CrossTalk

April 2001 The Journal of Defense Software Engineering Vol. 14 No. 4

The Promise of Web-Based Applications

News and Updates

Web-Based Applications

On the Cover:
Kent Bingham, Digital Illustration and Design, is a self-taught graphic artist/designer who freelances print and Web design projects.

Best Practices

Open Forum

Departments

CrossTalk

Sponsor: *Lt. Col. Glenn A. Palmer*
Publisher: *Tracy Stauder*
Associate Publisher: *Elizabeth Starrett*
Managing Editor: *Pam Bowers*
Associate Editor/Layout: *Benjamin Facer*
Associate Editor/Features: *Nicole Kentta*
Graphic Designer: *Janna Jensen*

Voice: 801-586-0095
Fax: 801-777-5633
E-mail: crosstalk.staff@hill.af.mil
CrossTalk Online: www.stsc.hill.af.mil/Crosstalk/crostalk.html
CRSIP Online: www.crsip.hill.af.mil

Subscriptions: Send correspondence concerning subscriptions and changes of address to the following address. You may e-mail or use the form on p. 31

Ogden ALC/TISE
7278 Fourth St.
Hill AFB, Utah 84056-5205

Article Submissions: We welcome articles of interest to the defense software community. Articles must be approved by the CrossTalk editorial board prior to publication. Please follow the Author Guidelines, available at www.stsc.hill.af.mil/CrossTalk/xtlkguid.pdf. CrossTalk does not pay for submissions. Articles published in CrossTalk remain the property of the authors and may be submitted to other publications.

Reprints and Permissions: Requests for reprints must be requested from the author or the copyright holder. Please coordinate your request with CrossTalk.

Trademarks and Endorsements: This DoD journal is an authorized publication for members of the Department of Defense. Contents of CrossTalk are not necessarily the official views of, or endorsed by, the government, the Department of Defense, or the Software Technology Support Center. All product names referenced in this issue are trademarks of their companies.

Coming Events: We often list conferences, seminars, symposiums, etc. that are of interest to our readers. There is no fee for this service, but we must receive the information at least 90 days before registration. Send an announcement to the CrossTalk Editorial Department.

STSC Online Services: at www.stsc.hill.af.mil
Call 801-777-7026, e-mail: randy.schreifels@hill.af.mil

Back Issues Available: The STSC sometimes has extra copies of back issues of CrossTalk available free of charge.

The Software Technology Support Center was established at Ogden Air Logistics Center (AFMC) by Headquarters U.S. Air Force to help Air Force software organizations identify, evaluate, and adopt technologies to improve the quality of their software products, efficiency in producing them, and their ability to accurately predict the cost and schedule of their delivery.

The Two Sides of the Web We Weave

Doesn't there seem to be some good and bad in almost everything we do these days? And aren't there always "two sides of the coin?" Take the World Wide Web. I'd say it has two sides, too.

There is no doubt that the Web has been a very positive influence and convenience in our everyday lives. From travel agencies to bookstores, we now have resources just a few mouse-clicks away instead of miles away as in the not so distant past. Unfortunately, the web has brought another dark side to our society. Whether it's an expensive hobby or a time sink for some, it's just another medium of vices for others.

As you will read in this month's issue, the Department of Defense is dealing with its own two sides of the web. Pam Dargan writes of the bright side in *Ideal Collaborative Environment*. In our workplace today, we enjoy many electronic collaboration capabilities such as bulletin boards and chat rooms to share information and lessons learned. And after some technical challenges are overcome, our future at work may soon include virtual teams and virtual meetings. The airlines won't be happy with these near-term ways of doing business.

On the other side, Robert Martin warns us of *The Vulnerabilities of Developing on the Net* and how through the Common Vulnerabilities and Exposures Initiative we can reduce our risks when working with vulnerable commercial software on the shelf today. In addition, Lt. Col. Alford Jr writes of the risk our nation faces in *Cyber Warfare: A New Doctrine and Taxonomy*. He explains how cyber systems can make nations vulnerable to warfare without violence, and how cyber warfare may be the greatest threat that nations have ever faced. This is a scare that not many have taken seriously yet.

Also, I was surprised to learn at an information technology seminar recently that there now exists more than 30,000 hacking oriented Web sites. Amazingly, "click and hack" programs can be downloaded from these sites. You no longer need to be a guru to be a hacker. Thus it has never been more important to safeguard your personal and workplace computers with appropriate security precautions.

Fortunately, because of the Web, most of us can say that we work in a more productive and user-friendly environment than we did just five years ago. Web technologies and applications continue to progress and enter into almost every workplace. I believe that the defense software community is just beginning to discover the benefits of the World Wide Web.

One benefit that you may not have experienced yet is CROSSTALK online at www.stsc.hill.af.mil. This month's issue and all of our back issues are available along with subscription forms, theme announcements, and author guidelines. I'd like to thank all of our on-line readers for coming back each month. Don't forget to drop us a line to let us know how we can better meet your on-line needs.

I am pleased to feature in this month's issue our first announcement on our search for finding successful software projects (See page 4). The Office of the Director of Defense Research and Engineering is sponsoring an effort to recognize the top five software projects in the government. This effort is one response to the recommendations of the Defense Science Board Task Force on Defense Software. CROSSTALK, is coordinating this effort, and we are looking for projects focused on quality, performance, and customer value. We are currently accepting nominations from project managers, teams, and customers.

Tracy L. Stauder

Tracy L. Stauder
Publisher

OFFICE OF THE DIRECTOR OF
DEFENSE RESEARCH AND ENGINEERING
3040 DEFENSE PENTAGON
WASHINGTON, D.C. 20301-3040

30 JAN 2001

MEMORANDUM FOR ALL GOVERNMENT SOFTWARE PROJECTS

SUBJECT: CrossTalk's Top Five Software Projects

As the Department of Defense Executive Agent for Software Intensive Systems, I am pleased to announce that *CrossTalk, The Journal of Defense Software Engineering*, will be dedicating its January 2002 issue to the top five software projects in the government. The aim is to recognize outstanding performance of software teams and to promote best practices. The *CrossTalk* staff is accepting nominations for this honor through 20 July 2001.

To be eligible for this award, projects must have been performed under a government contract (internal government contracts also eligible) and provided a software deliverable to the customer during the period of January 2000 through June 2001. The deliverable may be a completed contract or an incremental deliverable.

The selection of the top 5 software projects will be based upon the projects' adherence to quality and to the contracted cost, schedule, and requirements. Specific nomination criteria, process, and forms can be acquired from the CrossTalk Web site located at http://www.stsc.hill.af.mil/CrossTalk

Delores M. Etter

Delores M. Etter
Deputy Under Secretary of Defense
(Science and Technology)

The Vulnerabilities of Developing on the Net

Robert A. Martin
The MITRE Corporation

Disaster has struck. You would think that firewalls, combined with filtering routers, password protection, encryption, and disciplined use of access controls and file permissions would have been enough protection. However, an overlooked flaw in the commercial web server application allowed a hacker to use a buffer overflow attack to leverage the application's privileges into administrator-level access to the server. From there it was easy to gain access to other machines within the Intranet and replace the public Web pages with details of the hack. With the company's public site showing a live video stream of an ongoing internal, private and sensitive company meeting, it left little room for doubt as to how badly they had been hacked.

While most organizations have addressed the various aspects of implementing cyber security, many are failing to successfully address the one security area where someone can bypass all other efforts to secure the enterprise. That area is finding and fixing known security problems in the commercial software used to build the systems. There may be an answer, however, that will transform this area from a liability into a key asset in the fight to build and maintain secure systems. The answer rests in an initiative to adopt a common naming practice for describing the vulnerabilities, and the inclusion of those names within security tools and services. The initiative has been in practice for more than a year across a broad spectrum of the information security and software products community: It is called the Common Vulnerabilities and Exposures (CVE) initiative.

To Err Is Human

Every programmer knows they make mistakes when writing software, whether it be a typo, a math error, incomplete logic, or incorrect use of a function or command. Sometimes the mistake is even earlier in the development process – reflecting an oversight in the requirements guiding the design and coding of a particular function or capability of a software program. When these mistakes have security implications, those with a security bent will often refer to them as vulnerabilities and exposures.[1]

All types of software, from large complex pieces to small and focused ones, are likely to contain software mistakes with security ramifications. Large complex software like operating systems, database management systems, accounting systems, inventory management systems, as well as smaller applications like macros, applets, wizards, and servlets need to be evaluated for mistakes that can impact their security integrity. Remember that when we put these various software products together to provide an overall system, each of the software elements that make up the system could be the one that compromises it.

Things were different in the past when an organization's computer systems were stand-alone and only interacted with other systems within the same organization. Only a few systems used tapes and file passing to exchange information with outside systems. The same holds true for government and military systems, including weapons. This isolation meant that errors in commercial or developed software usually had limited impact, at least from the public's point of view. In fact, most errors, crashes, and oversights went unnoticed by the general public. At most, these problems would cause occasional troubles for an organization's closest business partners.

There Is No Hiding Now

The same is not true today. Very few of today's organizations, whether in the private sector or government, have or build self-contained systems. It is the norm for employees, customers, business partners, and the general public to have some degree of access and visibility into the minute-by-minute health and performance of an organization's software environment. Processing delay, calculation mistakes, system downtime, even response time slowdowns are noticed and often draw criticism.

Accompanying this increased visibility is an explosion in the different ways systems are accessed and used. Web and application servers have been created to help make systems interconnect and leverage Internet-based technologies. Access to web sites, purchase sites, online help systems, and software delivery sites makes the organizations that own the sites very visible. To better support business partners and employees working at remote locations, on the road, or from home, we have connected our backroom systems to the corporate Intranet and extranet. New technologies have emerged, like instant messaging, mobile code, and chat, whose functionality requires effortless access by users across organizational boundaries. The movement to highly accessible systems, driven by the need to save time and make businesses more efficient, and the reality of having to do more with less, has dramatically increased the impact of mistakes in commercial software.

While errors in self-developed software can still have a major impact on an organization's ability to function, it is the vulnerabilities and exposures in the commercial software they use to build systems that creates the bigger problem. A mistake in a commercial program can open a front or a back door into situations that most organizations strive to avoid. A mistake permitting unauthorized access can expose private information about customers and employees. It can allow hackers to change information or perform services with your systems to their own advantage. In addition, a vulnerability can allow them to shut down your internal and publicly accessed systems, sometimes, without your knowledge. In those cases where the vulnerability or exposure allows someone to make changes or bring down systems, or when the theft of services and information is eventually

noticed[2], there can be a huge impact to the organization's public image[3]. There can also be legal liability and direct operational impact.

What Can You Do?

Determining the vulnerabilities and exposures embedded in commercial software systems and networks is a critical "first step" to fixing the problems. A simple patch, upgrade, or configuration change could be sufficient to eliminate even the most serious vulnerability, if you know what you need and how to get it.

To find information about vulnerabilities in commercial software that your organization uses, you have to do some research and probably spend some money. With commercial software, the customer has little or no insight into the implementation details. At the very best you may have an understanding of the general architecture and design philosophy of a package. Companies offering commercial software treat the design details and software code as business-critical private information. In addition, since most of these companies are highly competitive, commercial software vendors are sometimes reluctant to share their problems, even with their customers.

Who Knows?

So how do you find out about commercial software vulnerabilities if the vendors are not going to tell you? During the last decade, three groups have emerged who share the same curiosity. For sake of discussion we will refer to these as the hackers[4], the commercial interests, and the philanthropists. The hackers, unfortunately, want to find vulnerabilities and exposures so they can exploit them to gain access to systems.

Those with commercial interests want to be hired to find the mistakes, or they want you to buy their tools to help you find the vulnerabilities and exposures yourself. They offer their services through consultants who will evaluate your software systems, and through tools that you can buy and run yourself. Some proffer the use of their tools as an Internet-based service. This group includes software and network security companies that provide security consulting services and vulnerability assessments, databases of vulnerabilities and exposures, and the tools for security services and vulnerability evaluations.

The philanthropists include security researchers in various government, academic, and non-profit organizations, as well as unaffiliated individuals that enjoy searching for these types of mistakes, usually sharing their knowledge and tools freely.

Each group has members focused on sharing information: among like-minded hackers, for a price in most cases for the commercial interests group, and generally for free in the philanthropists group. For all three groups the search for vulnerabilities and exposures in commercial software is challenging since the commercial marketplace is constantly developing and authoring new classes of software capabilities and new ways of using them. This mushrooming of commercial software capabilities also creates an ever-changing challenge for organizations using commercial systems. The challenge is to correctly configure and integrate the offerings of various vendors without opening additional vulnerabilities and exposures from configuration and permission mistakes.

How to Find Out

In response to the arduous task of tracking and reacting to new and changing vulnerabilities and exposures, the members of these three groups are using Web sites, news groups, software and database update services, notification services like e-mail lists, and advisory bulletins to keep their constituents informed and current.

So information on vulnerabilities in commercial software is available. That is great, right? Well, not quite. There are several problems. The biggest is that each organization (or individual) in these three groups has been pursuing their vulnerability discovery and sharing efforts as if they were *the* source of information on vulnerabilities. Each uses its own approach for quantifying, naming, describing, and sharing information about the vulnerabilities that they find. Additionally, as new types of software products and networking are introduced, whole new classes of vulnerabilities and exposures have been created that require new ways of describing and categorizing them.

Another problem is that finding the vulnerabilities and exposures within systems is just the first step. What we really want to do is to take the list of vulnerabilities and get them fixed. This is the software vendors' domain – those who create and maintain our commercial products. Unless they use the same descriptions and names as the hackers, commercial interests, and philanthropists groups, it is difficult, confusing and frustrating to get the fix for any particular problem you find.

A Closer Look at Who Knows What

The Internet is the main conduit hackers use to share information on vulnerabilities and how to exploit them. Different member organizations in the commercial interests group have their own mechanisms for sharing vulnerability information. For example, the tool vendors create vulnerability scanners that are driven by their own vulnerability databases. The intrusion detection system (IDS) vendors build different types of software systems for monitoring your network and systems for attacks. There are also scanner and IDS tools available from the philanthropists group as freeware. Both the scanner and IDS providers have to continuously update their tools with new information on what and how to look for problems. Examples of these organizations and tools are shown in Table 1.

Scanners typically include tests that compare version infor-

Table 1. *Scanner and IDS Offering Examples*

Product	Tool Type	Organization
Centrax	scanner/IDS	CyberSafe
CyberCop	scanner	Network Associates
Dragon	IDS	Network Security Wizards
HackerShield	scanner	Bindview Corporation
LANPATROL	IDS	Network Security Systems
Nessus	freeware scanner	Renaud Deraison & Jordan Hrycaj
NetProwler	IDS	Axent Technologies
QualysGuard	ASP-based scanner	Qualys
RealSecure	IDS	Internet Security Systems
Retriever	scanner	Symantec Corporation
SAINT	scanner	World Wide Digital Security
Secure IDS	IDS	Cisco Systems
STAT	scanner	Harris Corporation
SWARM	scanner	Hiverworld, Inc.

mation and configuration settings of software with an internal list of vulnerability data. They may also conduct their own scripted set of probes and penetration attempts. IDS products typically look for indications of actual attack activities, many can then be mapped to the specific vulnerabilities that these attacks could exploit. The scanner market recently developed a self-service-based capability. It uses remotely hosted vulnerability scanners on the Internet that you can hire to scan your Internet resident firewalls, routers, and hosts. The results of the scans are provided through a secure link, and you can usually run the scans whenever you want. These scans are shielded from everyone but you, including the service provider. IDS capabilities are often available as part of a managed security service, where the organization contracts out the intrusion detection and monitoring to a security services vendor.

Both IDS and scanner tool providers harvest information about vulnerabilities and exposures from public information sites, hacker sites, newsletters, and advisories. They also have their own investigative researchers who continuously look for new vulnerability information that will make their company's offering better than the competition[5], as well as providing them with the "free" advertising that comes with finding and publicly reporting new vulnerabilities and exposures. Typically, these researchers serve as consultants within the company, offering their services to evaluate an organization's systems and networks. Their parent companies also offer databases of vulnerabilities for a fee, although some also share the information openly as raw information on a Web site.

Some members of the philanthropists group also offer very sophisticated search and notification services for free, but their veracity, quality, and levels of effort vary considerably. Examples of vulnerability-sharing organizations are shown in Table 2.

Site Name	Type	Organization
arachNIDS	free IDS database	Max Vision Network Security/Whitehats
CERIAS Vulnerability Database	database	CERIAS/Purdue University
Fyodor's Playhouse	hacker web site	Insecure.Org
Online Vulnerability Database	database	Ernst & Young's eSecurityOnline.com
ICAT Metabase	free web site	NIST
Bugtraq mailing list Database	mailing list database	SecurityFocus.com
PacketStorm	hacker web site	Security, Inc.
SWAT Database	database	Avent Technologies
Vigil@nce AQL	database	Alliance Qualité Logiciel
X-Force Database	free web site	Internet Security Systems

Table 2: *Vulnerability Sharing Examples*

In addition to freeware scanner, IDS tools, and vulnerability databases, the philanthropists group's government and academic members offer several announcement, alert, and advisory services that are widely used and highly valued. Some commercial interests group companies offer these types of free services as well. Examples are shown in Table 3.

There are numerous venues for finding out what vulnerabilities and exposures exist in your organization's commercial software systems, as well as many tools and service providers willing to help you determine which vulnerabilities and exposures you have.

The three groups we've covered -- hackers, commercial interests, and philanthropists – all address locating the vulnerabilities and exposures in the commercial software that forms the base of your live systems and networks.

Service	Type	Organization
Bugtraq	e-mail list	Bugtraq
Casandra	alerts	CERIAS/Purdue University
CERT Advisories	advisory	CERT Coordination Center
CyberNotes	monthly newsletter	NIPC
Razor	advisory	Bindview Corporation
S.A.F.E.R.	monthly newsletter	The Relay Group
SANS NewsBites	e-mail list	SANS Institute
Security Alert Consensus	e-mail list	Network Computing and SANS
SecurityFocus Newsletter	newsletter summary of Bugtraq e-mails	SecurityFocus.com
SWAT Alerts	alerts	Avent Technologies
X-Force Alert	advisory	Internet Security Systems

Table 3: *Alert and Advisory Services Examples*

We will now address finding the "fixes." The product vendors who make the software in which these vulnerabilities were found provide the solutions for vulnerabilities. Many of them have their own methods of providing their customers with software fixes and updates. Until recently, most vendors were not very proactive in distributing patches and updates outside of their normal software development cycle. This has improved considerably. Now, many major vendors provide alerts and advisories concerning security problems, fixes, and updates (See Table 4).

Service	Type	Organization
IBM ERS	advisory	IBM
Microsoft Product Security Notification Service	advisory	Microsoft Corporation
SGI Security Advisory	advisory	Silicon Graphics, Inc.
Sun-alert	alert	Sun Microsystems, Inc.

Table 4: *Vendor Alert and Advisory Services Examples*

But can these various vulnerability services, tools, and databases, along with the software vendor's update announcements effectively combine to help you assess, manage, and fix your vulnerabilities and exposures? The short answer is that it used to be very difficult, but now a way to do it seems to be at hand. So what was wrong, and what changed?

The Tower of Babel

In 1998 if you tried to use these various tools, services, and databases you were faced with a problem rooted in each ones heritage. Each had developed its own naming standards and methods for defining individual entries in their respective vulnerability data stores. Table 5 shows how the same vulnerability was referred to by 12 different names by 12 leading organizations. With such confusion, it was very hard to understand what vulnerabilities were faced, and what vulnerabilities were or were not being looked for by each tool. Then the vulnerability or exposure still had to be mapped to the software vendor's name for the problem to get a fix.

Driven by our own attempts to develop an integrated picture of what was happening in our networks and in trying to select some new tools, The MITRE Corporation[6] started to design a method for working through the confusion of vulnerability and exposure information. This method was based on the creation of a reference list of unique names that would then be mapped to the appropriate items in each tool and database. In January 1999 the first public statement of our idea for a

Organization	Name used to refer to vulnerability
AXENT	phf CGI allows remote command execution
BindView	#107 – cgi-phf
Bugtraq	PHF Attacks – Fun and games for the whole family
CERIAS	http_escshellcmd
CERT	CA-96.06.cgi_example_code
Cisco Systems	HTTP - cgi-phf
CyberSafe	Network: HTTP 'phf' Attack
DARPA	0x00000025 = HTTP PHF attack
IBM ERS	ERS-SVA-E01-1996:002.1
ISS	http - cgi-phf
Symantec	#180 HTTP Server CGI example code compromises http server
Security Focus	#629 - phf Remote Command Execution Vulnerability

Table 5: *Vulnerability in the Tower of Babel*

reference list of unique names for vulnerabilities and exposures was presented at the 2nd Workshop on Research with Security Vulnerability Databases, held at Purdue University. MITRE presented a paper [1] at this conference that outlined the basic ideas and approach for what is today called the Common Vulnerabilities and Exposures (CVE) initiative.

Our vision for CVE was to provide a mechanism for linking together vulnerability-related databases or concepts – and nothing more (See Figure 1). Rather than viewing this narrow scope as a limitation, we saw it as an advantage. By agreeing to limit the use of CVE to the role of a logical bridge, we could avoid competing with existing and future commercial efforts. This was important, since it was critical that commercial organizations concur with the CVE concept and proceed to incorporate the initiative into their various products and services.

Figure 1: *CVE List as a Bridge*

By March 2001 the CVE effort had evolved into a, cross-industry effort involving more than 30 occupations in creating and maintaining a standard list of vulnerabilities and exposures. Almost half of the known vulnerabilities and exposures are either listed or under review, and presently 29 organizations are building nearly 50 products or services that use CVE names as a key element of their functionality.

How CVE Works

The CVE initiative is an international community activity focused on developing a list that provides common names for publicly known information security vulnerabilities and exposures. The CVE list and information about the CVE effort are available on the CVE Web site at cve.mitre.org/cve/.

The common names in CVE result from open and collaborative discussions of the CVE editorial board. This board, as shown in Table 6, includes members from numerous information security-related organizations around the world, including

Area of Expertise	Organizations
Academic/Educational	UC Davis, SANS, CERIAS
Network Security Analysts	Vista IT, Genuity
Other Security Experts	IBM Research, MITRE, Zero-Knowledge Systems
Intrusion Detection Experts	Silicon Defense, SANS
Tool Vendors	The Nessus Project, ISS, PGP Security-Network Associates, BindView, AXENT, CyberSafe, Symantec, NFR, Hiverworld, Harris, Cisco
Software Vendors	IBM, Sun Microsystems, Microsoft
Incident Response Teams	CanCERT, CERT/CC, DOD-CERT
Information Providers	NTBugtraq, Security Focus, National Institute of Standards and Technology (NIST), Ernst & Young, eSecurityOnline.com

Table 6: *CVE Editorial Board Composition*

commercial security tool vendors, members of academia, research institutions, government agencies, and other prominent information security experts. The board identifies which vulnerabilities or exposures will be included in CVE, then determines the common name, description, and references for each entry. The CVE name, for example CVE-1999-0067, is an encoding of the year that the name was assigned and a unique number N for the Nth name assigned that year.

MITRE maintains the CVE list and Web site, moderates editorial board discussions, and provides guidance throughout the process to ensure that CVE remains objective and continues to serve the public interest. Archives of board meetings and discussions are available for review on the CVE web site at cve.mitre.org/board/archives/. Other information security experts are invited to participate on the board on an as-needed basis, based upon recommendations from board members.

The key tenets of the CVE initiative are:

- One name for one vulnerability or exposure.
- One standardized description for each vulnerability or exposure.
- Existence as a dictionary rather than a database.
- Publicly accessible for review or download from the Internet.
- Industry-endorsed via the CVE editorial board and CVE-compatible products.

What Does CVE-Compatible Mean?

CVE-compatible is a phrase that indicates that a tool, Web site, database, or service uses CVE names in a way that allows for a cross-link with other repositories that use CVE names. To be CVE-compatible, the product, service, database, or Web site must meet the following three requirements:

- CVE Searchable: A user can search using a CVE name to find related information.
- CVE Output: Information is presented that includes the related CVE name(s).
- Mapping: The repository owner has provided a mapping relative to a specific version of CVE, and has made a good faith effort to ensure accuracy of that mapping.

Different products and repositories address different portions of the complete CVE list. For example, some might deal with UNIX, while others cover Windows NT. When looking at CVE-compatible items, you will need to evaluate them against your organization's specific needs in terms of platforms coverage and the software products that you use.

Why Use CVE-Compatible Products?

CVE compatibility allows you to use your vulnerability databases and tools together since they can "talk" to each other through shared CVE names. For example, if a report from a vulnerability scanning tool incorporates CVE names, you can quickly and accurately locate fix information in one or more of the separate CVE-compatible databases and Web sites to determine how to fix the problems identified by the vulnerability scanner. Also, with CVE-compatible tools, you will know exactly what each tool covers because the CVE list provides a baseline. Simply determine how many of the CVE entries are applicable for your platforms, operating systems, and commercial software packages, and use this subset to compare against the tool's coverage. Before the use of common names, it was extremely difficult to identify the vulnerabilities of your systems[7], or to determine whether a particular tool or set of tools covered them.

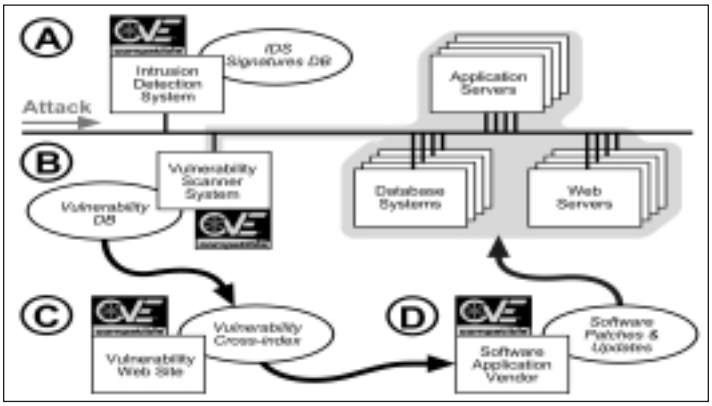
Improving the Process

The CVE effort is changing the way organizations use security tools and data sources to address their operational security posture. The example organization in Figure 2 is able to detect an ongoing attack with its CVE-compatible IDS system (A). In a CVE-compatible IDS, specific vulnerabilities that are susceptible to the detected attack are provided as part of the attack report. This information can then be compared against the latest vulnerability scan by your CVE-compatible scanner (B) to determine whether your enterprise has one of the vulnerabilities or exposures that can be exploited by the attack. If it does, you can turn to a CVE-compatible fix database at the software vendor or you can use the services of a vulnerability Web Site like ICAT Metabase[8], which lets you identify (C) the location of the fix for a CVE entry (D), if one exists.

Identifying Your Risk

Another thing you can accomplish with CVE-compatible products, that would be hard if not impossible to do before common names were adopted, is improve how your organization responds to security advisories. If the advisory is CVE-compatible it will include CVE entries. With that information you can see if your scanners check for these vulnerabilities, and determine whether your IDS has appropriate attack signatures for the alert.

Figure 2: *A CVE-enabled process*



Additionally, for systems that you build or maintain for customers, the CVE compatibility of advisories and announcements will help you directly identify any fixes from commercial software vendors in those systems (if the vendor fix site is CVE compatible). This is a much more structured and predictable process for handling advisories than most organizations currently possess.

Making Guidance Actionable

Earlier this year, a group of concerned security professionals put together a "Top 10" list [2] that outlined the most common, critical Internet security threats. The effort was orchestrated by the System Administration, Networking, and Security (SANS) Institute and brought together a consensus list from a wide variety of security experts. To help bring specificity and make the recommendations actionable, each of the top 10 suggestions had the appropriate CVE names, detailing each of the specific issue areas for a variety of platforms and products. A total of 68 CVE names were called out in the list of 10 threats.

Advanced Research Corporation	Internet Security Systems, Inc.
Alliance Qualité Logiciel	Max Vision Network Security/Whitehats
AXENT Technologies, Inc.	NIST
BindView Development	The Nessus Project
CERIAS/Purdue University	Network Security Systems
CERT Coordination Center	Network Security Wizards
Cisco Systems	NTBugtraq
Computer Security Laboratory, UC Davis	PGP Security, Network Associates
CyberSafe	Qualys
CYRANO	SANS
Ernst & Young	Security Focus, Inc.
Harris Corporation	Security Watch
Hivenworld, Inc.	Symantec
Intranode	Tivoli Systems Inc.
Intrusion.com	World Wide Digital Security

Table 7: *Organizations Developing CVE-Compatible Products*

Who Is CVE-Compatible?

While the list of organizations with CVE-compatible products is expanding, at this writing the vendors in Table 7 are those working toward compatibility. For a current list visit the CVE web site at cve.mitre.org/compatible/.

Today, there are several members of each type of tool, service, repository, and announcement capability that support CVE names. Underrepresented areas are vendor announcement and vendor fix sites; however, several vendors are actively discussing adding CVE names to their announcements. By the time you read this article there should be several vendors using CVE names in their announcements and alerts. In addition, like the CVE editorial board, the list of organizations working on or delivering CVE-compatible products has become international in scope.

Conclusion

The application of all known security fixes and patches is the complement of standard security protection mechanisms. Keeping current on fixes offers a robust method for keeping the commercial software that makes up your organization's software infrastructure healthy. Vulnerabilities and exposures will always be a part of our systems, as will the groups that find and share

information about vulnerabilities and exposures in commercial software. With common name integration and cross-referencing abilities emerging in vulnerability and exposure tools, web sites, and databases, it is becoming possible to deal with these mistakes and improve our systems' security. Handling security incidences is more systematic and predictable as CVE is supported within the commercial and academic communities. As vendors respond to user requests for CVE-compatible fix sites, the complete cycle of finding, analyzing, and fixing vulnerabilities will be addressed.◆

On-Line Resources

The on-line resources of this article contain hyperlinks to further references. For the full list please see page 32 of this on-lne version.

References

1. Mann, David E. and Christey, Steven M., Towards a Common Enumeration of Vulnerabilities, 2nd Workshop on Research with Security Vulnerability Databases, Purdue University, West Lafayette, Ind., Jan. 21-22, 1999.
2. Jackson, William, Top 10 System Security Threats Are Familiar Foes, *Government Computer News*, Jun. 12, 2000.
3. Sullivan, Bob, Hospital Confirms Hack Incident, MSNBC, Dec. 9, 2000.
4. Lemos, Robert, Power Play: Electric Company Hacked, ZDNet News, Dec. 15, 2000.
5. Mell, Peter, The ICAT Metabase, Computer Security Division at the National Institute of Standards and Technology, icat.nist.gov/icat.taf Dec. 19, 2000.

Notes

1. Vulnerability is a mistake that someone can directly use to gain access to things they are not supposed to have. An exposure is a mistake that gives that person access to information or capabilities that he or she can then use, as a stepping stone, to gain access.
2. A computer hacker broke into a hospital in the Seattle area and thousands of medical records were downloaded. The hacker's activities went unnoticed by the hospital, and when the hacker went public with his accomplishment, his claims were initially denied. The next day, the hospital confirmed the intrusion [3].
3. A Microsoft Web site was penetrated by a Dutch hacker through the Web server's "IIS Unicode" vulnerability that let him copy files, execute commands, and change files [4].
4. Unlike its original meaning that referred to a hacker as a prolific and inventive software programmer, hacking during the past few years has come to refer to the act of circumventing security mechanisms of information systems or networks. "Black-hat" hackers are those intent on doing harm, as opposed to "white-hat" hackers, who are usually working in support of organizations to help them assess and understand the vulnerabilities and exposures in their systems. Black-hat hackers are sometimes referred to as crackers.
5. As an alternative to tracking and recording each update, patch, and upgrade that gets applied to each platform in the enterprise, the use of vulnerability scanners is an attractive choice for monitoring the health of software applications. These tools are benefiting from the vigor of the market place's hunt for vulnerability information and the development of testing approaches that can turn up the presence of vulnerabilities or exposures in the "deployed" systems of an organization. However, due to "false positives," "false negatives," and incomplete coverage to date, these tools are not a panacea.
6. MITRE, working in partnership with government, is an independent, nonprofit corporation working in the public interest.
7. The CVE initiative is in the process of analyzing and categorizing all of the "legacy" vulnerabilities and exposures, and assigning them CVE numbers. Numerous members of the security vulnerabilities reporting and tracking community have donated their legacy databases to the CVE effort to support this effort.
8. The ICAT Metabase is a searchable index of computer vulnerabilities and exposures. ICAT is not itself a vulnerability and exposure database, but is instead a searchable index leading to vulnerability resources and patch information [5].

Did this article pique your interest?

Would you like to learn more about correcting vulnerabilities and exposures in commercial software that is used to develop your organizations infrastructure? Then attend the Thirteenth Annual Software Technology Conference 2001 on April 29-May 4 in Salt Lake City. Robert A. Martin will speak on this topic in Track 9 on May 2.◆

About the Author

Robert A. Martin is a co-lead for MITRE's Cyber Resource Center Web-site, and a principal engineer in MITRE's Information Technologies Directorate. At the culmination of his five years of Y2K leadership and coordination efforts, Martin served as the operations manager of the Cyber Assurance National Information Center, a 24x7 cyber security watch center within the President's Y2K Information Coordination Center. Today, Martin's efforts are focused on the interplay of cyber security, critical infrastructure protection, and e-Business technologies and services. Martin received a bachelor's degree and a master's degree in electrical engineering from Rensselaer Polytechnic Institute and a master's of business degree from Babson College. He is a member of the ACM, AFCEA, Institute of Electrical and Electronics Engineers (IEEE), and IEEE Computer Society.

Robert A. Martin
The MITRE Corporation, MS B155
202 Burlington Road
Bedford, MA 01730-1420
Voice: 781-271-3001
E-mail: ramartin@mitre.org

The Ideal Collaborative Environment

P. A. Dargan
Litton-TASC Inc.

What immediately springs to mind when discussing cross-organizational electronic collaboration are the capabilities of today's collaboration tools: e-mail, bulletin boards, chat sessions, and virtual rooms. However, today's tools provide only a framework for the ideal environment. This paper builds on current collaboration capabilities to describe the Intelligence Community Collaboration Forum's vision for the ideal collaboration environment and the challenges in achieving it.

The Intelligence Community Collaborative Operations Network (ICON) Program Office was established in December 1998 to facilitate collaboration and the free flow of information across the intelligence community (IC). As its main starting point, the ICON Program Office initiated an IC collaboration forum in March 1999 to identify collaboration programs and systems within the IC and Department of Defense (DoD) and to achieve a consensus on common goals. During its first year, more than 150 IC members attended the forum. Its membership included representatives from the majority of intelligence agencies as well as the DoD. This paper highlights key findings of forum sessions from its inception to the present, defines common terms, and describes today's electronic collaboration capabilities, and the ideal collaboration environment for the future.

Current Collaboration Capabilities

At the forum outset, participants discovered a need for a common framework of collaboration terms. Basic terms such as "collaboration" and "interoperability" held different meanings for participants depending upon their organizational background. This section defines key terms.

Collaboration is "the process of shared creation: two or more individuals with complementary skills interacting to create a shared understanding that none had previously possessed or could have come to on their own [1]." The following list defines levels of collaboration:

- Level 1: At a basic level, individuals operate independently and interact to accommodate their own specific needs passing documents back and forth and sharing information, but not as part of a working group or team [2].
- Level 2: At the next level, a group of individuals exchange information as part of a community of interest, but not to achieve a common goal [2].
- Level 3: At the highest level, collaborators operate as a team to achieve a common purpose by working together and gain ing new insights [2].

Asynchronous collaboration allows data (e.g., a message) to be sent as soon as it is ready, regardless of whether recipients are ready to receive it. Asynchronous collaboration includes e-mail with attachments, threaded discussion databases, bulletin boards, and persistent electronic "rooms" where members can store and access common documents and files at their convenience. Table 1 lists asynchronous collaboration capabilities. Asynchronous collaboration capabilities emphasize support for collaboration levels one and two.

Bulletin Board: central electronic repository that allows users to post information for members to share.
E-mail: a network service that allows users to send, create, receive, view, store, and forward messages.
People Locator: ability to find other user's userids to send e-mail.
Group Calendars: ability to share calendar information between different users.
Threaded Discussions: asynchronous postings to which people can link responses.
Virtual Persistent Workspace (Virtual Room): a permanent, networked environment where a group of users share expertise and contribute to solving problems via virtual teaming activities.

Table 1: *Asynchronous Collaboration Capabilities*

Synchronous collaboration manages (synchronizes) the sending and receiving of dynamic text, audio, and video, such that only a single user can dominate a discussion at a time. However, a group of users can work together to share ideas in real-time. This mode of collaboration is also referred to as "real-time collaboration." It supports collaboration at the highest level and includes text-based chat sessions, electronic whiteboards, awareness knowledge, and live audio/video conferencing.

Audio/video conferencing: audio/video session where users are able to see and hear each other electronically on a desktop computer.
Audio: real time aural information from a microphone to a speaker.
Video: real-time visual information from a camera.
Awareness: mechanisms that allow users to know who is on-line.
Chat: real-time sharing of text data (i.e., real-time e-mail; "instant" messages).
Shared applications: ability to view and control an application on another's desktop.
Virtual Teams: individuals forming and acting as a team by means of electronic collaboration tools or information services to accomplish a mutual goal, regardless of location.
Whiteboard: shared drawing board analogous to a "chalk board" that may be blank or display an image that a group of users can individually mark up and review in real-time.

Table 2: *Synchronous Collaboration Capabilities*

Point-to-point and multi-point communication are the two modes of communication provided for synchronous collaboration. Collaboration tools that feature point-to-point communications require each computer system to handle all of its communication with other computer systems involved in a synchronous collaboration session. Point-to-point tools are effective only for a small group of users because communication bandwidth can rapidly become saturated as more users join a session. Collaboration tools that feature multi-point communication are applicable for small and large groups of users, since they provide server capabilities that manage bandwidth and facilitate communications between user platforms.

The International Telecommunications Union has defined several communications standards used by collaboration tools for both multi-point and point-to-point communications. H.323: defines audio and video conferencing protocol for networks based on the Internet protocol (IP). T.120 provides standards for data-conferencing capabilities, including application

sharing, electronic whiteboard, chat, and file transfer functions.

Commonly Used Collaboration Tools

- *Collaborative Virtual Workstation* (CVW): This is a software prototype developed by MITRE that supports a collaborative environment optimized for supporting persistent, geographically dispersed virtual rooms. CVW provides chat, audio/video conferencing, application sharing, electronic whiteboarding, and multi-point communications. At the time this paper was written, MITRE was looking for a vendor who would assume responsibility for managing and improving the software [4].
- *Information Workspace* (IWS): General Dynamic developed IWS as a Web-based, collaboration environment featuring virtual rooms, audio/video conferencing, chat, electronic whiteboarding, and application sharing with multipoint communications.
- *Microsoft NetMeeting*: A Microsoft product that supports point-to-point communications for its audio/video conferencing, chat, application sharing, and electronic whiteboarding.
- *IBM Lotus Sametime:* A Lotus product that interfaces with most Web browsers and provides audio/video conferencing, chat, application sharing, electronic whiteboarding, and awareness with multipoint communications. Considered by Giga to be the leading collaboration tool [5].

The Ideal Collaboration Environment

While e-mail, bulletin boards, chat sessions, and virtual rooms are powerful features to enable cross-organizational electronic collaboration, they are only initial capabilities.

This section builds on existing collaboration capabilities to describe the forum participants' vision of the ideal collaboration environment. The capabilities are not described in any particular order because participants considered each one critical for the ideal environment.

Following each capability, the author discusses technical challenges to overcome before it can become a reality.

Capability No. 1: Rapidly find the right people with the right expertise.

To enable collaboration, users are able to quickly identify, locate, and contact specific staff or subject-matter experts from community organizations, academia, and industry; they are provided with detailed information such as the contact's full name, job title, organization, phone number(s), pager number(s), address(es), levels of access, and areas of expertise. For example, the collaboration system could display a menu that enables users to select from technical subject-matter areas such as telecommunications, data warehouses, parallel processing, and enterprise portals to find experts from industry, academia, DoD, or another IC organization, along with sufficient contact information. A user could also search for names with only partial information. For instance, if a user knew part of the last name, "Bran," and that he was located at the Pentagon, the collaboration system could perform a partial name-string search for all Army personnel at the Pentagon and return a list of possible matches.

Challenges

The real challenge is being able to identify and locate experienced personnel and subject-matter experts from external organizations since each employs staff with particular areas of expertise. To achieve this capability, it is necessary to link each organization's directory through a mechanism like a meta-directory. Today's collaboration tools rely on the directory services provided by the network or third-party vendor products. The schema can often be tailored to provide additional information about staff. For example IBM provides directory services for Lotus Sametime through Domino, another Lotus product. However, because most commercial products are not interoperable, organizations need to consider using either the same directory service products, or defining point-to-point customer interfaces that interconnect the directories.

There is another challenge: Collaboration tools need open Java ports to exchange data using communication protocols such as User Datagram Protocol (UDP). Because firewall systems have difficulty filtering UDP, especially for malicious code and new viruses, organizations may not be willing to open these ports for collaboration.

Currently, two standards compete to support directory service protocols: Lightweight Directory Access Protocol (LDAP) and X.500. The protocol standards are not compatible, although custom interfaces can be developed to exchange packets between the various commercial products used by different organizations. Vendors have begun developing software to implement meta-directory mechanisms, generally using LDAP as the preferred standard. However, they are only in the preliminary stages of developing the required capabilities.

A number of vendors united recently to establish a directory services standards consortium, the Directory Interoperability Forum, to achieve the vision for unified, cross-organizational, open standard directory services. Its membership includes IBM, Lotus, Novell, Oracle, Cisco, Sun-Netscape, Citrix, and Unisys [6].

Capability No. 2: Quickly organize and conduct virtual teams and meetings.

This capability builds on capability No. 1 by organizing key personnel and subject-matter experts from around the world into virtual teams as soon as the need arises, coordinating and conducting ad hoc and formal virtual meetings using electronic collaboration tools. The collaboration system could provide a set of virtual conference rooms to conduct a desktop teleconferencing session at any time of day. Virtual conference rooms could be scheduled so that sessions could be conducted at an appointed time. During the session, the collaboration system could project team member images talking and sitting around a conference room table – just as if the session was being held in a video teleconference room – even though each desktop teleconferencing user was remotely located.

Of critical concern is the ability to contact an individual in another organization when a crisis occurs. Lotus' Sametime and AOL's Instant Messenger provide a feature that will instantly notify on-line users that “John Doe” wants to talk to them If he/she is not on-line, Lotus Sametime will phone the individ-

ual and/or page him/her to obtain a timely response. This should be a standard feature in collaboration tools.

Challenges

The most conspicuous challenges relate to providing sufficient communications connectivity across all organizations, quality of service, and trusted security mechanisms to support virtual meetings. Unless the same collaboration tools are used by each organization, conducting virtual meetings with attendees from various organizations is simply not possible today. Further, many organizations do not provide the necessary transmission speed, reliability, and communications bandwidth necessary to avoid problems with spotty audio quality or choppy video. Users want phone quality audio and smooth video.

Many vendors are unable to supply details on how many concurrent audio/video conferencing and chat sessions can occur simultaneously, and how many users it would take to significantly degrade performance. These are significant questions concerning service quality. More businesses are engaging in cross-organization virtual meetings through information provider (IP)-based networks – and industry is pushing the envelope. Vendors will be forced to define more efficient data compression algorithms for convergent networks where data, voice, and video are carried over the same transport mechanism, whether they are wireless, optical, coaxial, or some entirely new innovation.

Forum participants said that as information exchanges with foreign personnel and academic experts increase, the need for language translation tools will become more prominent. Commercial speech-recognition translation modes include speech-to-text, text-to-text, and text-to-speech. However, commercial tools – no matter what their level of sophistication – have not achieved direct speech-to-speech translations. Vendors such as Lernout and Hauspie [7] offer a suite of tools that can be combined to translate from speech-to-speech using the incremental steps. The tools can be integrated with the leading commercial collaboration tools. But language translation tools have a long way to go before they achieve highly accurate translations for all languages and dialects.

Capability No. 3: Enable cross-organizational collaboration to support the business lifecycle.

This capability gives interagency and internal staff the ability to brainstorm together, exchange insights, and develop products electronically according to business workflows that unify related work activities to enhance the timeliness of product delivery. For instance, the collaboration system would include a workflow tool that automatically tracked the status of an intelligence product developed by several organizations through its origination, revision, approval, release, and final dissemination.

Challenges

Modern vendor group-ware/workflow tools can be readily adapted to coordinate well-defined business processes such as logistics, proposal development, and retail services. However, these tools are not easily adapted for highly complex activities and business rules such as those represented by the community's business life cycle.

In addition, the lack of interoperability across available tools poses another viable concern when individual organizations come to rely on different tools. Industry will have to develop more sophisticated workflow tools than currently available. The tools will also need to include expert system capabilities to implement the community's business rules in the workflows.

Meanwhile, collaboration tools that enable personnel from different organizations to develop a joint product raise new questions: Who becomes the product originator? Who owns it? Who is responsible for updating and maintaining it? The answers to these questions are critical to determining where it should be stored, how it should be stored, when it should be archived, who should be able to review it, who can access it, how data conflicts should be resolved, and who should update it. Organizations need to develop common rules of engagement for procedures on handling information management and configuration management issues related to joint products.

Capability No. 4: Build, find, and exchange information across organizational boundaries.

Forum participants described an environment that uses rapid, intelligent search tools across organizational servers to find information that looks for keywords within their required context, pruning the search space to provide a few relevant matches. The collaboration system would allow users to use or define the context of a keyword or phrase. For example, a search for White House news could apply specific search criteria such as public newscasts about the president, vice president, first lady, first family, and White House staff.

Challenges

There are three major challenges: more powerful search and retrieval tools; immediate access to information; and mechanisms that effectively link cross-organizational products, databases, and information. Current search and retrieval tools typically provide hundreds of meaningless keyword matches, often too many to review. Companies are beginning to develop "intelligent" search tools that use clustering mechanisms to select matches from documents then show paragraphs (rather than partial sentences) where a match occurred. Keyword/context searches are more difficult and require a knowledge base association with keywords. Associative databases are being investigated as one means of improving search routines for keyword/context matches.

Immediate access to information, especially as it grows beyond terrabytes, will remain a significant challenge. Several mechanisms are in use to speed information access: networking caching technologies and storage area networks (SANs). Companies such as Akami and Inktomi provide network caches composed of hardware and software to store web pages and other information for rapid retrieval. SANs employ a high-speed network (via fibre channel) that interconnects storage devices enabling users to share devices through network servers. These are just the beginning of innovations as network technologies experience vast performance improvements, and network appliances provide more sophisticated services.

To consolidate and display information, metadata models are employed to interconnect cross-organizational servers, databases, products and information of different organizations. But metadata models do not resolve data inconsistencies and conflicting data, nor fuse information from different sources. As time progresses and metadata modeling approaches are more mature, best practices will exist for developing models that take care of data inconsistencies and conflicts, present the information in an understandable format, and employ expert knowledge to fuse information to improve reporting accuracy.

Capability No. 5: Deliver the right information to the right people as soon as it is available.

Participants described a "smart product delivery" capability that profiles users to deliver targeted information considered relevant to their areas of interest even if it is not specifically requested. In addition, the information is provided as soon as it is received in the required user format that is compatible with his or her local applications. For example, a user asks to be notified when the London Stock Exchange drops to a certain level and to be provided with a list of companies affected. The collaboration system could respond with a list of firms that the user's profile indicated would be of interest such as major banking institutions and large, high technology firms. Additionally, the information could be provided in a preferred format such as Word Perfect or Microsoft Word.

Challenges

Existing digital library tools can filter information from news broadcasts and other information sources to alert users as soon as a keyword match occurs. As mentioned earlier though, these tools present the same retrieval challenges – search tools provide hundreds of matches to sort to find a meaningful match. This capability builds on the powerful search and retrieval features described with capability No. 4. In addition, unless a user requests specific information, it is highly unlikely that the tools will be able to find related useful items.

Vendors have already developed knowledge discovery for marketing purposes. These tools automatically use attributes such as a consumer's demographics and use of product X to determine whether he or she might be interested in a product Y, and subsequently send him or her product Y information. Again cross-organizational communications connectivity is required here so that products can be automatically disseminated to users.

Because applications vary, it is important to provide the product in required format. Many desktop publishing tools generate files in other tools' formats, but this is not universally true. Directory service vendors are considering adding user profiles in the directory service products, so that product files will be generated automatically in the format required by a user's application.

Capability No. 6: Provide and maintain sufficient security.

Security systems need to be proactive: denying unauthorized access, detecting and disabling intrusions before damage or compromise occurs, and protecting systems from malicious code and viruses. An advanced security system such as biometric recognition could be used here to reduce the potential for unauthorized access, or a system monitor that identified suspicious system behavior based on a user profile. Here a user logging onto the system at midnight on a Friday would raise a red flag if the user's typical work schedule for the past year has been weekdays from 7 a.m. to 5 p.m.

Challenges

Since commercial tools that provide information security services (e.g., virus checkers, firewalls, and intrusion detection systems) are immature, there are protection gaps:

- An ability to eavesdrop on individual conversations via unauthorized, remote access of desktop system microphones.
- The ability for imposters to send deceptive messages and data or to participate in a collaboration session without discovery.
- A difficulty handling excessive denial-of-service actions, causing the system to crash.
- The inability to detect and protect from intrusions that exploit operating system weaknesses.
- A difficulty detecting malicious code and new viruses that enter through message attachments and other means.
- A difficulty detecting spoofing.

The introduction of Public Key Infrastructure for public and private certificates to access organizational applications, systems, devices, and data is a starting point for resolving some of these problems. But collaboration tools will need to be modified to take advantage of them. Industry is also improving biometric recognition and encryption mechanisms, but firewall services are inadequate, seriously lagging behind the ingenuity of hackers to create new damaging viruses and malicious code.

Capability No. 7: Employ technology and community standards.

The need for industry standards and commercial standards-based products is seen as the single most important factor for enabling application and data interoperability. But where industry standards are lacking, organizations need to define their own standards. The ideal collaboration environment provides services using standards-based products.

Challenges

Due to their immaturity, it will be several years before vendors understand what features collaboration tools require, including defining common physical standards, more sophisticated displays and user interface, security, metadirectories, and service quality. In the meantime, it is important to concentrate on monitoring vendor consortia developments in the standards arena and to procure commercial products that feature widespread marketplace acceptance and show potential as open standards. For a more detailed discussion on challenges for open standards, refer to [8].

Conclusion

Both non-technical issues and electronic collaboration tools must be considered since increasing the information flow dramatically increases the potential for error. For example, the IC

has typically operated in a need-to-know environment whereas the Internet provides openness and public access. What is the delicate balance between these opposing operating modes?

Collaborators are also responsible for assessing the expertise of information from others before basing conclusions on new information sources. This trust poses a challenge for virtual teams whose members are remote and unknown: how to develop trust among unknown staff to support developing joint products? Furthermore, the IC has traditionally rewarded individuals for independent product development. What reward structure would motivate individual staff to cooperate with team members across the intelligence community? Hall conducted a study on collaboration in the community that uncovered barriers to collaboration and discussed such concerns [9].

Yet as these technical and non-technical issues are addressed to achieve the ideal collaboration environment, users will forget what it was like to collaborate simply by phone, video teleconference centers, and physical meeting rooms. ◆

References

1. Schrage, Michael, *No More Teams! Mastering the Dynamics of Creative Collaboration*, DoubleDay, 1995.
2. Schrage, Michael, *Shared Minds - The New Technologies of Collaboration*, Random-House, Inc. New York, 1990.
3. Advancing Directory Standards, White Paper, The Directory Interoperability Forum, www.directoryforum.org
4. See collaboration.mitre.org
5. Rasmus, Daniel, Knowledge Management Report, Giga Information Group, Norwell, Mass., December 1998.
6. Advancing Directory Standards, White Paper, The Directory Interoperability Forum, www.directoryforum.org
7. See www.lhsl.com
8. Dargan, P. A., Best Practices for Open System Challenges, CROSSTALK, November 1997.
9. Hall, Tamra, CIA's Baseline Study for Intelligence Community Collaboration: Final Report - December 1999, Information Sharing Solutions Office of Advanced Analytic Tools, Central Intelligence Agency, 1999, collaboration.mitre.org/prail/IC Collaboration Baseline Study Final Report/toc.htm

Did this article pique your interest?

Would you like to learn more about collaboration environments and implementing the electronic tools to create the ideal model? Then attend the Thirteenth Annual Software Technology Conference 2001 on April 29-May 4 in Salt Lake City. P.A. Dargan will speak on this topic in Track 6 on May 2. ◆

"As a rule, software systems do not work well until they have been used, and have failed repeatedly, in real applications."

DAVE PARNAS

About the Author

P. A. Dargan is a principal member of Technical Staff at Litton-TASC with more than 21 years of experience in all aspects of software system engineering, including university software engineering instruction. She specializes in developing open architectures and migration strategies based on open system standards. She is an international guest speaker and invited author on open systems. Dargan has a bachelor's degree in mathematics from Virginia Polytechnic and State University and a master's degree in computer science from George Mason University.

P. A. Dargan
Litton-TASC Inc.
4801 Stonecroft Blvd.
Chantilly, VA 20151
Fax: (703) 449-3400
E-mail: pdargan@erols.com; padargan@tasc.com

The Potential of Extensible Markup Language

David A. Hayes
Space and Missile Defense Technology Center

The Department of Defense should leverage commercial best practices as it upgrades and extends its Battle Management, Command, Control, Communications, and Computers/Intelligence (BMC4I) architecture. The family of technologies associated with the Extensible Markup Language (XML) is the modern foundation of some of these best practices. This article describes XML and its potential application to BMC4I.

The origin of computer markup languages such as Extensible Markup Language (XML) is in the publishing processes that we have employed for centuries. Prior to the invention of computers, markup was simply the editorial comments that a copy editor made on a report or article often in the margins of a paper. The markup added information to the paper's content by specifying format or other meaning. In modern computer science, any information or markup added to a document that specifies the meaning of its content is known as metadata. In turn, any language used to markup a document is known as a metalanguage.

The first computer metalanguage, Standard Generalized Markup Language (SGML), was standardized in 1986, a long time ago in computer-years. SGML provided a standard means to separate content and format in documents of all kinds. Such separation is very useful in applications where information is presented on a variety of media for which no single format is adequate. Early adopters of SGML include the IRS, Patent Trademark Office, SEC EDGAR Database, and Army SGML Registry and Library (ASRL) for its technical manuals.

Next came a series of standards for the HyperText Markup Language (HTML) about a decade ago. Most readers will recognize HTML as the language of the Internet's very popular World Wide Web. The HTML is a simple metalanguage with a rigid syntax designed for the presentation of a common class of office or technical report, with headings, paragraphs, lists, illustrations, etc. In addition HTML has limited support for multimedia. The success of these metalanguages set the stage for the development of the next generation, XML, which is an evolution of SGML without the limitations of HTML.

Extensible Markup Language

The World Wide Web Consortium (W3C) released its proposal for XML in 1996, and approved the standard in 1998. From the start, XML has received an extraordinary amount of attention from public and private industry. Much of this attention did not stem from XML's immediate contributions but from its potential for future contributions.

XML is a standard markup metalanguage for describing, archiving, and communicating digital information. XML is a method for putting structured data in a text file. Therefore, it is readable by both man and machine.

An XML document looks like HTML, but isn't HTML. Both XML and HTML use tags and attributes. Tags are words bracketed by the delimiters, "<" and ">". Tags may contain attributes of the form "name=value." The rigid syntax of HTML specifies what each tag and attribute means and often how the text between them will look in a browser display. XML uses tags to delimit pieces of data, and leaves the interpretation of the data to the application that reads it. As a result, XML-formatted documents can be readily displayed or used by a variety of applications, not just a limited number of Web browsers.

To assist in an application's interpretation of XML, the Defense Information Systems Agency (DISA) provides namespace registry services for XML metadata for BMC4I domains such as Ground Operations, Aerospace Operations, and Geospatial Imagery etc.

The registry diides.ncr.disa.mil/xmlreg/index.cfm enables the consistent interpretation and use of XML, both vertically within a system and horizontally across systems. (The namespace registry is another member of the family of XML technologies.)

Like HTML, XML can use procedural languages to implement applications that further define interfaces, manage data, and permit greater interoperability. Java and JavaScript are two common network-friendly procedural languages used with XML.

Unlike HTML, XML does not contain format information. If XML information is presented, then a style language such as the Extensible Style Language (XSL) defines the presentation format. (The XSL is another member of the family of XML technologies.) This separation of content and format means that an XML-based BMC4I system will not break with each application or new presentation media.

The tags in an XML document delimit and define data. The XML is flexible in that tags can be created to communicate any digital data including text and multimedia. The XML is a metalanguage because its tags may contain metadata, or extra information about message data, with no predefined syntax.

Table 1 is an example of an XML data island. You can observe many qualities of the language in this example. Data elements consist of a start tag (bracketed by "<" and ">"), some data or other data elements, and an end tag (bracketed by "</" and ">"). For every start tag there is an identically named end tag. Other data elements may be nested in as many levels as desired within a data element. Nested data elements must be completely nested within its parent's tags. Metadata may be specified in the start tag. Metadata attributes are completely enclosed in double-quote marks. These qualities are why XML is known as a well-formed language.

Here is an example that illustrates the benefit of metadata. The following United States Message Text Format (USMTF) line simply presents a remark: RMKS/179 248//. The following XML line says the same thing — and more: <location_target>179 268</location_target>. The XML line makes it clear to an extraction routine or human reader what the data between the tags means. The number is not a Canadian zip code or

```
<XML ID="150300 Weather">
<weather-report>
    <date>25 March 2000</date>
    <time>08:00</time>
    <area ZIP="35807">
        <city>Huntsville</city>
        <state>Alabama</state>
        <region>South Central</region>
        <country>USA</country>
    </area>
    <measurements>
        <skies>partly cloudy</skies>

        <temperature>36</temperature>

        <wind>
            <direction>SW</direction>
            <windspeed>6</windspeed>
        </wind>
        <humidity>87</humidity>
        <visibility>10</visibility>
    </measurements>
</weather-report>
    </XML>
```

Table 1. *Example of an XML Data Island*

some other interpretation of the USMTF line. The XML line clearly describes the number as a target coordinate. Thus, metadata gives meaning to XML data.

There is a price for this extra information. The XML metadata in the form of data tags creates a transmission overhead. The metadata imposes an added burden on network capacity and processing to parse the message. Data compression, local computation and manipulation of data, intelligent communication of knowledge as opposed to raw data, and granular updates are mechanisms that mitigate the network load penalty.

An uncompressed XML message is significantly larger than an MTF version of the same message. This is not a problem. The XML can be compressed for more efficient network transmission. Such was the case in the United States-led multinational global command, control, communications, and intelligence 1999 Joint Warfighter Interoperability Demonstration. In that demonstration XML was compressed via a smart compression utility (XMill by AT&T). Dr. Robert Miller of MITRE reports that XML-based Air Tasking Orders (ATO) were actually smaller than the original ATOs in compressed MTF format as indicated in Table 2. Compressed XML and MTF files were found equivalent in size in the 2000 JWID. Therefore, we can conclude that larger XML messages do not stress network capacity more than current formats.

XML does not ensure data compatibility with a given application. Therefore, interoperability standards are required just like previous data exchange methods. Developers of any Defense Information Infrastructure Common Operating Environment such as BMC4I systems must express XML with a standard lexicon and grammar. The previously described XML namespace registry can provide the necessary standardization of XML tags. It is a vital component in the implementation of any metalanguage.

Table 2. *ATO Compression*

File Type	Size (bytes)
MTF-ATO	300K
MTF-ATO (compressed with Pkzip)	72K
XML-ATO	2.2M
XML-ATO (compressed with XMill)	46K

XML, like any text, can be easily encrypted for secure communications. Significant security concerns can be addressed by using the Secure Internet Protocol Router Network , or commercial secure sockets layer technology.

Commercial concerns are rapidly developing the necessary tools and applications to realize the potential of XML in electronic commerce. The capabilities of these commercial tools and applications will drive down the cost of XML development allowing more affordable military-specific implementations.

Many XML applications are inexpensive, powerful, and rapidly evolving. The XML development is driven by the dynamics of the commercial marketplace. As such, XML-based products enjoy an efficiency of effort normally associated with a competitive commercial market. The development of XML-based BMC4I systems can leverage these commercial efficiencies. XML is license free, platform independent, and well supported.

XML Potential for BMC4I

Future XML-based BMC4I systems will likely focus on two areas: The first deals with acquiring information from disparate (often *legacy*) sources used by military systems. The second focuses on distributing dynamic content to users. The Internet community has adopted two words to describe these tasks: aggregation and syndication. The XML's information-exchange prowess applies to both of these tasks.

Aggregation is the process of collecting, organizing, and integrating data from disparate sources. One potential application for XML BMC4I aggregation is in the creation of a Global Family of Interoperable Integrated Air Pictures (GFIIAP) used for Air and Missile Defense. In the case of GFIIAP, legacy databases of satellite orbits can be aggregated with in-theater sensor data to form a more complete and accurate picture. Unexpected deviations of the data stream from any source can be processed based on embedded metadata in XML tags. The metadata can also help clarify issues of time latency, track correlation, and fusion.

An XML aggregation application could be developed to organize references to metadata rather than the actual data itself. The application would collect and build a repository of metadata links, which are in turn archived in XML format. Intelligent agents working with this repository would expedite searches, access, and distribute data as needed. Another set of artificial intelligence agents would gather input from a variety of sources, including databases, XML, Tactical Digital Information Link (TADIL), and Message Text Format (MTF) documents.

Syndication is the process of disseminating data. On a BMC4I network, syndication is often seen in the form of information that participants get from third party information providers. For example, a tactical operations center (TOC) might request information from a weather forecast database to pass on (perhaps aggregated with other information and appro-

priately tagged) to a weapon system. This information is syndicated from this weather database. In this scenario, the weather database is the syndicator or third party, the TOC is the subscriber, and the weapon system is the user.

A BMC4I syndication tool set could be developed to define, extract, and publish syndicated information from large frequently updated databases. The tool set would make information available to subscribers through XML used to describe the structure and content of syndicated information. Information would be distributed by using two complementary components. The first component lets syndicators define the information that the subscriber wishes to receive, i.e. all satellite tracks over a theater of operations in the next hour. The second component reads this definition and performs the actual information extraction and distribution to the subscriber TOC. The TOC could aggregate the information locally and provide it to users.

In addition, XML can be used as a tactical language "Rosetta Stone." XML can translate or encapsulate common military communication languages in use today such as TADIL A/B/J, MTF, and other languages.

Conclusion

The Joint Technical Architecture (JTA) mandates the XML 1.0. W3C Recommendation, 10 February 1998 (Reference: REC-xml-19980210, www.w3.org/TR/1998/REC-xml-19980210) for domain- and application-specific markup languages defined through tagged data applications. The JTA states, "This allows new capabilities to be defined and delivered dynamically."

XML conveys complex information while retaining enough flexibility to accommodate future modifications to the message content in ways that are unforeseen today. Therefore, it is ideal for supporting rapid prototyping, evolutionary, and spiral development of evolving BMC4I systems.

The most important benefit of XML as it applies to BMC4I is that it advances and develops good information management practices that are responsive to modern computer technology and standards. XML is not always the best solution, but it is always worth considering.◆

Notes

1. A "two-line" satellite element set actually has three lines. Line zero is a twenty-four-character name consistent with the name in the North American Air Defense Satellite Catalog celestrak.com/NORAD/documentation/tle-fmt.html

2. Mil-Std 6016A is the TADILJ/Link 16 Message Standard

Additional On-Line Readings

1. diides.ncr.disa.mil/xmlreg/index.cfm
2. diides.ncr.disa.mil/shade/index.cfm
3. www.computer.org/proceedings/meta/1999/papers/21/rdaniels.html

Acknowledgements

The author would like to thank Leo Kimminau, and Dr. Robert Miller for their contributions and assistance.

About the Author

David A. Hayes is cheif of the Computer Technologies Division of the Advanced Technology Directorate of the Space and Missile Defense Technology Center in Huntsville, Ala., where he manages small business and innovative research developments. He is a subject matter expert for BMDO Global Defense BMC4I and the Army's Space Control BMC4I programs. He holds a bachelor's degree in electronics engineering from the University of Tennessee, in Knoxville, and a master's degree in management from the Florida Institute of Technology.

David A. Hayes
U.S. Army Space and Missile Defense Command
Commander U.S. Army Space and Missile Defense Command
Attn: CSSD-TC-TD-AS, David Hayes
P.O. Box 1500
Huntsville, AL 35807-3801
Phone: 256-955-3340
Fax: 256-955-1432
E-mail: David.Hayes@smdc.army.mil
http://users.liveonthenet.com/~dhayes/

Web Sites

Common Vulnerabilities and Exposures

www.cve.mitre.org

The Common Vulnerabilities and Exposures (CVE) site is a list of standardized names for vulnerabilities and other information security exposures. CVE is a dictionary, not a database, which makes it easier to share data across separate vulnerability databases and security tools. CVE content is a collaboration of the CVE editorial board including representatives from security-related organizations, academic institutions, government, and other security experts.

Organization for the Advancement of Structured Information Standards

www.oasis-open.org

OASIS, the Organization for the Advancement of Structured Information Standards, is a non-profit, international consortium that creates interoperable industry specifications based on public standards such as Extensible Markup Language (XML) and the Standard Generalized Markup Language (SGML), and others related to structured information processing. OASIS members include organizations and individuals who provide, use, and specialize in implementing the technologies that make these standards work in practice.

TechNewsWorld

www.technewsworld.com

TechNewsWorld.com is a real-time news service, updated 24-hours a day, in multiple languages. It provides a central resource for technology news links from around the world. In addition to headlines, TechNewsWorld.com also analyzes each news article providing small excerpts and a hyperlink to the original news article, as well as links to related stories from many other sources.

Does Your Internal Management Meet Expectations?

Eli Schragenheim,
ELYAKM Management Systems

H.William Dettmer,
Goal Systems International

Projects that fail to meet expectations often began with an ineffective plan. Estimating task duration in projects is one of the chief culprits. Safety time included in task estimates is nearly always squandered, rendering its value nearly useless. Parkinson's Law, the "student syndrome," and multi-tasking all conspire with natural dependencies between activities to effectively assassinate schedules. A new solution to the planning problem is Critical Chain Project Management (CCPM). CCPM can provide about 90 percent confidence of finishing projects at or before planned times, which are inevitably shorter than traditional critical path schedules as planned originally. Buffer management provides warning of danger to the delivery schedule early enough to apply less extreme corrective measures. Schedule, cost, and performance are all enhanced.

More than 20 years ago in a programming class, the professor for system analysis told us that a delay equal to 100 percent of the planned time is certainly within the normal range for any software project.

Is it much different today? At that time we already knew the professor's statement was true, but the question is "Why?" It is easy to accept the idea that young programmers might be overconfident about their ability to write the code in almost no time. It's also easy to understand why a young programmer might think that debugging time might approach zero. However, the reality of software development during the past 20 years should have taught the industry some lessons. There is no shortage of statistics on the ratio between the time it takes to write the code and the time required to realize a working prototype. A lot is known about software development – after all, the disappointments of the past should have made some kind of impression on developers. So why are we continually disappointed? By disappointment we are referring to *internal management expectations,* not our promises to the clients.

Whenever there is a gap between initial expectations and the real world, it should prompt us to review both the way we set our expectations, and the way we try to meet them. When we constantly fail to meet our expectations, we cannot simply justify it by lack of experience or by the significant amount of uncertainty involved. We *do* have the experience, and we are capable of approximately estimating the impact of the uncertainty. Generally speaking, uncertainty should reflect itself in both delays and early completions. Though we may sometimes be disappointed, we should also have experiences where projects finish earlier than expected. But, is this a common occurrence?

Setting Expectations

Let us first review the process of setting expectations. Suppose you are the CEO of a small software company. You want to add a new module to an existing application that will verify the sensitivity of some processes to certain random variables. This module will rely on your current database, which contains historic data. All you are asking is for the module to apply the appropriate statistical formulas and generate a report. Nothing very complex. The technology is understood and the needs are clear. You present your request to your chief programmer and ask how many people he needs, and for how long. The chief programmer, after consulting with his people, returns with an overall estimate of six man-months, and three months actual project duration before a prototype is ready for beta-testing with clients.

Would you be surprised if the prototype is not ready until *nine* months, instead of three? Would you be surprised if it was finished in exactly three months? Could it take a mere two months?

What does the chief programmer mean by three months of total project time? Certainly, the time required to deliver such a module cannot be predicted precisely. It is a random variable that depends on the level of complexity, which often can only be determined after actual programming begins. It also depends on the skills of the specific people involved, what other jobs those people are required to do simultaneously, and how much pressure they are under to finish on time. These are just a few of the parameters that impact actual duration.

Does an estimated three-month completion time mean that three months is the "expected value" – in other words it will take three months *on average?* Is it an optimistic view, meaning "if we are lucky it will be done in three months?" Or is it a pessimistic view, meaning "it should not take longer than three months?"

To understand the answer, consider the question again. What would you – the reader currently in the shoes of the CEO of that small software company – mean when you asked for an estimate: an *average, optimistic*, or *pessimistic* one? Which of these three would give you the information needed for the decisions you must make: a) Will you proceed to develop that feature? b) When will you promise a first working module to your clients? c) How much of your resources must you dedicate to developing that module?

In most cases, you would probably ask for a pessimistic estimation. A reasonable worst-case scenario tells you the potential full impact of that decision on your operations. This is even more the case when you must promise your client a delivery date. You would like to be able to meet your commitment to the client, so you would probably prefer your chief programmer to commit to a date. Now, if your chief programmer understands that this estimate is a commitment, he or she certainly would not want the project to take more than three months. The chief programmer's estimate will probably be based on the

pessimistic view where some "safety" time has been added to the average (expected) time.

If this is the case, then why do the chances of finishing the project in a mere two months become so remote? If the chief programmer's estimates typically add substantial safety time to average durations, we should frequently see many projects finish early. Yet this is almost never the case in the reality of the software world. More commonly, it might take six months or more to deliver the module. And if that is true, what confidence should we have in the chief programmer's "pessimistic" estimates in the future?

A Hidden Cause

This phenomenon can be explained by recognizing a vicious cycle. Any adjustment to the estimate (making it even more pessimistic than before) does not improve the chances of meeting the estimated/expected time. This is because the estimate itself, which includes a large degree of safety time embedded in it, serves to prolong the project. The effect at work here is known as Parkinson's Law: Work will expand to consume the time allotted for it.

Applied to projects, Parkinson's Law suggests that the duration of any project stretches – at least – to the full duration of the time planned for it. In other words, once the project is planned for three months, human behavior will not let it finish early. By not allowing the project to finish earlier than estimated, the chances of finishing late actually increase. Here is how this happens.

There are three major human motivations behind Parkinson's Law. The first is called the "student syndrome." The term originates from the academic world where students typically wait until just before an assignment is due before beginning it. As long as we think we have enough time to finish our part in a project, we do not feel any real pressure to get started. Consequently, almost all the efforts are concentrated near the end of the time allowed for the activity, and only very little is done at the beginning. In essence, the built-in safety time is squandered. By the time work is actually begun, a timely task completion depends on mere luck with no safety time to accommodate the unexpected.

The second motivation is embedded in organizational politics. If you estimate the necessary "pessimistic" time to be three months, you certainly are not going to admit it actually took only two months. Even if the estimate was imposed on you, it is not likely that you would admit to finishing early, as you could expect even tighter times to be imposed on you in the future. Here again, safety time is frittered away. So there are two human behavioral issues at work. One says, "I don't have to start right now – there's plenty of time." The other says, "Even if I finish early, I'm not going to tell anyone about it."

Lastly the third motivation is even stronger than the preceding two: The code can *always* be improved. We can add features that were not requested by the client, but are nice to have all the same. We can improve the screen layout or use 3D graphics – even though 2D would do fine. (It looks *so* much more sophisticated in three dimensions.) We can also write the code in a more clever or elegant way.

Giving programmers this latitude to define what any function or module should actually do serves to consume the safety time prior to the delivery date. We do not think we are dragging the task. Actually, we feel completely the opposite: There are *so many* more things to do in the specified time! After all, no one really expects early completion. But any additional code – bells and whistles – tends to complicate the truly necessary part of the code and may cause huge quality problems. Even one more day devoted to additions that are not truly required can result in weeks of searching for and fixing bugs thus causing even more bugs. The critical balance between the perceived time to write the code and the amount of time needed for effective debugging is shattered. Put another way, "The road to hell is paved with good intentions."

Even if the safety time added to the average estimate were relatively small, the impact of Parkinson's Law is such that most of the time the best we could ever hope for is to meet the original schedule. Jobs would almost never finish early. Unfortunately, it is usually never even this good. Over many tasks, the nature of the distribution of time for developing a module makes the full impact of Parkinson's Law quite deadly.

What does this distribution look like? It is not a nice, symmetrical normal distribution. Think about your own experiences. Sometimes working on a one-month job – a true average estimate, not the estimate given to the boss for public consumption – may take three months. This may not happen frequently, but sometimes it does occur. On the other hand the one-month job will never be finished in three days, or probably in a week either. But it might take two weeks instead of a month. What we find is that such a distribution of time is not symmetrical. This occurs more so in software development because of the tendency to add the niceties that complicate the code. The resulting distribution of task completion times is more likely to be skewed, as indicated in Figure 1.

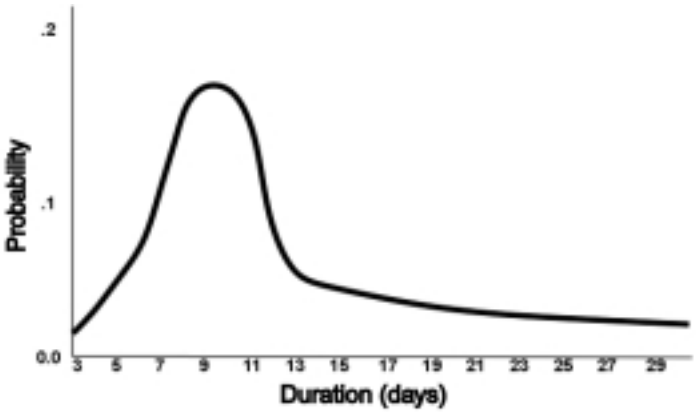


Figure 1. *Predicted Project Task Duration*

This figure shows a typical probability distribution over many projects of the number of days it takes to do similar tasks – or even similar whole projects. The area under the curve equals a probability of 100 percent. The median (a 50 percent chance of being less or equal to that number) is 10 days. The possible outcome may be any number from three on, but above 30 the probability is too small to be of practical value. Yet in some infrequent cases – but not absolutely rare – such a job

might take 30 days, though most likely it would require 9-10 days.

Notice something very critical in this chart. If you would like to add enough safety time to your median estimate, you need to *double* your estimate of the expected time to be adequately protected.[1] And even doing so, you will not be perfectly protected. In such a case, estimating 20 days seems about right, though it might actually take longer than that. But it might be more reliable all the same, were Parkinson's Law not involved.

So, updating expectations does not prevent new disappointments. The expectations set the deadline, and the people on the job see that deadline and try to meet it – not beat or exceed it, just *meet* it. However, enough unexpected incidents and problems occur to delay completion even more. Toward the end of the project, pressure often builds to sacrifice some of the original features that were planned.

Dependencies Impact Outcomes

The example cited above dealt with a simple, straightforward, single module. In most software projects, such a module is just one part of the overall project. Between the various modules in the project, and between the tasks within each module, there are certain dependencies. Dependencies make a significant difference in the outcome, because their sensitivity to delays has a greater impact than not merely finishing early.

Figure 2 shows a typical project activity network in which three different modules, each done by a different programmer (or a team) are integrated into one larger module, resulting in a requirement to test and modify the original code. Integration cannot start until all the three inputs are done – in other words, after the last one finishes.

If all modules behave according to a skewed distribution

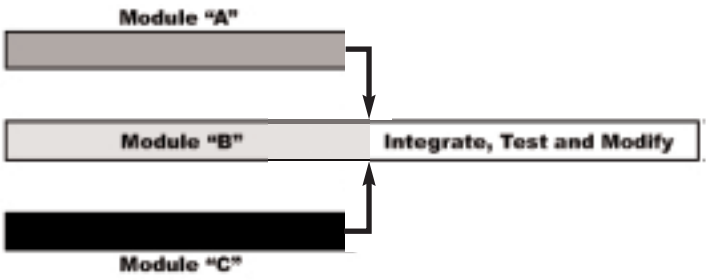


Figure 2. *Software Project Activity Network (Three Modules)*

(Figure 1), there is a fair chance that the overall effort will finish much later than the linear sum of the longest path (critical path). In other words the average of the sum is more than the sum of the averages. Being late in just one of the inputs is fully transferred to the last task, regardless of how early the other two are completed. (Variation accumulates at the end of the process.)

Combine the addition of safety time to every task, wasting that safety, adding unnecessary bells and whistles, compound the effect of all these with dependencies, and the result is a late project. Updating the original expected delivery date does not help. In fact, it creates the aforementioned vicious cycle. Expectations fail to be met on a regular basis - a huge problem.

Expressing the Problem as a Conflict

The Theory of Constraints suggests that a fully recognized problem that has not been overcome is the result of an unresolved conflict. A problem seems unresolvable because what appears to be the solution on one hand seems to intensify the problem from a different aspect.

Let us consider the vicious cycle described above. On one hand, safety time must be added to the various tasks to fairly estimate how long the whole project might take. On the other hand, we *should not* add safety to tasks because that safety is eventually wasted and we inevitably find ourselves facing worse delays.

By expressing the conflict clearly, we can surface some underlying assumptions that might be challenged. Figure 3 illustrates our conflict. The objective (A) is to manage our software company well. In order to manage our company well, we must plan software projects realistically (B). We must also complete our software projects as early as possible (C). In order to plan software projects realistically, we must plan for adequate safety time in each task (D). In order to complete our software projects as early as possible, we must *not* incorporate safety time for each task (D'). On one hand we must add safety time to each task; on the other hand we must not. But we cannot do both.

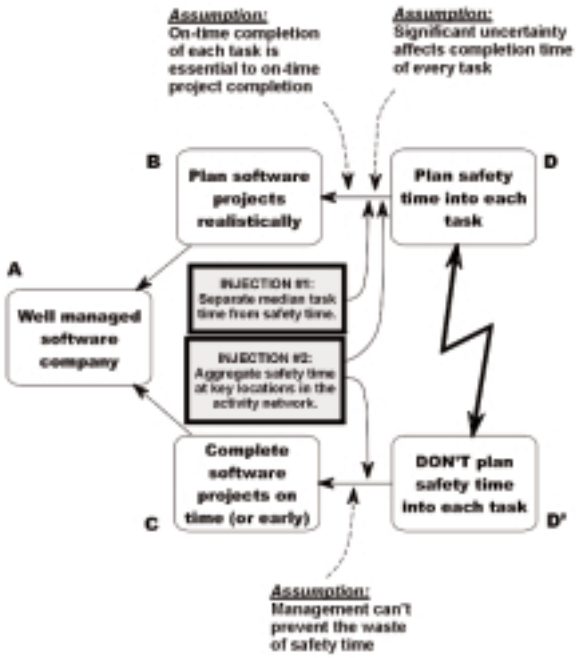


Figure 3. The Project Planning Conflict

The typical management solution is to compromise between the two seemingly contradictory actions. In our software case, senior management typically does this by arguing with the estimator until agreement is achieved, reluctantly. For instance, you could argue with your chief programmer that instead of three months you want it to be ready in only two months, with only four total man-months invested. The chief programmer might resist, and you might eventually agree to 10 weeks.

But a compromise approach still leaves the problem in place: You still cannot realistically estimate when the project will be completed while time is wasted, delaying project completion

without giving any real added value.

The importance of structuring the conflict as it appears in Figure 3 is that it allows us to articulate underlying (and probably unstated) assumptions. If any of these assumptions are invalid – or if they can be *made* invalid – then one or both of the conflicting prerequisites (D or D') can be replaced with some other alternative that satisfies the requirements of B and C, yet does not pose a conflict. Three key assumptions are shown in Figure 3.

One of these assumptions says that the completion time of each task is affected by much uncertainty. Can we challenge that assumption? Up to a point, we can. There are certainly ways to reduce uncertainty. For instance, knowing who is going to do the job and being familiar with his or her capabilities can help reduce the range of expected time for delivering the module. But can we reduce the amount of uncertainty low enough to dissolve the conflict? In many cases this is not possible. So this is probably a valid assumption.

What about the other two assumptions? Let us start with the one at the bottom (between C and D'): "Management can't prevent the waste of safety time." If we could create a situation where we can continually monitor the use of the safety time, we might be able to discourage people from wasting it. Can we do this?

First, we must clearly differentiate between the average estimation and any additional safety time. This would allow us to monitor the use of the safety time, which would reduce people's inclination to waste it. It would certainly allow management to ask whether the consumption of the safety time is caused by including features that were not required by the clients. However, this does not fully invalidate the assumption. It isn't practical to track *every* task and inquire why it used its safety time. Note that the median 50 percent estimation for every task means that half the time the task would use the safety time, even without any conscious intent to waste it. Ineffectiveness in tracing excessive use of safety time might still encourage people to behave according to Parkinson's Law.

The final assumption is even more important to invalidate. Even in the software world, most projects involve several different resources and task dependencies. The ultimate objective is that the overall project should not be delayed. On-time completion of individual tasks may be irrelevant. It might take longer to finish a particular task, but if we can still complete the whole project on time, why should the delay of an individual task matter? Moreover, there is no point including safety time for every task if that time is nearly always squandered anyway. Accumulating safety time for the project as a whole (at key points and at the end) is statistically superior; knowing that individual task safety time has been eliminated neutralizes people's tendency to waste time in return for no real value.

The main idea is to add safety time, but not to every individual task. We should use discrete safety time at critical locations within the project. Each task should be planned to consume the median estimate of the time required. When needed, the common safety time would be available to use. Making such safety time common to many tasks would make it much more difficult to waste.

The Nature of the Solution

To break the vicious cycle depicted in this conflict (Figure 3), we need to get rid of at least one invalid assumption. Invalidating multiple assumptions would be better. The last two assumptions cited above, very common today in managing software development, can be invalidated with the application of a new approach based on several new policies:

•Replace critical path concept with the notion of a "critical chain" as the constraint to earlier completion of any project.

The critical chain[2] is defined as the longest chain of operations linked by either finish-start connection or by resource availability. Resource availability is typically compromised by contention: We cannot complete some tasks concurrently, because they must be done by the same resource unit. The critical chain concept replaces the critical path concept, which usually ignores resource dependencies. The total length of the critical chain dictates the length of the project as a whole.

•Strip safety time from individual tasks. What should remain is the median. By doing so, we make it clear to every programmer that we expect them to concentrate only on the necessary features, avoid the student syndrome, and strive to complete their work early.

• Establish a project buffer This concept requires aggregating individual task safety time, while scheduling individual tasks based on average estimated completion time without safety time. The buffer is a specified length of time – usually considerably less than the sum of the individual task safety times – that is placed at the very end of the project, immediately after the last functional task. The actual completion time of the project is assumed to be at the end of the project buffer. This buffer compensates for the safety time eliminated from the individual tasks along the critical chain by re-inserting part of that safety time as a buffer for the whole chain. If we avoid the effects of Parkinson's Law and take advantage of the fact that some tasks would normally finish early, we can potentially realize a substantial time saving. Some of this aggregated safety time can be used to buffer other tasks; some of it can actually be eliminated, resulting in early delivery of the entire project. For planning purposes, the buffer looks like a task, with predicted start and finish time, but it has no resource assigned to it.

•Establish feeding buffers. The project buffer is the primary protection mechanism. It directly protects the project's due date from any delay along the critical chain. Delays in tasks that are *not* on the critical chain can delay the project completion *only* if they delay a critical chain task. To preclude this from happening, time buffers are inserted wherever a task not on the critical chain merges with a critical chain task. These are called "feeding buffers." They represent safety time to neutralize delays that otherwise might pass all the way through to the completion of the last task in the project.

•Manage the buffers. Once the critical chain schedule is constructed, with appropriate buffers inserted at key points, the project manager must then monitor the actual state of the buffers as the project is executed. At any point in time we can look at the current buffer consumption, meaning the accumulated delay along the chain that ends with the project buffer.

"Buffer penetration" constitutes the total time consumed, for all tasks to date, over and above expected average task time.

For example, let us assume that the project is a single sequence of 20 discrete tasks, each of approximately the same duration. Let us also assume that we have planned a 40-day project buffer. By the completion of the fifth task (roughly one-quarter of the way through the project), we have had one task finish on time; one finish two days early; and three tasks finish three, five and seven days late, respectively. Because one task finished two days early, the 40-day buffer is penetrated by 13 days (3+5+7-2). One quarter of the way through the project, we've used 32.5 percent of the project buffer. This is a disturbing indication because the protection (buffer) is being consumed at a faster pace than the progress of the project. At the moment, this might not be enough to warrant any actions to expedite, but it is enough of an indication to keep us watching the next few activities very closely. If any of those activities turn out to overrun their expected times (i.e., if more of the project buffer is consumed), we would probably consider options to accelerate progress.

The comforting aspect of this for the project manager is that the amount of the buffer consumed gives a clue to how much is left to protect the remaining tasks. It provides warning of a possible late delivery very early in the schedule, while there is still time to make small corrections versus crashing the project. The project buffer status, relative to where we are on the critical chain, gives the project manager invaluable information about the status of the project as a whole.

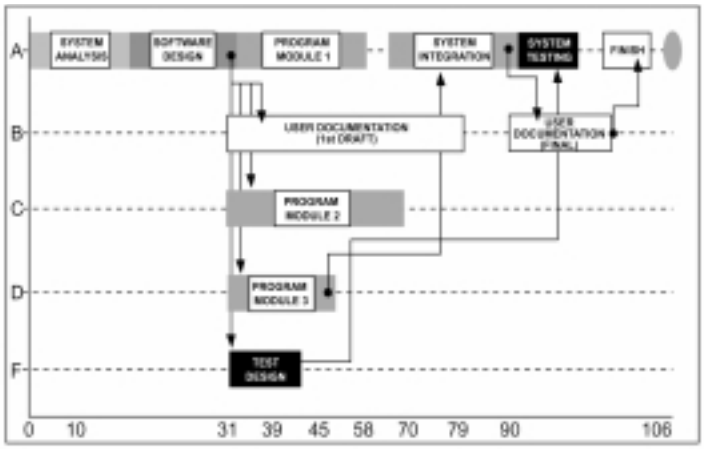
Other actions are required to apply this approach in multi-project environments, but that is beyond the scope of this article.

An Example

Figure 4 shows a basic structure for a simplified software project. The tasks outlined here are: system analysis, software design, program module 1, program module 2, program module 3, test design, user documentation (first draft), system integration, system testing, final user documentation, and finishing (packaging and shipping).

Task durations include individual safety time for every task providing a 90-percent probability of finishing on time or early. The estimated time to complete the whole project is 106 days.

Figure 4. *Three Module Software Development Project*



There is one problem in this plan. Only two programmers are assigned to the project. We have three modules to program that could theoretically be done in parallel, but because of the lack of one programmer, two of the modules will need to be completed consecutively. Also, system integration and system testing cannot begin until all the modules are completed and both programmers are available to support these activities.

The first step to achieving a realistic schedule is to resolve the resource contention. We can also identify the critical chain – the chain of tasks that dictate the length of the project. Figure 5 shows the original schedule replanned with the critical chain in mind. The figure highlights those tasks (heavy arrow). According to the adjusted plan, the project should be completed on day 118.

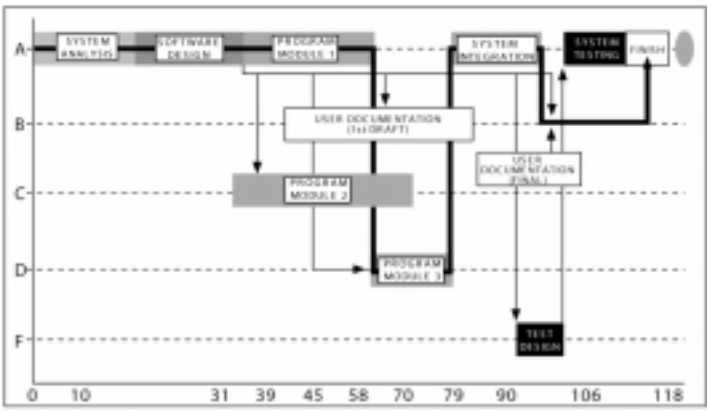


Figure 5. *Critical Chain Activity Network*

We have simulated this example to evaluate the impact of Parkinson's Law. For our computerized simulation, we have assumed that Parkinson's Law applies only 75 percent of the time. In other words when a task seems to have enough time, in 75 percent of the cases the performer will fill the excess time so that delivery to the next step in the process would happen exactly at the scheduled time. The other 25 percent of the time, when the task should take less than the planned duration, it actually does take less time. In Figure 5 we have simulated the case characterized 1,000 times. The project finished on time or early in only 34 percent of the runs (340 times out of 1,000). Remember that every task estimate was inflated so that in 90 percent of the cases it would finish on time or earlier, yet the project as a whole finished on schedule just one-third of the time. So adding all that individual task safety time did not do much good.

Following the concepts described earlier, the next step is to strip the safety time from each task and establish a project buffer and feeding buffers. That arrangement looks like Figure 6 (See page 24). The blocks with the beveled ends indicate the buffers.

Notice that all the tasks are now planned to take only about half of the time they were assigned previously. But we are not fooling ourselves – not all of them will meet those schedules. Actually, we expect that about half of the tasks will actually take longer than planned, so we place a buffer of 30 days at the very end of the project. While the last task appears to finish on day 59, we actually expect it to finish at/or before day 89. Notice also the feeding buffers at the right side of rows A, B, and C.

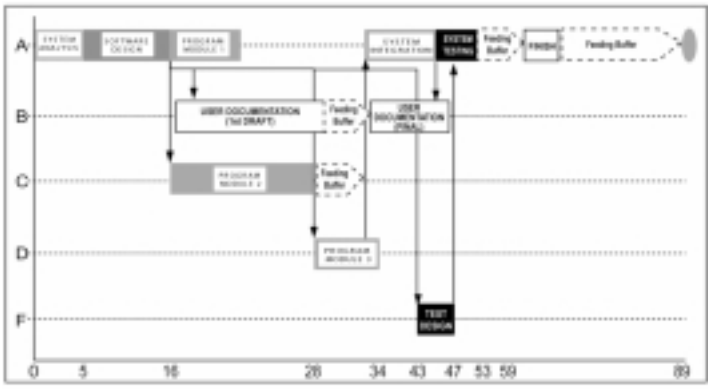


Figure 6. Project Buffers and Feeding Buffers

They protect critical chain tasks from delays induced by non-critical chain tasks.

Finishing a project in 89 days rather than 118 days seems very favorable. But we know that our original project plan was not too realistic. Do we really stand a chance of completing in only 89 days?

Even though we assume that half of the tasks would need more than the expected time to complete, the shorter planned time for all tasks helps to drastically reduce the impact of Parkinson's Law. After running the simulation in this configuration (with project and feeding buffers in place) 1,000 times, the project finished on time in 92 percent of the runs; 8 percent of the time it took more than 89 days. Buffers cannot offer 100-percent protection. But which would you rather have: 34 percent probability or 92 percent?

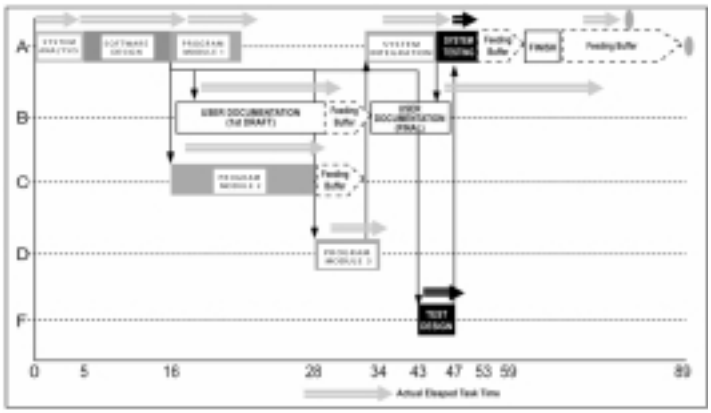


Figure 7. *Critical Chain Activity Network (Simulation Run #77)*

Figure 7 shows one discrete run out of the 1,000. The critical chain tasks are indicated by the heavy black arrow in Figure 5. The actual run is reflected in the hollow arrows above the "plan" bars. The dark stripes in each buffer symbol show how much of the buffer was consumed. Only one of the three feeding buffers was fully consumed. The consumption of the project buffer resulted from delays in the critical chain tasks, especially the next-to-last one. All in all, the project finished in 77 days in this run, well within our expectations.

Conclusion

Due to the vicious cycle inherent in software projects, updating our expectations only worsens a bad situation.Two central ideas to solve the vicious cycle emerge from verbalizing the problem as a conflict between two required actions: Add safety time to every task on one hand, but refrain from adding safety time to every task on the other.

The solution is developed from challenging two basic assumptions. First we cannot prevent wasting safety time; second we must strive to ensure that every individual task is finished at the planned time.

Accumulating the buffers where safety is truly needed enables us to set and achieve realistic expectations. Monitoring these buffers is crucial to successful project execution. Though not all tasks can be expected to go exactly as planned, effective buffer management assures a much higher probability that the overall project will be delivered on time.◆

Notes

1. There are three common values that represent the "center" of a statistical distribution: *mean, median* and *most likely.* For project task estimation, we consider the median an easier measure to estimate intuitively. We also believe it to be more relevant for decision making. The mean value is affected by extreme values that should not, in our opinion, be part of routine decision making. Note that while the three values are different for skewed distributions, for the human intuitively doing the estimating, the differences can be ignored for practical purposes.
2. For more information about Critical Chain Project Management see *Project Scheduling According to Dr. Goldratt* by Timothy K. Perkins in January 2001 CROSSTALK. Also, visit the Goldratt Institute Web site at www.goldratt.com, the Goal Systems International Web site at www.goalsys;com, Goldratt's site at www.eligoldratt.com, and the Web site maintained "as a hobby" by David Shucavage at www.rogo.com/cac/index.htm

About the Author

Eli Schragenheim has been associated with the Theory of Constraints for 16 years. He is active both as consultant and educator, and as software developer of simulations used for management education. He works closely with Dr. Eli Goldratt, the father of the Theory of Constraints, especially on issues of software development. Schragenheim has a master's degree from Tel Aviv University, Israel, and a bachelor's degree in mathematics from Hebrew University, Israel. He is the author of "Management Dilemmas: The Theory of Constraints Approach to Problem Identification and Solutions," co-author with H. William Dettmer of "Manufacturing at Warp Speed: Optimizing Supply Chain Financial Performance," and co-author with Dr. Eli Golidratt and Carol Ptak of "Necessary But Not Sufficient." Schragenheim can be reached at:

E-mail: elyakim@netvision.net.il

H. William Dettmer has more than 20 years of leadership and management experience in operations and logistics environments. He was an adjunct faculty member at the University of Southern California and has taught extensively at the graduate level on the Theory of Constraints, total quality management, project management, systems analysis, management control systems, and organizational behavior and development. Dettmer has a bachelor's degree from Rutgers University and a master's degree from the University of Southern California. He is the author of "Goldratt's Theory of Constraints: A Systems Approach to Continuous Improvement," and "Breaking the Constraints to World-Class Performance," Dettmer can be reached at:

Goal Systems International
E-mail: gsi@goalsys.com
Voice: (360) 565-8300

"If computers get too powerful, we can organize them into a committee -- that will do them in."

BRADLEY'S BROMIDE

"There are two ways of constructing a software design: One way is to make it so simple that there are obviously no deficiencies, and the other way is to make it so complicated that there are no obvious deficiencies. The first method is far more difficult."

C.A.R. HOARE

Coming Events

April 29-May 4

Software Technology Conference (STC 2001)
www.stc-online.org

May 1-3
2001 IEEE Radar Conference
www.atlaessgrss.org/radarcon2001

May 12-19
23rd International Conference on Software Engineering, and *International Workshop on Program Comprehension*
www.csr.uvic.ca/icse2001

May 8-14
Software Testing Analysis and Review
www.sqe.com/stareast

May 22
7th Annual Montgomery Area IT Partnership Day
web1.ssg.gunter.af.mil/partnership

June 5-7
AFCEA's 55th International Convention and Exposition
www.technet2001.org

June 11-13
E-Business Quality Applications Conference
qaiusa.com/conferences/june2001/index.html

June 18-22
ACM/IEEE Design Automation Conference
www.dac.com

June 25-27
2001 American Control Conference
www.ece.cmu.edu/~acc2001

Letter to the Editor

Dear Editor:

I just received the December 2000 issue of CrossTalk. How could you leave out the Project Management Institute's (PMI®) Web site www.pmi.org ? You have shortchanged your readers on this one. PMI is one of the leading project management advocates in the United States. Since its founding in 1969, PMI has grown to be the organization of choice for project management professionalism. With nearly 70,000 members worldwide, PMI is the leading nonprofit professional association for project management. PMI establishes project management standards, provides seminars and educational programs, and professional certification that more and more organizations desire for their project leaders.

David Cottengim PMP,CSTE,CGFM,MBA
Business Manager, Project Integration
Defense Joint Accounting System
Defense Finance and Accounting Service

STC 2001

Register today for the Thirteenth Annual

Software Technology Conference

2001 Software Odyssey: Controlling Cost, Schedule, and Quality

29 April – 4 May 2001

Join us as we explore opportunities for perfecting today's technology at STC 2001. Enjoy an entire week of exciting presentations designed for professionals and managers in the DoD, contractors supporting the DoD mission, and individuals in industry and academia. Participants will learn about proven technologies, policies, and practices to aid in managing daily challenges common to information technology professionals throughout the DoD. ***It's not too late! Register today!***

Co-Sponsors' Individual Service Sessions

Monday, 30 April 2001 • 5:00 pm – 6:00 pm

The departments of the Army, Navy, and Air Force are hosting concurrent individual service sessions for all conference attendees in uniform and government employees. The sessions are officially open to all levels/ranks of attendees and are a great opportunity for the service members to hear from their particular service leader.

Co-Sponsors' Panel Discussion

Tuesday, 1 May 2001 • 8:00 am – 9:40 am

Lt Gen Harry D. Raduege, Jr.
Director,
Defense Information
Systems Agency

LTG Peter M. Cuviello
Director of Information
Systems for Command,
Control, Communications,
and Computers, U.S. Army

RADM Kenneth D. Slagle
Vice Commander,
Space and Naval Warfare
Systems Command,
U.S. Navy

Dr. Donald C. Daniel
Deputy Assistant Secretary,
Science, Technology,
and Engineering,
U.S. Air Force

The STC 2001 co-sponsors will hold a panel discussion moderated by Ms. Dawn C. Meyerriecks, Chief Technology Officer for the Defense Information Systems Agency (DISA) and Chief Engineering Executive for Information Processing within the Joint Interoperability and Engineering Organization (JIEO). Don't miss this opportunity to hear the latest information from the co-sponsors on how they are handling software in their organization.

Late Breaking News

Dr. Delores Etter, Deputy Under Secretary of Defense (Science and Technology), will be speaking with **Dr. Vitalij Garber** at the Opening General Session on Monday afternoon. Dr. Etter's presentation will focus on the *State of Software-Intensive Systems in the Department of Defense*.

In addition, an eleventh presentation track has been added to the STC 2001 conference schedule. This track includes presentations by DISA Tuesday afternoon and a DII COE tutorial, STSC workshops, and other exciting presentations throughout the day on Wednesday. The schedule for this track will be posted on the STC Web site and updated as additional presentation times are filled.

We encourage you to visit our Web site for up-to-date conference information as well as a dynamic conference schedule that allows you to *build your conference week online.*

STC 2001

www.stc-online.org

General Information
steinfo@ext.usu.edu
435-797-0423

Technical Content Inquiries
stc@hill.af.mil
801-777-7411

Trade Show Inquiries
steexhibits@ext.usu.edu
435-797-0047

Media Relations
stcmedia@ext.usu.edu
435-797-1349

Closing General Session

Thursday, 3 May 2001 • 4:30 pm – 5:30 pm

Maj Gen John L. Barry
Director of Strategic Planning,
Deputy Chief of Staff for Plans and Programs,
Headquarters USAF

Join us at the Closing General Session where Major General Barry will discuss the interrelationship between the major areas of VISTA (Visionary Ideas Shaping Tomorrow's Airmen) and how they fit into an overall architecture that shapes the Air Force's strategic planning process.

Networking Events and Optional Activities

Admission to each of the following networking opportunities (with the exception of the two noted as "optional tours") is included in the conference registration fee. A companion ticket package is available for $45 and includes the Opening Welcome Reception, "Drag 'n Drop" Social, and "1964"...The Tribute - An Evening of Entertainment.

Opening Welcome Reception
Monday, 30 April 2001 6:30 pm - 8:00 pm

Lunch Options
Tuesday & Wednesday 11:30 am - 1:00 pm

Salt Lake City Evening Overview *(Optional Tour)*
Tuesday, 1 May 2001 5:00 pm - 7:00 pm

"Drag 'n Drop" Social
Wednesday, 2 May 2001 4:30 pm - 6:00 pm

"1964"...The Tribute - An Evening of Entertainment
Wednesday, 2 May 2001 6:30 pm - 8:00 pm

"Light Byte" Luncheon
Thursday, 3 May 2001 11:30 am - 1:00 pm

Dinner Cruise *(Optional Tour)*
Thursday, 3 May 2001 6:15 pm - 10:00 pm

Conference Registration Fees

Regular registration fee:

Active Duty Military/Government*	$620
Business/Industry/Other	$770

* *Military rank (active duty) or government GS rating or equivalent is required to qualify for these rates.*

Cyber Warfare: A New Doctrine and Taxonomy

Lt. Col. Lionel D. Alford Jr.
U.S. Air Force

Software is a key component in nearly every critical system used by the Department of Defense. Attacking the software in a system – cyber warfare – is a revolutionary method of pursuing war. This paper discusses the limitations of current doctrine and suggests new cyber warfare taxonomy.

Karl von Clausewitz defined war as "… an act of violence intended to compel our opponent to fulfill our will … In order to attain this object fully, the enemy must be disarmed, and disarmament becomes therefore the immediate object of hostilities … [1]." At the end of the second millennium, this definition no longer describes the full spectrum of modern warfare. In the near future, – with software alone – we will have the potential to make war without the use of violence and fulfill the second half of von Clauswitz's definition. Today's software-intensive systems make this possible.

Cyber describes systems that use mechanical or electronic systems to replace human control. In this paper the term includes systems that incorporate software as a key control element. Cyber warfare can be executed without violence, and therefore the dependence on software intensive systems – cyber systems – can make nations vulnerable to warfare without violence.

Full-Circle Protection Required

Cyber warfare is the conduct of military operations according to information-related principles [2]. However, this does not define the full degree of capabilities now possible in cyber warfare. Limiting the scope of cyber warfare to information-related principles does not describe what happens when an enemy disrupts the electrical power grid of a nation by hacking the controlling software (Figure 1). Information is not only at risk – so is the fundamental control of civilization. As technology progresses, this fundamental control will continue to devolve into networks and software-controlled electronics [3].

This transition has already occurred in aviation. Previously, 100 percent of an aircraft's performance and capabilities were defined by hardware – the physical makeup of the aircraft. Today in the most advanced aircraft, 75 percent or more of the aircraft's performance and capability is absolutely dependent on the software [4]. Without software, aircraft would not be controllable or reach the desired performance capabilities[1]. In some cases, through software, aircraft performance is gaining limited independence from physical configuration[2]. Software dependence and hardware independence are growing: modern aircraft fly-by-wire, their engines are controlled-by-wire, their weapons are fired- and dropped-by-wire. Systems that in the past were entirely hardware with mechanical control are being replaced by software with software control. Software defines the strength of modern systems, and through networking provides a basis for the integration of many disparate items. These networked software systems are under attack today, and the attacks are increasing (Figure 2).

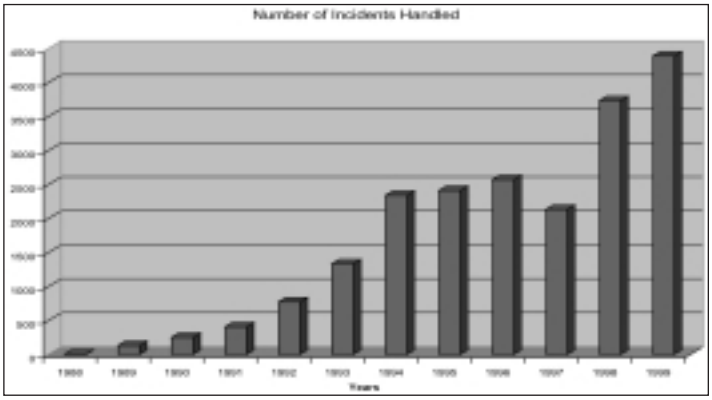


Figure 2. Number of CERT Incidents Handled (3)

Unfortunately current Department of Defense (DoD) doctrines and instructions do not adequately cover the scope of cyber warfare [5]. Several handle information warfare as a discrete part of a military system. These include Joint Publication 3-13 Joint Doctrine for Information Operations (JP3-13), Joint Publication 3-13.1 Joint Doctrine for Command and Control Warfare (JP3-13.1), and instructions such as DoD 5000.2-R Mandatory Procedures for Major Defense Acquisition Programs, and Major Automated Information System Acquisition Programs. Current doctrine does not address software as the major element of a military fighting system.

Yet as the above discussion shows, many software and soft-

Figure 1. Infiltration of a Utility

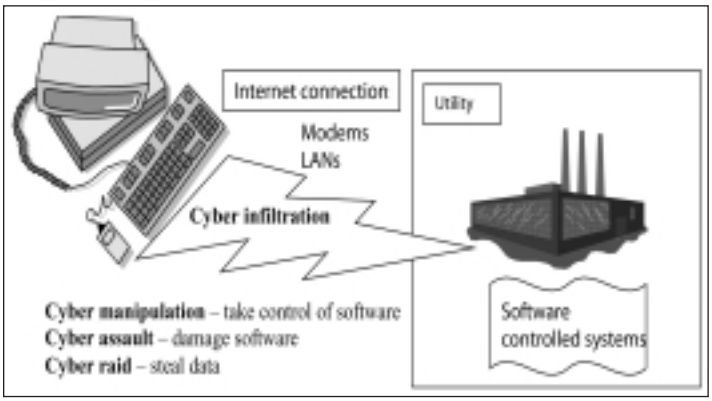
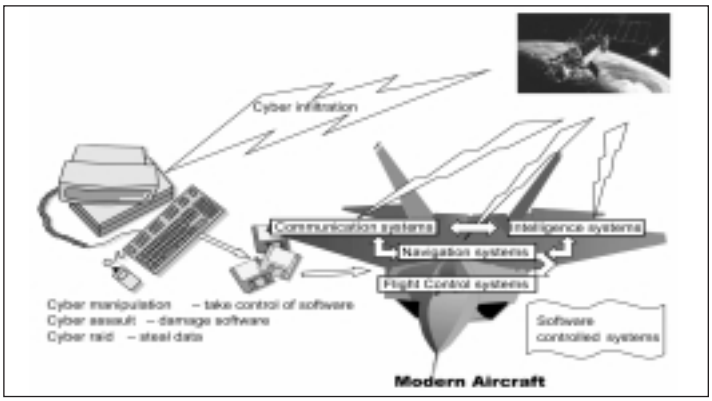


Figure 3. Infiltration of an Aircraft



ware-controlled systems cannot be separated from the system being developed. The F-22 weapon system is an example of a software-controlled aircraft system that contains and communicates with integrated information systems (Figure 3, page 27). The F-22 is not a closed system; external information systems update and integrate F-22 combat operations during flight. Through these external connections, both the information systems and the basic software systems of the F-22 can be attacked. Current information warfare doctrine in the Joint Pubs is mainly concerned with security of external C4I systems integrated on the F-22, but software-intensive systems make internal systems of the F-22 vulnerable to cyber warfare attack. Our doctrine must account for these vulnerabilities and provide methods of offense and defense. Definitions for building future weapon systems cyber forces doctrine and recommended methods to incorporate them follow.

Cyber Warfare Definitions

Joint Pub 3-13, Joint Pub 3-13.1, and DoD 5000.2-R focus on information systems but not software controlled systems; these documents' definitions are not sufficient to describe the full range of cyber warfare. The Computer Emergency Response Team® coordination center does provide a strong set of common terms to define cyber system security for the DoD [6], but these terms do not discuss military doctrine or national security.

Furthermore, these terms focus on current methods of defense against infiltration and attack; they do not focus on future cyber force capabilities. We need a new taxonomy that includes the full range of cyber operations and aids the development of a national cyber warfare doctrine. (See accompanying sidebar).

Military Cyber Warfare Targets

Any military system controlled by software is vulnerable to cyber attack. The first step in any attack is cyber infiltration; all systems that incorporate software are vulnerable to cyber infiltration[4]. Actions following cyber infiltration can affect organizations via the transfer, destruction, and altering of records – cyber raid. Software within systems can be manipulated or systems controlled by that software can be damaged or controlled – cyber manipulation. The software itself can be copied, damaged, or rewritten – cyber assault.

Military Command, Control, Communications, Computers, and Intelligence (C4I) systems are particularly vulnerable, and are the primary focus of DoD cyber-related doctrine. JP 3-13 and JP 3-13.1 both provide doctrine for information related warfare. C4I systems are a very complex mix – from radios to radars, mainframes to PCs. Military C4I uses interfaces through the Internet, base and organizational Local Area Networks (LAN), modems, civilian and military communication systems, navigation systems, and radios in all frequency ranges. Military C4I systems are extremely vulnerable because they interconnect.

Cyber infiltration can enter at many points and potentially affect a myriad of systems. These systems and their interactions are so complex that any modern military organization is unlikely to trace the full potential of any single cyber infiltration. The possibility exists for cyber attacks of every type, and the results can be catastrophic. For instance, nuclear weapon control systems are incorporated into military C4I. As demonstrated by recent incursions in DoD networks, databases, and Web sites [7], almost any dedicated foe can engage in cyber attacks against military computer systems [3]. Since military computers are the cores of national C4I, successful intentional and unintentional cyber warfare attack against such targets pose a national security peril.

Weapon systems are cyber attack targets, but current DoD doctrine adequately covers cyber attacks on military hardware systems such as aircraft, vehicles, etc. that require software to operate [8, 9, 10]. As noted previously, the F-22 is a cyber-controlled aircraft (Figure 3). Infiltrating and degrading the aircraft's systems directly or via its C4I connections can be as devastating as shooting it out of the sky. Cyber infiltration of the C4I system providing data to modern aircraft allows an avenue for cyber raid, manipulation, and assault.

Because many systems like the Global Positioning System

A New Taxonomy of Cyber Terms

Cyber warfare (CyW) - Any act intended to compel an opponent to fulfill our national will, executed against the software controlling processes within an opponent's system. CyW includes the following modes of cyber attack: cyber infiltration, cyber manipulation, cyber assault, and cyber raid.
Cyber infiltration (CyI) - Penetration of the defenses of a software-controlled system such that the system can be manipulated, assaulted, or raided.
Cyber manipulation (CyM) - Following infiltration, the control of a system via its software that leaves the system intact, then uses the capabilities of the system to do damage. For example, using an electric utility's software to turn off power.
Cyber assault (CyA) - Following infiltration, the destruction of software and data in the system, or attack on a system that damages the system capabilities. Includes viruses, overload of systems through e-mail (e-mail overflow), etc.
Cyber raid (CyR) - Following infiltration, the manipulation or acquisition of data within the system that leaves the system intact and results in transfer, destruction, or alteration of data. For example, stealing e-mail or taking password lists from a mail server.
Cyber attack - See CyI, CyM, CyA, or CyR.
Cyber crime (CyC) - Cyber attacks without the intent to affect national security or to further operations against national security.
Intentional cyber warfare attack (IA) - Any attack through cyber-means to intentionally affect national security (cyber warfare) or to further operations against national security. Includes cyber attacks by unintentional actors prompted by intentional actors. (Also see Unintentional Cyber warfare attack (UA).)
Intentional cyber actors (I-actors) - Individuals intentionally prosecuting cyber warfare (cyber operators, cyber troops, cyber warriors, cyber forces).
Unintentional cyber warfare attack (UA) - Any attack through cyber-means without the intent to affect national security (cyber crime).◆

and future intelligence systems automatically update aircraft information and intelligence, they can allow undetected aircraft infiltration. Intelligence, navigation, and communication systems are integrated with each other and to a host of other aircraft systems – the flight control system (through the autopilot), propulsion system (through the autothrottles), radar system, master warning system, and environmental control system. Using the correct control inputs or reprogramming an infiltrator could produce any level of systems damage, from driving the aircraft off-course to overwriting the flight control software.

Identifying Cyber Warfare Vulnerabilities

The first rule in identifying cyber warfare (CyW) vulnerabilities is that any software-controlled system that can accept input can theoretically be infiltrated and attacked. This means all systems that accept input are vulnerable. Fundamentally, there are two ways to infiltrate cyber systems: physical and signal inputs.

•**Physical infiltration** is made through the system hardware. For example, the on/off switch, keyboard, mouse, cockpit controls, flight controls, and removable media provide physical inputs into a system. The first line of defense for a software-based system is to secure the physical inputs and outputs of the system. If these are not secure, the system is not secure. Any system can be compromised if a cyber attacker can enter the facility/aircraft/vehicle and directly infiltrate the system. The cyber infiltration can be maintained afterwards by installing repeaters and remote input devices on the hardware.

For example, electronic bugs on phone lines are a common method of surreptitious surveillance; modem and LAN lines are equally vulnerable. An easy method of physical infiltration is to use a spare LAN connection on a hub or router. Using common network parts, a connection can be made directly, or through a Radio Frequency (RF) transmitter (wireless connection) from the LAN to an infiltrator's computer. These infiltration methods are only discovered by careful system audits or visual inspection [11].

•**Signal infiltration** comes through existing indirect/direct connections to a system. These connections are typically LANs, Infrared (IR) devices, RF connections (radios), and modems (phone lines). Any system with an external connection can theoretically be infiltrated; only the number of direct and indirect connections into the system limits the number of potential entry points. For instance, a system with an Internet server is vulnerable to cyber infiltration from any computer connected to the Internet. An isolated network with a modem is vulnerable to any computer that can call into it. These input paths are used to infiltrate the system and then assault, manipulate, or raid it.

Physical infiltration may be protected by physical security: walls, fences, restricted areas, identification, guards, etc. Signal infiltration has similar defenses, but these are incorporated within the software/hardware itself (for instance, passwords, coded signals, firewalls, terminal identification, isolation, and system monitors).

The second rule of identifying CyW vulnerabilities is to expect *every* software-controlled system to be the objective of an attempted cyber infiltration. Even isolated systems can experience cyber assault through a computer virus brought in on a contaminated floppy disk. Because cyber attacks are largely unpredictable, all systems must have some degree of protection, and the level of protection must be commensurate with the likelihood and consequences of expected attack. Every vulnerable system needs proactive and effective virus-protection in place.

All infiltration's should be assumed to be cyber attacks, until proven otherwise. Unintentional actors (U-actors) will be influenced by intentional actors (I-actors). The anonymity of the Internet makes it possible for a cyber operative to pass information on password-cracking, system phone numbers, infiltration techniques, and programs to U-actors. Many U-actors are young, immature, and unsophisticated. However, I-actors, operating through U-actors on the Internet may make some attacks that appear unintentional. The recent cyber infiltration of information systems by California teens trained by the Israeli hacker "Analyzer" is an example of this mentoring relationship [12].

I-actors can easily influence the direction of attacks by providing system access numbers and system passwords. Trojan horse programs written and passed to U-actors achieve an entirely different result than the U-actor intended. The outcome, from the perspective of the I-actor, is the same as if the attack had been made directly. Because passwords and infiltration data are shared by U-actors across the net, the I-actor's mission package is likely farmed out to more than one U-actor, or data may be passed through multiple U-actors. This ensures many attacks on the same target and further muddies the trail back to the source. This also means organizations that detect attacks and neutralize them should be prepared to receive the same attack over and over again. In addition, organizations that detect attacks must share data on the attacks immediately with other organizations [13].

Measuring Cyber Defense Effectiveness

The effectiveness of cyber forces cannot be measured by a lack of detected cyber infiltration against targets. This is because undetected cyber infiltration is certainly taking place [14], and most cyber infiltration's and attacks go undetected [13]. The only reasonable measure of effectiveness is detecting cyber infiltration when it happens. This is why a multi-layered approach to cyber system defenses is necessary. If the policy of the United States regarding CyW is wholly one of defense, the absolutely perfect measure of defense effectiveness is that every cyber infiltration is identified and the U- or I-actor neutralized.

The success of cyber operations against and in support of the U.S. government must be classified. As mentioned previously, when a cyber attack occurs, with due regard for active cyber operations, the detecting agency should immediately inform all possible targets [13]. But, when an agent of the government is the victim of successful cyber infiltration or attack, that agency should not release the degree or effects of any cyber operation against it. Acknowledging the results would be similar to acknowledging the classification of publicly published materials. It would tell the enemy they are successful and provide information so the next attack might be even more effective. The best approach is for the agency to make no comment at all and provide immediate recovery and cleanup as part of its cyber operations. This keeps the I- and U-actors guessing and allows

the effective use of the offensive and defensive methods above. This is not to say the agency should not report the attack to proper authorities and provide suggested methods of protection.

In light of today's cyber warfare, the first proactive step is to develop a strong doctrine that includes all the dimensions of current and future cyber warfare threats. Taxonomy and cataloged security methods go a long way to build a framework for this doctrine. The challenge is to put the required effort and funding forward to ensure a strong level of security for all software-controlled systems.

Conclusion

Cyber operations have the potential to overcome any system controlled by software. The military systems we are developing today depend on software and software-controlled components to operate. Cyber warfare defenses must be incorporated into all of these military systems. The future of warfare makes it imperative that cyber warfare concerns become the interest of every software and hardware developer – not only of military systems but civilian systems as well.

Cyber warfare may be the greatest threat that nations have ever faced. Never before has it been possible for one person to potentially affect an entire nation's security. And, never before could one person cause such widespread harm as is possible in cyber warfare. Like radioactive fallout, the affects of cyber warfare can devastate economies and civilizations long after the shooting war is over. This genie can't be put back into the bottle; societies will not want to give up the manifold prosperity brought about by cyber systems. But, a nation must ensure that it maintains the upper hand in cyber warfare. If our nation cannot, then even with the most powerful military and defense economy in the world, we face an insurmountable threat to our future prosperity and security[5].◆

References

1. von Clausewitz, Karl, *On War*, Book I, translated by Michael Howard and Peter Paret, Princeton University Press, 1976.
2. Arquilla, John and David, Ronfeldt, *Emergent Modes of Conflict*, *Cyberwar is Coming*, The RAND Corporation, 1992.
3. Vatis, Michael A., Cybercrime, Transnational Crime, and Intellectual Property Theft, Statement for the record before the Congressional Joint Economic Committee, 1998, www.ilspi.com/vatis.htm
4. U.S. Air Force, Bold Stroke, Executive Software Course, 1992.
5. Stein, George J., Information Warfare, *Airpower Journal*, Vol. IX, No. 1 Spring 1995.
6. Carnegie Mellon, Software Engineering Institute, CERT® Coordination Center, Glossary of Terms, 1997, www.cert.org/research/JHThesis/appendix_html/Glossary.html
7. Lemos, Robert, DoD Confirms Hacker Boast, ZDNN, 1998, www.zdnet.com/zdnn/content/zdnn/0421/309056.html
8. Joint Publication 3-13, Joint Doctrine for Information Operations, 9 Oct. 1998.
9. Joint Publication 3-13.1, Joint Doctrine for Command and Control Warfare, 7 Feb. 1996.
10. DoD 5000.2-R, Mandatory Procedures for Major Defense Acquisition Programs and Major Automated Information System Acquisition Programs, 27 Feb 1998.
11. Marshall, Victor H., Intrusion Detection in Computers, Summary of the Trusted Information Systems (TIS) Report on Intrusion Detection Systems, 1991.
12. Cole, Richard, FBI Hunts Master Hacker, ABC News: High Technology, *The Associated Press*, 1998.
13. Howard, John D., *An Analysis Of Security Incidents On The Internet 1989 - 1995,* Carnegie Mellon University, 1997. http://www.cert.org/research/JHThesis/Start.html
14. Lee, Stella, Most Computer Hackers Go Unnoticed, *South China Morning Post*, 1998. www.infowar.com/HACKER/hack_030198s_b.html-ssi

Notes

1. The F-16 is unstable below Mach one, and uncontrollable without its software based flight control system. The Boeing 777 and the Airbus 330 have software flight control systems without any manual backup; the performance of these air craft is dependent on their digital flight control systems.
2. The F-22 in high angle of attack flight uses software controlled vectored thrust and flight controls to maneuver the aircraft.
3. As seen in allegations that a *Cincinnati Enquirer* reporter stole voice mail messages from Chiquita Brands International [7], CyR is becoming a common method to take information from cyber systems.
4. The *hacker* is a U-actor commonly characterized as affecting cyber infiltration without further damage to a computer sys tem.
5. The views expressed in this paper represent the personal views of the author and are not necessarily the views of the Department of Defense or of the Department of the Air Force.

About the Author

Lt. Col. Lionel D. Alford Jr is an aeronautical test policy manager for the Headquarters Air Force Materiel Command, Wright-Patterson Air Force Base, Dayton, Ohio. He is an Air Force experimental test pilot with more than 3,600 hours flying more than 40 different types of aircraft. He is a member of the Society of Experimental Test Pilots. Lt. Col. Alford has served in worldwide military operations as a member of three different operational combat squadrons. He is a graduate of the Air Ground Operations School, the Combat Aircrew Training School, the All Weather Aerial Delivery Training School, Defense Systems Management College, and the USAF Test Pilot School. He was an instructor for three Air Force aircraft and a senior Air Force evaluator. He has a master's degree in mechanical engineering from Boston University and a bachelor's degree in chemistry from Pacific Lutheran University. Lt. Col. Alford is a computer experimenter and programmer, and is currently working on certification as a Microsoft system engineer.

Lionel D. Alford Jr.
HQ AFMC/DOP
4375 Chidlaw Road, Room S143
Wright-Patterson AFB, OH 45433-5006
Phone: 937-257-8496
Lionel.alford@wpafb.af.mil

A Tale of Two Developers

It was the best of times, it was the worst of times, it was the age of wisdom, it was the age of foolishness, it was the epoch of belief, it was the epoch of incredulity, it was the season of light, it was the season of darkness, it was the spring of hope, it was the winter of despair, we had everything before us, we had nothing before us, we were all going direct to Heaven, we were all going direct the other way.

–Charles Dickens

Dickens, introduction to *A Tale of Two Cities* could easily have been describing a tale of two software developers where wisdom, foolishness, hope, and despair reside in the same industry, organization, and projects. There are two factions in the software development community. They differ in thought, style, and process and are often at odds, yet dependent upon one another.

In one corner are disciplined designers, in the other are free-form programmers. Don't let the names mislead you. The designers program, and the programmers design. They share a passion for developing software but their methods and attitudes are as different as a skier is to a snow boarder on the alpine slopes.

In my college days there was only one way to glissade down a slope of white powder and that was with two planks – one on each foot. It was fresh, challenging, exhilarating, and the only thing that changed from year to year was the color of your parka and lift ticket prices.

As I grew older a new wave swept across the peaks and onto the slopes: baggy clothes, multi-colored hair, and the two planks were fused into one fat board. It was called shredding, and it was everything skiing was not. Skiing had become a safe, tidy, aristocratic sport full of rules and regulations. Shredding was free, open, and rebellious. A skier's herky-jerky turns down steep slopes were the antithesis of the boarder's fluidity; the two worlds did not mix.

Likewise, in my college days there was only one way to develop software and that was creative free-form. No rules, you choose the tools, damn the fools approach. We were pioneers, and it was exhilarating. Little did we know that the hardware limitations we complained about actually saved us from mass chaos.

Now a new wave sweeps across the cubicles and onto the software scene. Dress is suit and tie, hair combed, and discipline is the name of the game. Its roots are in the industrial quality movement, and it is everything free form is not. Free form has become unsafe, untidy, and ignorant as hardware limitations disappeared and complexity skyrocketed. Disciplined designers are prepared, methodical, and stuffy. The two worlds do not mix.

Back on the slopes another subtle but startling trend has arrived. Snowboarders bored by the ease of riding are strapping on skis. Skiers like Johnny Mosely are taking more air and stomping more radical jumps than borders. I donned a fat plank when my son picked up the sport and found a new exhilaration. The line drawn in blood between skiers and boarders is fading, giving way to increasing mutual respect. There is a new prestige associated with "multi-glissers" who can exploit a slope in any gear, under all conditions, at any time.

Perhaps our fellow bit benders could learn something from the alpine armistice. Software zealots and rebels who share the same passion are blind to the complimentary talents each hold. We are no longer trying to get *hello* on the terminal screen. Today's massive and complex software projects require both discipline and creativity. We should respect and cultivate a new breed of software developer who apply creativity and discipline to any project, on any platform, at the right time.

Will that happen before I prevail over "Grizzly," the 2001 Olympic downhill run at Snowbasin, Utah? Let the games begin.◆

– Gary Petersen, Shim Enterprise Inc.

The Vulnerabilities of Developing on the Net – Web site Applications –

Continued from page 10...

Table 1: Scanner and IDS Offering Examples

Product	Web site	Tool Type
Centrax	http://www.cybersafe.com/centrax24/index.html	Scanner IDS
CyberCop	http://www.pgp.com/products/cybercop-scanner/default.asp	Scanner
Dragon	http://www.securitywizards.com/intro.html	IDS
bv-Control	http://www.bindview.com/products/bvControl/internetsec/	Scanner
LANPATROL	http://www.netsecuritysys.com/products.html	IDS
Nessus	http://www.nessus.org/	Freeware scanner
NetProwler	http://www.axent.com/Axent/Public/Main?nav=Products	IDS
QualysGuard	http://www.qualys.com/info/qualysguard.html	ASP-based scanner
RealSecure	http://www.iss.net/securing_e-business/security_products/intrusion_detection/	IDS
Retriever	http://enterprisesecurity.symantec.com/products/	Scanner
SAINT	http://www.wwdsi.com/saint/	Scanner
Secure IDS	http://www.cisco.com/univered/cc/td/doc/pcat/nerg.htm	IDS
STAT	http://www.STATonline.com/	Scanner
SWARM	http://www.hiverworld.com/swarm/	Scanner

Table 2: Vulnerability Sharing Repository Examples

Site Name	Web site	Type
arachNIDS	http://whitehats.com/ids/ids.html	free IDS database
CERIAS Vulnerability Database	http://www.cerias.purdue.edu/	database
Fyodor's Playhouse	http://amy.insecure.org/index.html	hacker web site
Online Vulnerability Database	http://www.eSecurityOnline.com/	database
ICAT Metabase	http://icat.nist.gov/icat.taf	free web site
Bugtraq mailing list Database	http://www.securityfocus.com/bid/	mailing list database
PacketStorm	http://packetstorm.securify.com/	hacker web site
SWAT Database	http://www.axent.com/swat	database
Vigil@nce AQL	http://vigilance.aql.fr/	database
X-Force Database	http://xforce.iss.net/	free web site

Table 3: Alert and Advisory Services

Service	Type	Web sites
Bugtraq	e-mail list	http://www.securityfocus.com/forums/bugtraq/
Casandra	alerts	https://cassandra.cerias.purdue.edu
CERT Advisories	advisory	http://www.cert.org/advisories/
CyberNotes	monthly newsletter	http://www.nipc.gov/cybernotes/cybernotes.htm
Razor	advisory	http://www.razor.bindview.com/
S.A.F.E.R.	monthly newsletter	http://www.safermag.com/
SANS NewBites	e-mail list	http://www.sans.org/sansnews
Security Alert Consensus	e-mail list	http://www.networkcomputing.com
SecurityFocus Newsletter	newsletter summary	http://www.securityfocus.com/forums/sf-news/
SWAT Alerts	alerts	http://www.axent.com/swat
X-Force Alert	advisory	http://xforce.iss.net/maillists/

Table 6: CVE Editorial Board Composition

Site Name	Web site	Site Name	Web site
AXENT	http://www.axent.com	NFR	http://www.nfr.net
BindView	http://www.bindview.com	Microsoft	http://www.microsoft.com
CanCERT	http://www.cancert.ca	MITRE	http://www.mitre.org
CERIAS	http://www.cerias.purdue.edu	The Nessus Project	http://www.nessus.org
CERT/CC	http://www.cert.org	NIST	http://csrc.nist.gov/icat
Cisco	http://www.cisco.com	NTBugtraq	http://www.ntbugtraq.com
CyberSafe	http://www.cybersafe.com	PGP Security, Network Associates	http://www.pgp.com
DOD-CERT	http://www.cert.mil	SANS	http://www.sans.org
Ernst & Young	http://www.ey.com	Security Focus	http://SecurityFocus.com
Genuity	http://www.genuity.com	Silicon Defense	http://www.silicondefense.com
Harris	http://www.harris.com/harris	Sun Microsystems	http://www.sun.com
Hiverworld	http://www.hiverworld.com	Symantec	http://www.symantec.com
IBM	http://www.ibm.com	UC Davis	http://www.ucdavis.edu
IBM Research	http://www.research.ibm.com	Vista IT	http://www.vistait.com
ISS	http://www.iss.net	Zero-Knowledge Systems	http://www.zeroknowledge.com

ATTENTION AIR FORCE SOFTWARE PROGRAM MANAGERS!

Wrestling with legacy code?
Slaved to proprietary hardware?
Having difficulty hiring the talent you need?

The Air Force Computer Resources Support Improvement Program (CRSIP) is listening!

CRSIP invites your input to the Air Force's Software Technology Investment Strategy and your participation in a roundtable and panel discussion of software needs (technology, acquisition management, and process improvement) on Tuesday, May 1 from 12:00-1:00 p.m. during the 2001 Software Technology Conference in Salt Lake City, Utah. The complete conference schedule will be posted soon at www.stc-online.org

STC 2001 SM

To gain wider input to Air Force requirements for software improvement projects, CRSIP is interviewing engineers and program managers of software intensive programs for inputs. The results of the needs analysis will be summarized at STC, and used to structure the Air Force's investment strategy for software improvement. If you would like your input to be a part of this effort, see http://crsip.hill.af.mil for a questionnaire and more information.

Computer Resources
Support Improvement
Program (CRSIP)

CrossTalk / TISE
5851 F Avenue
Building 849, Room B04
Hill AFB, UT 84056-5713

PRSRT STD
U.S. POSTAGE PAID
Kansas City, MO
Permit 34