FL. While there, he led development efforts focused on real-time operating systems, networks, and other system software in the application areas of industrial automation, process control, data acquisition, and telecommunications.

Making the Tactical Case for Process Improvement

Watts S. Humphrey

In the December and March columns, I addressed how to make the strategic case for process improvement. It is easier to justify a process improvement program when you deal at a strategic level. Process improvement is a long-term strategic investment and it is often hard to justify to managers whose principal focus is short term. This column discusses how to handle the shorter-term tactical improvement case. We start on the assumption that you are in an organization where the management is focused on tactical issues and where nobody, at least nobody at a senior level, is willing to look beyond the current month or quarter. These managers are generally so consumed by current problems that they cannot consider anything that will pay off in a year or more. While the likelihood of success for a short-term improvement program is low, the situation is not entirely hopeless.

In this column, I talk about tactically based improvement programs and some of the strategies that you can use to bring them off. While there is no guarantee that your efforts will work, there generally are ways to achieve useful results in a relatively brief period. Although you may not succeed, it is better to try something than do nothing. So give it a shot. You might make some useful improvements, and, even better, your success might convince some managers to think and act strategically. This column recommends a series of short-term tactical steps that can lead to a long-term strategic improvement program

The Tactical Situation

In the typical organization, management claims to be thinking strategically. However, imagine that you work for a company in a highly competitive industry that is struggling to improve its quarterly earnings. While process improvement is accepted as a great idea, nobody will invest any significant money in it. What is more, your improvement proposal must be cost justified, and you must show that the costs can be recovered in less than a year without seriously disrupting any project.

Management claims that its goal is to be the industry leader, but the first question they are likely to ask is, “Who else is doing this and what good has it done them?” While you might be tempted to suggest that they sound like followers instead of leaders, restrain yourself and try to think about how to justify this proposal in a way that management will buy. This is the situation. What can you do?

Possible Approaches

While management knows all of the buzzwords, would like to act strategically, and talks about being the industry leader, managers are hypnotized by their short-term problems. Either business realities will not permit a long-term view, or some higher-level manager is under severe pressure to show improved quarterly financial results. Under these conditions, you must ignore all the high-sounding phrases about leadership, quality, and improvement, and focus instead on a few pragmatic steps that will fit the current realities.

The only way to break through management's resistance is to somehow demonstrate that the organization's current short-term problems cannot be fixed with short-term Band-Aids. You must take a strategic view. The two principal approaches you can take are to: (1) make the strategic improvement activities tactically attractive, or (2) start a small tactical effort and gradually build it into a strategic improvement program.

Making a Strategic Improvement Program Tactically Attractive

The U.S. Air Force's decision to evaluate bidders based on their process maturity made the Capability Maturity Model® (CMM®) process-improvement program attractive to many managers. The Air Force evaluations forced many tactically focused managers to invest in process improvement to avoid losing business. This illustrates the advantage of having a customer demand quality improvement: it makes strategic improvement programs tactically attractive.

This strategy will generally work when you have a customer that is interested enough in quality to require that its suppliers commit to improvement programs. If you have such a customer, you can often connect your process-improvement proposal to that customer's demands. If you can get the support of your marketing department, your chances of success are pretty good. While you must keep the scale of your improvement program realistically related to the size of the likely new business, if an important customer insists on a quality program, you probably have a sound basis for a process improvement proposal.

Another approach that is almost as effective is to connect your improvement efforts to an already approved corporate improvement effort. Examples would be obtaining ISO 9000 certification or initiating a 6-sigma software quality improvement effort. Again, if you can show that the improvements you espouse will assist the organization in meeting already established goals, you have a reasonable chance of getting the improvement effort approved.

Build a Small Effort into a Strategic Program

If neither of these strategies work, you probably will not be able to make a strategic program tactically attractive, at least not in the short term. Under these conditions, you must focus on justifying a series of small improvement steps. The suggested approach is to identify one or two narrowly focused efforts that can be completed rather quickly and that don’t cost a great deal of money. Then put them into place and use the resulting benefits to help justify taking the next step.

If you are careful about what improvements you pick, and if you build support for each step as you take it, over time, you can probably keep the improvement program moving forward. Then you can gradually convert your short-term tactical activities into a longer-term strategic effort.

The critical issue in this situation is getting approval for the initial effort, and then getting the engineers and project managers to support each step as you take it. As long as you can demonstrate that the program is not too expensive and is producing results, and as long as you have the support of the project managers and working engineers, you can probably keep the program going. Then, given a little time, you should be able to show that you are saving the company money and improving project performance. This should allow you to gradually increase the size of the improvement program.

Suggested Tactical Improvement Priorities

In picking improvement efforts, concentrate on activities that are low cost, can be implemented by one or two projects, will produce immediate measurable results, and will attract strong project support. While there are several candidate improvement activities that meet these criteria, the ones that are probably the best bets for organizations at CMM levels 1 or 2 are code inspections, design courses, and the Personal Software Process/Team Software Process^{SM} (PSP/TSP) ^{SM}. These can all be focused on individual projects and they all support various aspects of a CMM-based process improvement strategy.

The approach is to pick efforts that can be implemented without a broad organization-wide effort that requires senior management approval. Then, if the involved project leaders agree to support the proposal, management will generally go along. Because these improvement efforts can be implemented without requiring changes in the entire organization, they are good candidates for quickly demonstrating the benefits of process improvement programs.

Code Inspections

Code inspections can be put into place quickly, and they pay enormous dividends [Fagan 1976, Gilb, Humphrey 1989]. While design inspections could also be helpful, they take more time and money and don’t have nearly as high a payoff—unless you have an effective design process in place. Therefore, it is usually best to defer initiating design inspections.

To start a code inspection program, first find a project leader who agrees to implement the initial trial program, and then train all of the engineers who will do the inspections. In addition, get some member of your staff qualified to moderate the inspections and to help the engineers to do the inspections properly. It would also be advisable to hire an expert to teach the initial courses. In starting an inspection program, a number of available references can be helpful, including Appendix C of my book *Introduction to the Team Software Process,* which describes the inspection process [Fagan 1976, Fagan 1986, Gilb, Humphrey 1989, Humphrey 2000a].

After you complete the first code inspections, you can usually use the engineers who did them as references to help convince the leaders of the other projects. While it will take time to put a complete code inspection program in place, it will provide substantial benefits, and it should give you a firm foundation for further improvements.

Design Courses

Until code quality is reasonably good, design improvements generally will not improve test time or product quality substantially. The reason is that poor quality code will result in so many test defects that the design problems will be lost in the noise. However, once you have code inspections in place, you get substantially improved code quality and reduced test time. That is when design courses would make sense.

While it is almost impossible to cost-justify a design course, you probably will not need to do so. Most people intuitively understand that design is important, and engineers generally will be interested in taking a design course. Start by identifying a project leader who is willing to sponsor the initial test, then find a qualified instructor and get some design courses taught. Assuming that the course is properly done, other projects will want to join in, and demand for the course will grow quite quickly.

The only additional requirement is that you have one or two qualified people available to consult with the engineers, and to advise them on how to use the design methods when they start applying them on the job. Then, after the design methods are in place, the engineers will have the criteria to judge the quality of the designs that they inspect. That is the time to introduce design inspections.

The PSP and TSP

After successfully introducing the inspection program and the design courses, you will have a substantial level of credibility and a modest staff. Then, you can think about tackling a more challenging improvement effort. This would be a good time to get yourself or one of your people trained as a PSP instructor and TSP launch coach [Humphrey 1995]. While this will take a few months and cost a little money, the training is readily available and will enable you to introduce the PSP and TSP into one or two projects in your organization [SEI].

After getting one or more staff members qualified as PSP instructors, look for a trial project. Training volunteers is the easy-sounding approach, but to successfully introduce the PSP and TSP, you must focus on entire teams, including the managers. Once you identify a team and have a qualified PSP instructor available, you can introduce the PSP and TSP in only three or four months. Even though it will generally be several months before you have data on completed projects, TSP's planning and tracking benefits will be apparent very quickly.

To spread the PSP and TSP to other projects, first convince the managers. The material in my March column should be helpful [Humphrey 2000b]. To convince the engineers to participate, get the first TSP team to meet with the potential new team, and have all the managers leave the room. The engineers from the first project will be most effective at convincing the second team to use the PSP and TSP. Once you have a few TSP teams in place, you will have the data, experience, and support to launch a broader-based improvement program.

The Commitment System

Assuming that your tactically focused improvement efforts have so far been successful, you can start to move toward a broader-based CMM improvement effort. Be cautious about moving too fast and keep your proposals modest until you are reasonably certain that they will be accepted. The suggested next step is to fix the organization's commitment system.

The commitment process defines the way organizations commit project costs and schedules to customers. Improvements in the commitment system generally produce significant benefits very quickly. The basic principles of the software commitment system are well known [Humphrey 1989, Humphrey 2000a, Paulk]. Improving the commitment system is an important early step in a CMM-based improvement program.

Changing the commitment system is much more difficult than anything you have attempted so far. For some reason, managers who make plans before they commit to constructing a building, starting a manufacturing program, or even taking a trip, do not appreciate the need for planning software projects. Changing the commitment process requires that the managers change their behavior. If you thought changing engineering behavior was difficult, wait until you try to change management behavior. This is why a full-scale CMM-based process improvement program can be difficult to launch.

While the commitment system is probably the most important area to improve, unlike inspections or the PSP/TSP, it cannot be done for only one or two projects. If you can change the commitment system, however, you should be able to launch a full-scale, strategic-based process improvement program. This should include a full CMM-based effort, as well as an expansion of the PSP and TSP to cover all of the development and maintenance projects in the organization.

Acknowledgements

In writing papers and columns, I make a practice of asking associates to review early drafts. For this column, I particularly appreciate the helpful comments and suggestions of Anita Carleton, Sholom Cohen, Jim McHale, Julia Mullaney, Mark Paulk, Bill Peterson, and Marsha Pomeroy-Huff.

A Note to My Readers

After publishing the March column, I found that I had made a mistake in calculating the maintenance savings in the example. The maintenance numbers were about twice what they should have been. The March column has now been corrected. I hope this mistake has not caused you any inconvenience or embarrassment.

In Closing, an Invitation to Readers

In these columns, I discuss software issues and the impact of quality and process on engineers and their organizations. However, I am most interested in addressing issues that you feel are important. So, please drop me a note with your comments, questions, or suggestions. I will read your notes and consider them in planning future columns.

Thanks for your attention and please stay tuned in.

Watts S. Humphrey
watts@sei.cmu.edu

About the Author

Watts S. Humphrey founded the Software Process Program at the SEI. He is a fellow of the institute and is a research scientist on its staff. From 1959 to 1986, he was associated with IBM Corporation, where he was director of programming quality and process. His publications include many technical papers and six books. His most recent books are *Managing the Software Process* (1989), *A Discipline for Software Engineering* (1995), *Managing Technical People* (1996), and *Introduction to the Personal Software Process*^{SM}(1997). He holds five U.S. patents. He is a member of the Association for Computing Machinery, a fellow of the Institute for Electrical and Electronics Engineers, and a past member of the Malcolm Baldrige National Quality Award Board of Examiners. He holds a BS in physics from the University of Chicago, an MS in physics from the Illinois Institute of Technology, and an MBA from the University of Chicago.

References

[Fagan 1976] Michael Fagan, "Design and Code Inspections to Reduce Errors in Program Development," *IBM Systems Journal*, vol. 15, no. 3, 1976.

[Fagan 1986] Michael Fagan, "Advances in software inspections," *IEEE Transactions on Software Engineering*, vol. SE-12, no. 7, July 1986.

[Gilb] Tom Gilb, Dorothy Graham, *Software Inspection*, Edited by Susannah Finzi, Reading, MA: Addison-Wesley, 1993.

[Humphrey 1989] W. S. Humphrey, *Managing the Software Process.* Reading, MA: Addison-Wesley, 1989.

[Humphrey 1995] W. S. Humphrey, *A Discipline for Software Engineering.* Reading, MA: Addison-Wesley, 1995.

[Humphrey 2000a] W. S. Humphrey, *Introduction to the Team Software Process*, Addison-Wesley, Reading, MA 2000.

[Humphrey 2000b] W. S. Humphrey, “Justifying a Process Improvement Proposal, *SEI Interactive*, March, 2000.

[Paulk] Mark C. Paulk and others, *The Capability Maturity Model: Guidelines for Improving the Software Process.* (Reading, MA: Addison Wesley, 1995)

[SEI] Information on PSP and TSP courses is available on the SEI web site at http//www.sei.cmu.edu/tsp.