The Opportunities and Complexities of Applying Commercial-Off-the-Shelf Components

NEWS AT SEI

Authors

Lisa Brownsword

David J. Carney

Tricia Oberndorf

This article was originally published in News at SEI on: June 1, 1998

Government acquisition policies for software-intensive systems now emphasize the use of commercial off-the-shelf (COTS) products. On the surface, "the COTS solution" appears straightforward. In actuality, many projects find its use less than straightforward. This article provides acquisition managers and policy makers with a basic understanding of how developing systems with COTS products is different and why and what new capabilities are being identified.

Government acquisition policies for software-intensive systems now emphasize the use of existing commercial products. Requests for Proposals often require the use of specific COTS products and sometimes specify the amounts to be used. As systems are reengineered, many include the use of COTS products. And as government budgets shrink and the desire for increasingly complex systems continues, there is rising interest in leveraging the use of commercially available products whenever possible.

Although on the surface "the COTS solution" appears straightforward and compelling, projects that apply COTS find its use less than straightforward. Rather, they encounter significant new trade-offs and issues. Applying COTS products is not merely a technical matter for system integrators. It has a profound impact on business, acquisition, and management practices, and organizational structures. Compounding the problem is the limited experience and guidance currently available on how to effectively approach system development with commercial components.

The Software Engineering Institute (SEI), along with other key organizations associated with the government and civil agencies, is creating and assembling best

practice guidance for acquisition and program managers, integrators, and testers through case studies, hands-on support, and analysis. This article is one of several venues the SEI is leveraging to provide acquisition managers and policy makers with an understanding of what is different in the development of systems with COTS products and why and what new capabilities are being identified. Due to the brevity of this article, the discussion is limited to a few essential aspects of the differences and capabilities of COTS products.

What Is different with COTS-based systems?

COTS products can be applied to a spectrum of systems. At one end of the spectrum are nearly packaged software solutions, such as Microsoft Office or Common Desktop Environment (CDE), that require no integration with other components. This kind of system maps well to the needs and operations of the government.

Further along the spectrum are COTS products that support the information management domain, such as Oracle or Sybase. These systems typically consist of both COTS products and custom components, with COTS products making up the majority of the system. Depending on how well the COTS products and custom components fit together, a small to moderate amount of customization is usually required to enable them to work cooperatively.

At the other end of the spectrum, there are systems composed of a complex mix of commercial and non-commercial products that provide large-scale functionality that is otherwise not available. Such systems typically require large amounts of "glue" code to integrate the set of components. These systems are typically in the embedded, real-time, or safety-critical domains.

Using COTS products for applications at the packaged software solutions end of the spectrum is relatively straightforward; however, using them for complex systems further along the spectrum is not. This article focuses on the issues and complications that arise when constructing complex systems with COTS products.

Fundamental paradigm change

Traditionally, organizations develop systems from scratch with control over all or most of the pieces. They

- collect and define requirements.
- identify an architecture to satisfy the requirements.
- design in detail individual subsystems to fit within the architecture.
- code, test, and debug modules to meet the specified requirements.
- integrate sets of modules and subsystems into the complete system.

With the use of COTS as components for a system, a fundamental change occurs: an organization now composes the system from building blocks that may or may not (generally do not) work cooperatively directly out of the box. The organization will require skilled engineering expertise to determine how to make a set of components work cooperatively—and at what cost.

This fundamental shift from development to composition causes numerous technical, organizational, management, and business changes. Some of these changes are obvious, whereas others are quite subtle; but if not addressed, either type of change can cause severe problems for the project. Consequently, organizations may have to modify their procedures and structures and, in some cases, create entirely new procedures.

Impact on typical lifecycle activities

Regardless of which lifecycle model an organization uses (waterfall, spiral, or iterative), they perform requirements, architecture, detailed design, code, test, and system integration activities. The use of COTS products has a pervasive impact on all lifecycle activities. This is illustrated by briefly examining the impact to requirements, testing, and maintenance activities.

Requirements describe the desired system behavior and capability with a set of

specified conditions. For a COTS-based system, the specified requirements must be sufficiently flexible to accommodate a variety of available commercial products and their associated fluctuations over time. To write such requirements, the author must know enough about the commercial marketplace to describe functional features for which actual commercial products exist.

There is a critical relationship among technology and product selection, requirement specification, and architecture definition. If you define your architecture to fulfill your requirements and then select your COTS products, you may have only a few or no available products that fit within the chosen architecture. Pragmatically, three essential elements (requirements, architecture, and product selection) must be worked in parallel with constant trade-offs among them.

As the testing of COTS-based systems is considered, you must determine what levels of testing are possible or needed. A COTS product is a “black box” and therefore changes the nature of testing. A system may use only a partial set of features of a given COTS product. Should you test only the features used in the system? How do you test for failures in used features that may have abnormal behavior due to unknown dependencies between the used and unused features of a COTS product?

Maintenance also changes in very fundamental ways—it is no longer solely concerned with fixing existing behavior or incorporating new mission needs. Vendors update their COTS products on their schedules and at differing intervals. Also, a vendor may elect to eliminate, change, add, or combine features for a release. Updates to one COTS product, such as new file formats or naming convention changes, can have unforeseen consequences for other COTS products in the system. To further complicate maintenance, all COTS products will require continual attention to license expirations and changes. All these events routinely occur. All these activities may (and typically do) start well before an organization fields the system or a major upgrade. Pragmatically, the distinction between development and maintenance all but disappears.

Emergence of nontypical activities

We can view the commercial marketplace as a continuous “product conveyor belt”—the marketplace constantly adds new products and technology to the belt, existing products evolve through continuous upgrades, and vendors remove products from the marketplace. The government has limited influence (and no direct control) over the speed, content, or variety of products on the product belt.

Consumers, such as the government, must constantly keep abreast of the state of the product belt. This requires new activities (with associated resource requirements) in the area of technology and product evaluation. Consumers must identify potential technologies and products; qualify candidates for fit within their system; and perform trade-off analysis between competing technologies and products. An organization must continuously perform the entire process of monitoring, evaluating, qualifying, and analyzing the impact of technology and products given the constant changes within the commercial marketplace. We should add that technology and product awareness and evaluation are not activities that the government can merely relegate to its contractors. The government must also have such a capability if it is to specify and manage its systems wisely.

Assembling COTS products also presents new difficulties. Although software COTS products are attempting to simulate the “plug and play” capability of the hardware world, in today’s reality, software COTS products seldom plug into anything easily. Most products require some amount of adaptation to work harmoniously with other commercial or custom components in the system. The typical solution is to adapt each software COTS product through the use of “wrappers,” “bridges,” or other “glueware.” It is important to note that adaptation does not imply modification of the COTS product. However, adaptation can be a complex activity that requires technical expertise at the detailed system and specific COTS component levels. Adaptation must take into account the interactions among custom components, COTS products, any nondevelopmental item components, any legacy code, and the architecture including infrastructure and middleware elements. This adaptation process has a cost—a potentially high one.

What should an acquisition or program manager do to get started using COTS?

Applying a COTS solution requires the government to create and maintain new competencies. We have alluded to a number of essential capabilities throughout the article. The following sections identify a number of actions to establish an effective infrastructure for the use of COTS products. These actions are not intended as a road map or to be all-inclusive. Rather, they are a set of practical actions to help organizations start to develop the necessary knowledge and experience base. We recommend that organizations begin now—ideally before the first (or next) COTS-based system development, reengineering, or maintenance project.

Know the regulations

There are various policies, regulations, and directives relative to the general use of commercial products. Policies also exist concerning the use of specific COTS products such as the Distributed Information Infrastructure Common Operating Environment (DII COE). Understand what policies and directives apply to your particular systems. Situations or directives may change. Therefore, an organization should have a "regulations guru" available whose ongoing work is to remain informed of the various government regulations and their impact to the organization's systems.

Know your marketplace

The COTS marketplace is huge and continually changes. Determine what subsets of the marketplace are relevant to your systems. Develop dedicated resources to become conversant with the available and emerging COTS technologies and products, and determine their impact for your applications.

Know how to evaluate technologies and products

Determining which products and technologies are most appropriate for a given system requires more than a market survey based on marketing literature and vendor demonstrations. Know how to develop evaluation criteria, conduct a satisfactory evaluation, and select viable technologies and products based on your criteria. Determine the amount of time to allocate for evaluation during the acquisition. Leverage previous projects to experiment with developing the expertise required.

Know how to develop requirements

It is vital to understand how to specify requirements; this strikes the optimum balance between desired user functionality and the available COTS product. Know how to make trade-offs between COTS products, your architecture, and your requirements. Again, leverage previous projects to experiment with developing the required expertise. Know how to manage system and COTS product evolution Learn how the development and maintenance of your systems will need to change as a result of the continual release of COTS product updates. Some sample areas to investigate include scheduling of COTS product updates into your baseline, impact analysis to determine potential interactions and changes to existing components, impact analysis to the operations of the fielded system, and identification and management of licensing issues for your intended COTS products.

Determine how to build your business case

Although the motivation for the use of COTS products for many organizations is cost savings, an organization should address the many business unknowns prior to making that determination. How are the costs of both the initial and reoccurring adaptation throughout maintenance determined? How should a program manager make the business case if the total lifecycle cost is higher?

Develop a metrics database to determine your business case

Currently, there is little data as to the cost, schedule, or quality benefits of COTS-based systems. Begin collecting the data needed to develop a realistic business case. Such data might include cost, time distribution across lifecycle activities, defects after the system is fielded, or efficiencies gained or lost in field operations.

Final remarks

Even if an organization obtains some parts from commercial sources, a COTS-based system is still a system with requirements. Only the people who pay for, maintain, and use a COTS-based system are concerned about the quality of the system—vendors are

not. Organizations must still design, assemble, test, manage, and maintain the system. There is no magic. There is no COTS “silver bullet.” The government's responsibility for its systems is not eliminated or reduced by a reliance on COTS products.

COTS systems requires acquisition and program managers, policy makers, systems integrators, and system designers to become smart consumers by understanding the business, management, organizational, and technical implications of applying COTS products to system development or reengineering. The worst thing you can do is to treat the shift toward the use of COTS products as merely a change in technology.

About the authors

Lisa Brownsword is a member of the technical staff in the Dynamic Systems Program at the SEI. Before the SEI, she worked for Computer Sciences Corporation in support of the NASA/Goddard Software Engineering Laboratory. Prior to that, at Rational Software Corporation, she provided consulting to managers and technical practitioners in the use of software engineering practices, including architecture-centered development, product lines, object technology, Ada, and computer-aided software engineering (CASE).

David Carney is a member of the technical staff in the Dynamic Systems Program at SEI. Before joining the SEI, he was on the staff of the Institute for Defense Analysis in Alexandria, Va., where he worked with the Software Technology for Adaptable, Reliable Systems program and with the NATO Special Working Group on Ada Programming Support Environment. Prior to that, he was employed at Intermetrics, Inc., where he worked on the Ada Integrated Environment project.

Tricia Oberndorf is a member of the technical staff at the SEI. She is a part of the Dynamic Systems Program and concentrates on the investigation of integration and open system issues. She has investigated a number of other integration and open systems questions in the context of CASE environments and other kinds of systems. Prior to the SEI, she worked for the Navy for more than 19 years.

COTS Software Evaluation

NEWS AT SEI

Authors

David J. Carney

Bill Pollak

Kurt C. Wallnau

This article was originally published in News at SEI on: June 1, 1998

The use of commercial off-the-shelf (COTS) products in DoD systems

In recent years, the U.S. Department of Defense (DoD) has actively encouraged the more frequent use of commercial off-the-shelf (COTS) components in DoD systems whenever possible. Government acquisition policies for software-intensive systems now emphasize the use of existing commercial products. The intention of these policies is to avoid wasteful duplication of components and systems that exist in general commercial use, and to exploit commonality among software systems, especially in the domain of information technology.

Creating a large system from COTS components represents a marked departure from previous paradigms of government system acquisition and development. The SEI is working to help DoD organizations and their contractors optimize their use of commercial software and manage their risks adequately.

COTS products are becoming more important and widely used

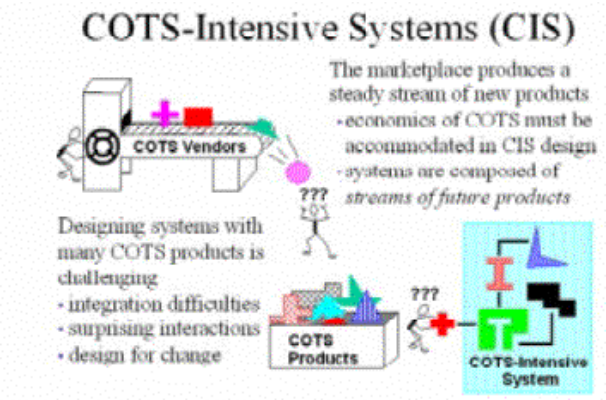


Figure 1: COTS-Intensive Systems

Although standards can and do improve our ability to integrate products, they will never, of themselves, make integration difficulties disappear. Moreover, in the commercial marketplace, innovation, not standardization, is often what distinguishes products and provides competitive advantage to software vendors.

As the software industry matures and grows, the COTS software marketplace is producing a stream of products and product innovations (see Figure 1). This process is gaining momentum and is showing signs of acceleration. An article in CIO magazine almost two years ago stated that more than 2000 new software products hit the market every month [1]. Keeping up with what is available is an enormous task.

Integrating COTS products in a system

To appreciate the complexity of integrating COTS products in a system, consider Lockheed Martin's work on the DoD Global Transportation Network (GTN)—a large-scale system that uses more than 50 COTS software products in the fielded system (many more products are involved if development tools are considered). Most of these products had at least one and sometimes more than one competitor in the marketplace, and many were quickly evolving. Furthermore, the requirements for the system were subject to change based on the capabilities of products emerging in the COTS software marketplace.

For the sake of argument, assuming a need for 50 products, if there were two other competitive products for every product that was selected, 150 product evaluations would be needed. Within the tight cost and schedule constraints of most system-development efforts, such a large number of products cannot possibly be examined to the depth required to understand all of the design implications.

To address this problem, Lockheed Martin was forced to develop new engineering practices to accommodate streams of complex, innovative, and unique software products. Design of the GTN system required managing multiple design options simultaneously, and required frequent assessments of the feasibility of any of these alternatives against capabilities found in the marketplace. Lockheed Martin's success with the GTN system hints at the direction toward which traditional design practices will have to evolve to accommodate the increasing prominence of COTS products in large-scale systems.

SEI work in COTS-based systems

Although some practitioners are quite sophisticated in their COTS product-selection practices, the mission of the SEI is to improve the entire practice of software system acquisition and development. A team from the SEI COTS-Based Systems (CBS) Initiative (Kurt Wallnau, David Carney, Ed Morris, and Patricia Oberndorf) has begun to identify best evaluation practices in a variety of domains as a way to help DoD programs such as the Navy JEDMICS (Joint Engineering Data Management Information and Control System) and the DARPA (Defense Advanced Research Projects Agency) EDCS (Evolutionary Design of Complex Systems) programs. The team also did a survey of best industry practice, based initially on insights gained from a cooperative research and development agreement between the SEI and Lockheed Martin.

[1] Bresnahan, Jennifer. The CIO Role: Mission Possible. *CIO* , October 15, 1996.
Evaluation is an inextricable part of design.

For more information

Contact Us

info@sei.cmu.edu

412-268-5800

Discussion with Members of the SEI COTS-Based Systems Initiative

NEWS AT SEI

Author

Bill Pollak

This article was originally published in News at SEI on: June 1, 1998

In this article, David Carney, Ed Morris, and Kurt C. Wallnau engage in a wide-ranging discussion about evaluation of COTS software. The views expressed in this article are those of the participants only and do not represent directly or imply any official position or view of the Software Engineering Institute or Carnegie Mellon University. This article is intended to stimulate further discussion about these topics.

The use of COTS products in commercial industry and in the DoD

How COTS software affects the development process Bill Pollak: In the introduction to the Features section, COTS Software Evaluation, we wrote about the Global Transportation Network (GTN) project. It's easy to see how such a project could get out of hand.

Kurt Wallnau: What they were doing with GTN reflects about the best way they could have handled the complexities they were facing. What's interesting is that from the outside, it did look like it was out of control; it did have certain characteristics of chaos. But when you spoke with the

KW: Wrong rules for COTS?

DC: Wrong rules for an NDI (non-developmental item) system. They had defined IPTs (integrated product teams). The company had a set of defined processes for product development, and since no one had told them not to use them, they were using these defined processes, and they were good old-fashioned waterfall life-cycle processes. But they were mandated to use a whole bunch of products from other sources. And it was just a fundamental mismatch.

Ed Morris: The only caution I have is that projects not doing COTS development often fail in the same way. Failure is not unique to COTS.

DC: Ed, this was six months late, seven months into the project! This was a pretty severe slip!

EM: The interesting thing is that most projects won't know they're six months late, seven months into the project.

KW: I think the rhetorical point is, here's a project that was COTS-based, or NDI, that was playing by the rules of existing thought on how to build systems, and they failed miserably. And then there's another project, which was building a much more massive NDI system—there were more than 50 COTS products in the fielded system—they were not playing by the usual rules, they were playing by a completely different set of rules, apparently, and they appear to be having some success; their customers are happy.

They were doing everything by the book, but it was the wrong book. They were playing by the rules, but they were the wrong rules.

Dave Carney: We did see one project that was out of hand despite being apparently very well in hand. They were doing everything by the book, but it was by the wrong book. They were playing by the rules, but they were the wrong rules. And in that sense, it was chaos.

DC: I think Ed's caution is a very valid caution. But I think it is safe to say, "Let's leave the comparison out of it." It is safe to say that they embarked on the project with the assumption that they were going to complete the project faster, if not better and cheaper, because they were using pre-existing components. And there was a radical mismatch between the reality of falling so far behind so quickly and the assumption that they were going to get there faster.

EM: The interesting thing is that COTS forces the life cycle forward, in fact. If this was a traditional waterfall development, they would be doing specifications. COTS forced them to implement earlier in the life cycle. Many traditional projects don't know that they don't know enough early on. And the interesting thing is, with this project, they could tell up front that they didn't know enough to be doing what they were doing.

DC: You see, I don't think it was a question of them not knowing what they were doing, it was that they didn't know how to do it. There were smart people all through that project, but there was the sense that everyone had their hands tied because nobody knew what to do next. And they were supposed to get a delivery of NDI from someone else, and it didn't come. And nobody knew who was in charge! And nobody picked up the phone and said, "Where's that thing?" It was amazing!

EM: In some ways, this is analogous to a single system being built by one organization in which different pieces are built at different sites. I mean, I could go back to my previous experience, where there were multiple sites, some working on the front end of the compiler and some working on the back end. And we were always getting delayed by someone who was not delivering a piece when it was expected. The difference is that, with COTS-based systems, there is no single source that says, "You will get this at some point in time."

DC: Well, if you take out all the good, but leave in the drawbacks to distributed development, it's probably intuitively true that many of those drawbacks will apply themselves to a COTS system. Because a COTS-intensive development is a distributed development, by definition. So it's prone to those kinds of drawbacks.

EM: It's true. It's distributed, and there's also no control over many of the components.

The need for product knowledge early in the life cycle

KW: One of the things that Ed said that I'd like to reinforce is that when you're dealing with COTS products, you want to try and move things forward, or up into the front end of the life cycle – whatever these things are. What I've seen is that the people who wanted to do the design, those who had the requirements for design at the abstract level, did not have detailed enough expertise on the individual products themselves, or what the capabilities and liabilities of the products were. Since they couldn't test feasibility of design, they started to work in a kind of vaporous level of

abstraction.

For example, they just assumed that things like browsers or CORBA would work, because these things have a really well-defined protocol, and there must be a way to map between protocols. Of course, things got a little more complicated than all of that. So they compressed the schedules because they expected to get the prototypes out within six months, because they were using off-the-shelf stuff and they expected to get a quick prototype. But when all the requirements feasibilities are compressed with detailed implementation, things start to behave in a way that's not expected.

We've had people from the SEI who were essentially like a "skunkworks" attached to the design process to test feasibility at very detailed levels, using model problems and so on, to help make design decisions. And I don't think that the program staff people were prepared to deal with that level of implementation complexity during the design. The members of the IPTs were at a high level of seniority in the group, so they felt that they were senior enough to specify requirements and do some sort of abstract design. But they were no longer really in touch with what was going on in the commercial marketplace.

BP: On another level, this is similar to some of the things that go on in the publishing world, where you take people who are used to designing for print and then you bring in desktop publishing; and when people get comfortable with desktop publishing, you bring in Web publishing. Some of this is simply future shock.

In the old world, you could draw your own boxes...with COTS, you don't have that freedom.

DC: I think what Kurt's getting at is that in the old world, you really could draw your boxes, and talk about the wings going here and the engines going there, and really not worry about the connections, and eventually down the road, all those details would get filled in. You can't do that anymore.

KW: Because you're in control of the mechanism, because you're going to design the whole cloth anyway.

DC: Right. And if you need water tanks in the wings, well, you add them later; you don't worry about that right now. But the thing is, with COTS, you don't have that freedom.

EM: The thing is, in the traditional process, the front end of the process acted as somewhat of a learning mechanism. Or at least at the point where you actually got to building something, you knew something about it. And as Kurt was saying, with COTS, you don't know enough at the point when you are implementing.

KW: And the other irony is, you start to compress things earlier on. You end up knowing less about it. You're trying to make big implementation commitments, instead of having a smooth commitment slope in which you gradually commit yourself to more and more design details. If you pick a product, or a set of products, it's kind of like a step function; you're bringing a large body of commitments in early, before you really have the basis to make those tradeoffs. And that's another consequence of moving things forward in the life cycle. Now, on the other hand, you could treat COTS as implementation details and mechanisms, and do the design, and then hope to find products. And I guess that's a feasible way of doing it; you just run the risk of not being able to find products that work.

BP: You mentioned that in your experience with JEDMICS (Joint Engineering Data Management Information and Control System), you employed kind of a skunkworks function.

KW: Yes, we had people from the SEI who provided the means for obtaining just-in-time product expertise to help in critical design decisions. A lot of design decisions require good, detailed understanding of the workings of the products. This is often not available to the designers, for a number of reasons. For one thing, it's just hard to keep track of the nuances of the products.

BP: So can you abstract from this experience the idea that it's good idea to have a

skunkworks function in any evaluation?

DC: Whether you want them to or not, there are some intricate implementation issues that arise, right at the very beginning, whereas in the previous way of designing systems, there was a high-level design that

you created while not knowing too much, and then you could leave that detail for the implementation. But the reality is, because of this compression, the implementation details are going to make themselves manifest regardless of whether you have the skunkworks function or not.

KW: I guess I hesitated because I'm becoming much more conservative as I get older about being categorical about saying, "This is how I do things." I think the issue is, how do you get that detailed knowledge at the time when you need it to make those design decisions? And what we're finding is working for us with JEDMICS is this use of a skunkworks function for solving model problems; to essentially provide ensemble solutions to these critical design problems. A skunkworks function can say "Not only can things work this way, but here are some good ways of doing it. Here are two or three options, because we know how these products fit together." Another approach would be to hire a consultant.

DC: There's another instance – a radically different COTS project. They wanted to build this COTS system, and this was a company that did large stock transactions. For them, having their system down for an hour might be, say, a $20 million loss. For a day, it might be an order of magnitude higher. So when they put their system together, they simply called up someone from the relational database company – there were other products involved, but we'll use the database as an example—and said, "We want to build a COTS system using your product, we don't want a custom-built system, but we need someone who knows everything internal about the product, every line of the code." And the guy from the database company said, "Well, that would be our Vice-President in Charge of Everything." So they asked how much this person cost, and the answer was, "Oh, he'd cost about a million dollars a day," and they said, "Thanks. We'll pay it." And they got this brilliant guy to put together the system, who knew everything about the internals. Now, that's a way to do it, if you can afford a million a day.

KW: But think about what that means. All that means is they needed to have deep, profound expertise about their products to make them work.

BP: How would you identify the abstract best practice to encapsulate what you're talking about?

KW: I think the best practice is making sure that you have that profound product knowledge available when you need to make design decisions, and be able to know when you need to have that knowledge. And there are different ways of getting that knowledge; one of them is hiring

expertise and the other is the skunkworks to develop just-in-time knowledge.

DC: I'd be more pessimistic. I'd say that you must be willing to realize that there is this black hole of cost—not just money cost, but cost—and try to be realistic to say, "Can we build the system given what we have? We don't have a million dollars a day for the best consultants." And if you can't, then what are the tradeoffs you can make? Should you not build the system, should you settle for a little bit less, and so on? It's all really an awareness issue. I'm agreeing with what Kurt said, but I'm suggesting that you may not be able to solve all your problems.

KW: That's a little bit darker.

DC: Well, that was what I was aiming for. Because in the ideal world, you should know what you need to do, and then go out and do it. But that doesn't always apply. You

may not be able to do it.

KW: I agree that there's a certain degree of uncertainty in building systems from COTS products that you need to accept, uncertainty about your knowledge of what the products do, about places where you have no knowledge at all. One of the uses of evaluation is to try to reduce the implications of uncertainty on the overall risk. I mean, how can you use evaluation to reduce or buy down that risk? Which also explains why a simple requirements-specification approach to COTS evaluation doesn't really get at the issue. Because on one hand, you may not know exactly what the requirements are. But also, you still have the fundamental problem of discovering whether the project actually meets that particular requirement. You're still kind of discovering. And that's just if you're looking at the product by itself. What we've discovered in JEDMICS and in other case studies is that the product-selection decisions aren't independent of each other; they're often dependent or co-dependent. Picking one product usually depends on picking another category of product A, B, or C, which usually has to do with the selection of another product some other place.

A framework for COTS evaluation?

BP: In our feature article, you categorized five ways in which COTS products influence the design phase of development for COTS-intensive systems. Is that the beginning of a framework for COTS evaluation?

KW: I think the danger of going in the direction of generality is that there is a general process that describes everything but that doesn't define anything in particular, and that's the risk.

DC: You go toward specificity, but there's a place at which you stop.

KW: Right, and we don't know where that is yet. We believe that evaluation helps people make decisions, and there are different kinds of decisions, therefore there are different kinds of decision aids, different kinds of evaluation techniques. We don't believe that any one of those techniques works for any and all situations. We don't believe that we're going to get so far in prescription that we're going to categorize all of the different areas of variability in your problem area, and be able to index this original collection of techniques. We made a stab at that way back ... about a year ago now. It was kind of a classification scheme; polarities and so on. The thing is, the space of variability is just a little too broad to be conveniently modeled in that way. So I think we're going to go somewhere in the middle. I think of a framework as a collection of concepts that say, "This is what's important, this is going to help you structure your understanding of evaluation. Here are some collections of techniques; here are some indicators of when they might be useful. Here are their strengths and limitations, and here are resource issues." To me, that would be a framework. Sort of a structured body of knowledge about evaluation, without necessarily being prescriptive.

We don't think we can ever have the definitive COTS development process that can be applied in all instances. But we certainly want to help people put together a process and select

techniques.

EM: What it comes down to is that we don't think we can ever have the definitive COTS development process that can be applied in all instances. But we certainly want to help people put together a process and select techniques.

DC: I don't think we'll ever have a universal process for this or for anything. But I do think that evaluation processes are absolutely doable, and our goal – I

The nature of software and perpetual change

BP: There seems to be something fundamentally different between, say, building a house and building a large software system. Somehow screws that are used in houses are different from software pieces that are used in large systems, and the key to understanding the kinds of problems that we've been talking about lies in understanding this difference.

DC: COTS software is bridges and tractors, it's not screws and pipes. doesn't really

exist with software the way it does with hardware; software is mostly conceptual.

DC: At least there has been very little of what has been called “product integrity” in software.

BP: What do you mean by “product integrity”?

DC: Well, a toaster toasts bread, for example. Generally, it doesn’t boil water. But word processors have spreadsheets, and spreadsheets have editing capabilities ... They tend to include more than a little, tiny, limited piece of functionality.

KW: “Integrity” means that, for example, a cell body has integrity if it's not leaking all over the place; I mean, is there a boundary around it so that we know what this is and what it does?

DC: In fact there's no boundary to what a word processing system does.

When we write software, we are conceptualizing something, and we conceptualize things differently. There is an external reality, but this external reality doesn’t really exist with software the way it does with hardware.

KW: There have been analogies of software to hardware, such as, “Why can’t software be like chips?” and “Why can’t software components fit together like hardware components?” And these analogies are quite useful on some levels, but often dangerous and misleading on other levels. When we write software, we are conceptualizing something, and we conceptualize things differently. And there are no universal laws. There is an external reality, but this external reality

KW: But it’s really not clear what products do these days. Look at Oracle. What does Oracle do?

DC: But you’re not talking about relational databases, you’re talking about the Oracle product. A relational database is a fairly well-defined chunk of functionality. But the Oracle product is a relational database, plus this bell, that whistle, and these little toots ...

BP: And you don’t buy pieces of functionality, you buy products.

DC: Yes. And that's just the way it is. It’s true in other fields, too; I mean, you don’t buy coils and a heating element, you buy a toaster. But because of any number of factors, one of which is age and maturity in the marketplace, toaster makers by and large don’t go too far afield.

KW: And toaster makers always make toast. But databases do lots of things.

BP: So what you’re saying is that it’s the nature of software itself that makes expansion of functionality possible.

DC: I’m saying it’s the nature of software that enables this phenomenon to occur. It’s the nature of software and the way it’s produced that makes it very easy for this spillover to happen. It's very easy. Think of all the little shell programs you've written that do something, and you think, “Hey, I can add this one little line, and it’ll do this too, and do that,” and then someone else looks at it and will say, from a different conceptual point of view, “Hey, wouldn’t it be neat if it could do that, too.” And it flips up on itself, and instead of doing what it was originally intended to do, all of a sudden, it’s over here. It’s very easy for that to occur in software.

KW: I mean, the market does tend to try to produce market integrity. So we kind of know what word processors do, and we kind of know what spreadsheets do, and we kind of know what relational databases do ... They differ substantially when they get into the features and what they actually implement, and that’s why the products are so hard to fit together, because the market produces those differentiators as well. It doesn’t serve anybody’s interests to make something completely compatible. Because it’s expensive to create these products, and you want your product to be hard to exchange, so that people will buy into it.

DC: I think the most interesting thing in the past six months is the way in the latest version of the Windows NT operating system, everything is done in the browser. The whole world looks like a Web site. Your whole world – your local file system, your local folder system – looks like a web site. This is a real change in perspective. And this kind of change happens all the time.

KW: We had some stable idea of the universe in which we lived, and then suddenly something came along and shook things up. And it's Java and the Web, maybe, today, and in four years we don't know what it's going to be, but we know it will be something because the market wants to produce these things, for whatever factors drive it.

DC: I don't think it's just market. There were two guys I knew who lived their lives through Emacs. Emacs was their shell. And you can do that. It's just that there's all these different points of view, and you can force an editor to be an operating system, and vice versa.

BP: Embedded software also affects the integrity of consumer products. I have hundreds of functions on my CD player that I never use. So, it used to be that a toaster was a toaster, but as software comes in, it becomes part of toasters and part of CD players...

KW: And we're going to find that our computers are going to be hooked into our TV sets, which are going to be hooked into our telephone systems. And before you know it, our toasters are going to be telling us when to put our coffee in.

DC: Well, you can schedule your toaster to call you at the office and come home to put something in to toast! All this stuff is not really science fiction.

KW: I don't know where this brings us to COTS evaluation, but it kind of leads us to think about these esoteric kinds of questions, like why things don't ever seem to fit together. We keep making progress, and we never quite get there, and once we really get close, as David suggested, something comes along and changes the end goals to something else. We were making progress with client-servers, and relational databases, and now we've got Intranets!

Ritualistic evaluation and make vs. buy decisions

KW: Ritualistic evaluation is much more commonplace than people realize. It showed up when I was a contractor. Quite often there is a clear front-runner in place. And there's a defensive mode that says "Oh, evaluation means that I have to have a list of things and I have to have a score." So you mentally start coming up with weights and categories of lists of things to match your intuition, because you've been told that's what evaluation is.

DC: I've read many of the directives, and the directives say "preference for," "use where possible," "where feasible," and so on. And as that goes down the bureaucratic food chain to someone who is lower in command, this "preference for" becomes an absolute mandate. The question is, does the mandate look at what is most apt or most fit, or are decisions being made for bureaucratic reasons?.

KW: This doesn't have anything to do with evaluation, it has to do with the decision to use COTS.

DC: Or the decision to buy COTS, which we have equated with evaluation. It's often the case that programs pick a product for nontechnical reasons, then perform an evaluation that justifies the selection. We have seen that in some cases. Or sometimes a project will make an evaluation, and even though there is a low functional fit, a product is chosen because the product is on hand.

KW: I'd say that they do the evaluation as sort of a post-hoc justification. This is far more common in Contractor Land than you'd imagine. I was a contractor, and I know this; you know what you want to bid, but you have

"Buy" brings in more than you think you'll be getting. What you think you're buying is just the tip of the iceberg that you can see.

BP: It illustrates just how different "buy" versus "make" really is. Because "buy" brings in all of these other things that we've been talking about.

DC: "Buy" brings in more than you think you'll be getting. When you buy, you don't just buy what you intend to buy, you buy all this other stuff. What you think you're buying is just the tip of the iceberg that you can see.

SEI Interactive, 6/98
http://www.sei.cmu.edu/interactive/Features/1998/June/COTS_Roundtable/Cots_Roundtable.htm
to provide an evaluation, and you'll jimmy the score to make it look right.

DC: After picking a product and committing to it, sometimes the vendor turns out to be an absolute bust; this can happen even with big, well-known vendors. The vendor might promise to send people to the program to tailor the product, then never show up; or the people that the vendor sends may not even know how to run the thing. Personally, I don't like any solution that is mandated without investigating alternatives, and for me that includes investigating the make-buy decision itself.

DC: Yes, but sometimes you can buy something other than what you think you're buying.

KW: Right. The world is never quite so categorical. In DoD systems, this option is more open to discussion; and really, it ought to be. But let me ask you this: Should the burden of proof be on a DoD program to demonstrate that it should or must use COTS, or should the burden of proof be on those who want to create custom-made systems?

DC: I think you're asking the wrong question. The question should be "How do we find a point to determine when it is most appropriate to buy vs. make?" And not place a burden of proof on either, but on both.

KW: I think you're right in an abstract and purely theoretical way. If we were dealing with a perfect world, we'd want to be completely informed before we make buy decisions. But, understandably, what we're dealing with is the natural inertia within the government/contractor community to sustain a large base of programmers, developers, and maintainers. And I think that unless you place some mandate that requires a fairly significant burden of proof for build decisions, you'll never overcome this inertia. So I think you have to say "If you can't show me why you can't use COTS, you have to use COTS."

You don't want to be in the business of building something if you can buy it. You want to be in the business of using, so you can build something else.

KW: I agree that there should always be a make-buy decision, but some commercial organizations would tell you that this is a no-brainer: You always buy. Because there's an economy of scale there. You don't want to be in the business of building something if you can buy it. You want to be in the business of using, so you can build something else.

DC: I agree that there may be a tendency for DoD programs to exaggerate their uniqueness. I think that, if you looked at it closely, you'd find that there are very few requirements that are specific to DoD. In terms of surge requirements, the DoD may be unique. But in terms of system functionality, for most things that are thought to be unique to the DoD, I'll bet I could find something analogous elsewhere. But on the other hand, as the ground shifts and it becomes required to use COTS products in systems, you're going to have some sustainment nightmares down the road. Program managers need to understand that there are ongoing sustainment costs in this new business model that have to be dealt with.

KW: We don't really understand the sustainment issues, the economics and sustainment of COTS, as well as we ought to. And it certainly ought to be part of what it means to evaluate a solution, based on one or more COTS products. I mean, how much is it going to cost you to sustain this thing? The question presupposes that you can predict the future shape of the markets and where things are headed, which you can't. All these sustainment estimates are based on market predictions. And who designs systems based on market predictions? We ought to, but we don't. That's

something we just don't know how to do at all. We're disagreeing here on whether it's a good idea to place a burden of proof on those who wish to do custom systems rather than COTS. All else being equal, you'd want to have enlightened decisions that were made on the basis of value. Still, there are good economic incentives to buy rather than make. Is the DoD's business personnel management? No. So the DoD should be out of the business of building personnel management systems. It shouldn't be building these systems.

DC: I'm not disagreeing that this is true for some systems. The point is, is the mandate being selectively applied, or appropriately applied?

KW: And it seems to me that in some cases, there ought to be a burden of proof. It might be in information management systems. "Show me why you should build one of these systems." Boeing builds airplanes. Ford builds cars. Airplanes are like Air Force airplanes; cars are like tanks. There's an assembly line, and they manage parts, and they manage things very well.

DC: Large banks clearly will write their own financial transaction systems. Or some pieces of it. What parts did they buy, and what parts did they build? I don't know. But I'm willing to bet it's not by mandate.

The use of COTS products in commercial industry and in the DoD

BP: How does the use of COTS products differ in commercial industry?

DC: Well, for one thing, industry organizations by nature care about a profit margin. And to the extent that future trends affect the profit margin, they care about them, so they will simply go in that direction. They are not constrained by government restriction, except in some cases. What is specific to the DoD is that the DoD has restrictions, legislation, and Congressional regulations to deal with. The idea that in buying something, it has to go out as a fair bid ... industry organizations don't have those kind of constraints. Someone says, "We want to buy something," and the CEO says, "Let's buy it."

KW: I don't know how to categorize this, but a curious thing happened to me about three weeks ago. A big financial insurance concern— mortgages, loans, stuff like that— came to us, and they wanted to know if we could do an architectural analysis and give them some COTS-system help. And it turns out that when they were talking about COTS, they named two or three major packages that they were going to buy and customize. They called it a COTS-solution system. And I asked about the other system they were building, the one where they didn't have COTS products, and they said that they had 150 programmers writing 2 million lines of Java. And I said, "Oh, you're using Java; are you using browsers?" and they said, "Oh yeah, we're using browsers" ... Basically, they listed a number of COTS products. And to them these things weren't COTS products at all. And of course, the DoD would consider that to be a major COTS decision.

DC: But let me take the other extreme of that. You wouldn't think about writing your own operating system (OS). For the B-2, the internals of the B-2, the DoD had to make a very clear decision about whether or not they were going to write their own OS—the one that ran inside the plane. Because nuclear weapons are involved. No one else would think about an operating system. My point is, you can't make a blanket statement about whether you should buy versus build unless you have some sort of context for it. And I think that none of the available real-time OSs worked. I know that they came to the issue and had to decide on it.

KW: They were making a real engineering decision.

DC: Right! That was a pure engineering decision. I think that in the really life-threatening nuclear systems, it's possible to make decisions more easily.

BP: So I might conclude from what you just said that in the commercial world, they use COTS more easily.

DC: I wouldn't have said that. They can make decisions more easily. And many of the decisions being made today are clearly that, for a wide variety of information

management systems, they use commercial products.

KW: Their incentives are clearer in industry.

BP: This is the statement that I've heard a number of times: "They've solved this problem in industry." That organization that Kurt was talking about—the big financial insurance concern—once they made the decision and were writing the 2 million lines of code, was it any easier for them than for the DoD?

DC: I think so, but I don't know.

BP: Is that because they have more experience?

DC: Actually, I think it's ... a lot of these places have less experience with building big systems. So in a sense they're newer kids on the block; they're willing to go along with the newer toy, I think. The DoD tends to be much more deliberate.

BP: Because if DoD systems screw up, worse things can happen.

DC: That's right. They're much more conservative. Industry organizations tend to be much more enthusiastic about things.

BP: And it looks like they're succeeding.

DC: Some are, but you don't hear about the failures. The stockholders are the ones who hear about the failures. A lot of companies do go belly up, and the government hasn't. But industry organizations don't publicize their failures. And there is no investigative board that really has to report it. The government gets, I think, worse press than it deserves, by comparison. And that applies to the DoD as well.

About the authors

Bill Pollak is a senior writer/editor, member of the technical staff, and team leader of the Technical Communication team at the SEI. He is the editor and co-author of A Practitioner's Handbook for Real-Time Analysis: Guide to Rate Monotonic Analysis for Real-Time Systems

(Kluwer Academic Publishers, 1993) and has written articles for the

Journal of the Association of Computing Machinery (ACM) Special Interest Group for Computer Documentation (SIGDOC) and IEEE Computer.

About the panelists

David Carney is a member of the technical staff in the Dynamic Systems Program at SEI. Before coming to the SEI, he was on the staff of the Institute for Defense Analysis in Alexandria, Va., where he worked with the Software Technology for Adaptable, Reliable Systems program and with the NATO Special Working Group on Ada Programming Support Environment. Before that, he was employed at Intermetrics, Inc., where he worked on the Ada Integrated Environment project.

Ed Morris is a member of the technical staff in the Dynamic Systems Program at SEI, where he is involved in the development of practices to support COTS evaluation. He has also worked with the CASE Environments Project at the SEI. He is co-author of Principles of CASE Tool Integration (Oxford University Press, 1994) and co-editor of IEEE 1348, Recommended Practice for the Adoption of CASE Tools. Before coming to the SEI, he worked at the Software Productivity Consortium, where he developed tools to model and predict the performance of Ada multitasking systems, and at SofTech, Inc., where he served as lead engineer for the development of an Ada runtime system and application support tools. Kurt C. Wallnau is a member of the technical staff in the Dynamic Systems Program at the SEI. Wallnau's current work in the SEI COTS-Based Systems Initiative is on product and technology evaluation practices and the design process for COTS-intensive systems. Before coming to the SEI, Wallnau was system architect for Central Archive for Reusable Defense Software (CARDS), a DoD software reuse program centered on the use of COTS software in application-specific architectures. He has published several papers relating to COTS software evaluation.

For more information

Contact Us

info@sei.cmu.edu

412-268-5800

The Era of Net-Centric Computing

NEWS AT SEI

Author

Scott R. Tilley (Florida Institute of Technoloy)

This article was originally published in News at SEI on: June 1, 1998

"In the future, network computers will be purchased

and used with the same enthusiasm as home

exercise equipment."

Scott Adams, The Dilbert Future: Thriving on Stupidity in

the 21st Century (New York, NY: HarperBusiness, 1997).

Is Mr. Adams right? Maybe. But it's important to

make a distinction between network computers

(NC) and Net-centric computing (NCC). An NC is

just one type of "thin client." NCC is more than just thin clients; it is an

emerging phenomenon whose effects will be profound and far reaching.

Almost anyone involved in computer science, information technology

(IT), or software engineering will be affected.

So what is NCC? The underlying principle behind NCC is a distributed

environment where applications and data are downloaded from network

servers as needed. This is in stark contrast to the current use of powerful

personal computers (PCs) that rely primarily on local resources. In some

respects, NCC resembles an earlier computing era of mainframes and

dumb terminals. However, there are important differences. NCC relies on

portable applications that run on multiple architectures ("write once, run

anywhere"), high bandwidth (for downloading applications on demand), and low-cost thin clients such as the NC, the NetPC, and Windows-based terminals (WBT).

SEI Interactive, 6/98

http://www.sei.cmu.edu/interactive/Columns/1998/June/Net_Effects/Net_Centric.htm

Thin clients

An NC uses Java for local processing. It was initially proposed by the "gang of four" (IBM, Netscape, Oracle, and Sun) as an alternative to the Microsoft/Intel duopoly. The NC vision was for a computer that did not run Microsoft Windows software, and it could use processors other than Intel's Pentium chips. This vision has at least one flaw: People still want to access their legacy applications (primarily Windows programs) and data.

The NetPC was an interim solution proposed by Microsoft and Intel to counter the NC. It is essentially a stripped-down PC with a sealed case. The selling point of the NetPC is a reduction in total cost of ownership because end users are unable to add or remove new hardware or software. It can be centrally administered and it can run Java applications if needed. In this sense, some consider it to be a better network computer than the NC itself.

The thin client that seems most likely to succeed is the WBT. A WBT is the thinnest of the thin clients, relying completely on a central server for applications and data. The WBT acts only as a display device, much like an X Station does on Unix. Early versions of WBTs are in fact current computers running Citrix Systems' WinFrame client. This application lets users on Windows, Macs, or Unix machines access a centralized computer running a modified multi-user version of Microsoft Windows NT Server. This technology (code-named "Hydra") is being rolled back into Windows NT 5 Server as the "Windows Terminal Server." I think it will prove successful because it lets organizations leverage their current IT infrastructure. The productive life of older 386- and 486-based PCs can be extended while newer machines are purchased incrementally.

Effects in the office

Irrespective of which type of thin client you adopt, how will NCC affect you in the office? Since almost everyone is affected by software these

days (like it or not!), you likely fit into at least one of the following three categories: user, developer, or administrator. For some users, the relief at not having to maintain a PC means they can instead concentrate on their primary tasks. Other users may chafe at the limitations that NCC brings with it. The removal of “personal” from PC means that users will no longer be able to significantly alter their desktop environments.

For developers, NCC offers an opportunity to greatly increase their customer base: An application written in an NCC-aware programming

SEI Interactive, 6/98
http://www.sei.cmu.edu/interactive/Columns/1998/June/Net_Effects/Net_Centric.htm

language, such as Java, means writing code once and having it immediately accessible on multiple platforms. It also means a different development environment, a new deployment model (renting applications versus buying), and new concerns about security. In a Netcentric world, security is not just for system administrators. For most developers, security is a quality attribute that is treated as an add-on to the system. For NCC, it needs to be treated as a first-class concern.

For administrators, NCC means a potential reduction in the cost and complexity of managing IT resources. The total cost of ownership issue has been cited as one of the motivating factors behind NCC, but so far little real data is available to suggest that NCC will be cheaper than today’s methods. It may in fact be more expensive because of the increased complexity of a heterogeneous and distributed environment.

Effects elsewhere

Outside of the office, the effects of NCC may prove even more significant. For software engineering, NCC offers a fundamentally new way of thinking about software. Basic issues such as version control need to be re-evaluated. For example, if a software application is being delivered to the user (and continually updated) using push technology such as Microsoft’s CDF or Marimba’s Castanet, what does it mean to say “the current version”? If the application is being monitored and updated in the manner of superdistribution, this may make software versions based on millisecond differences a reality.

Still not convinced this “NCC thing” will really affect you? Here’s another quote from Mr. Adams's book:

On the off chance that you are not familiar with the NC versus PC debate, allow me to provide some background. The NC is blah, blah, blah, Java, blah, blah, trying to screw Microsoft, blah, blah, no hard disk, blah, blah, Larry Ellison.

The “blahs” are Mr. Adams's, not mine. Larry Ellison is the CEO of Oracle and a big proponent of NCC in general, and of NCs in particular. Since the PC industry is driving many of the innovations in both academia and industry these days, the NC versus PC debate in the context of NCC will very likely affect you whether you follow the debate or not. There is currently a tremendous amount of discussion about NCs versus other types of thin clients. Time will tell which type will be the most popular, but history has shown that it doesn't usually pay to bet against the Redmond juggernaut.

SEI Interactive, 6/98

http://www.sei.cmu.edu/interactive/Columns/1998/June/Net_Effects/Net_Centric.htm

About the author

Scott Tilley is a visiting scientist at the SEI. He works with the Product Line Practice Initiative in the Reengineering Center, focusing on transitioning best practices in legacy-system reengineering in a disciplined manner. Before taking on this role, he was on leave from IBM and a member of the Rigi project in the Department of Computer Science at the University of Victoria. Tilley is the author of a 1993 book on home computing and has over 50 publications. He has a Ph.D. from the University of Victoria. He can be reached at stilley@sei.cmu.edu.

412-268-5800

Why Does Software Work Take So Long?

NEWS AT SEI

Author

Watts S. Humphrey

This library item is related to the following area(s) of work:

Process Improvement

TSP

This article was originally published in News at SEI on: June 1, 1998

In writing this column, I plan to discuss various topics of interest to software professionals and managers. In general, I will write about issues related to engineering productivity, quality, and overall effectiveness. Occasionally, I will digress to write about a current hot item, but generally I will be pushing the process improvement agenda. Because my principal interest these days is getting organizations started using the Personal Software Process (PSP) and Team Software Process(TSP), readers should know that a not-so-hidden agenda will be to convince them to explore and ultimately adopt these technologies.

Have you ever started what you thought was a two-or three-day job and have it stretch into a week or two? Before deciding you are just bad at estimating, look at how you spent your time. You will find you spend much less time on projects than you imagine. For example, on one project, several engineers used time logs to track their time in minutes. They averaged only 16 to 18 hours a week on project tasks. They were surprised because they all worked a standard 40-hour week.

This information soon turned out to be helpful. They were on a critical project and were falling behind. When they looked at the data, they found the design work took 50% longer than estimated. They knew they had a choice: either do the tasks faster, or put in more time. While there was pressure to race through the design, skip inspections, and rush into coding, the team resisted. They knew this would probably result in many errors and a lot of test time.

To meet their schedule, they needed to average 30 task hours a week. They all tried

Related Links

News

Heartbleed: Analysis, Thoughts, and Actions

TSP Symposium 2014 Goes Beyond Methodology to Focus on Software Quality

See more related news »

Training

See more related courses »

Events

Team Software Process (TSP) Symposium 2014

Nov 3 - 6

Team Software Process (TSP) Symposium 2014

Nov 3 - 6

valiantly to do this, but after Christmas, they realized that just trying harder would not work. They went on overtime and are now starting early in the morning, working late some evenings, or coming in on weekends. While they now average 30 task hours a week, they have to work over 50 hours a week to do it. They are also back on schedule.

Because this team had detailed time information, they could recognize and address their problem in time to save the project. The data identified the problem and pointed them toward the solution. Without good data on where your time goes, your estimates will always be inaccurate and you won't know how to improve.

Working Harder

When people say they are working harder, they actually mean they are working longer hours.

Barring major technology changes, the time it takes to do most tasks is relatively stable. The real variable is the time you spend on the tasks. But to manage the time you spend, you have to track it, and practically nobody does. Consider the following:

1. Our lives are filled with interruptions.
2. Software people do many kinds of tasks, and only some contribute directly to our projects.
3. Our processes are often informal and our working practices ad hoc.
4. Even if we wanted to, it is hard to do demanding intellectual work for long uninterrupted periods.

Interruptions

One engineer told me she had recently started to track her time and found she was spending much more time on interruptions than on her real work. For example, on one task of 108 minutes, her interruption time was over 300 minutes. This lost time, however, was not in big hour-long blocks but from an incessant stream of little 5- and 10-minute interruptions.

Interruptions come from many sources:

- telephone calls
- other engineers asking for help
- a coffee or rest break
- supply problems (i.e., printer or copier out of paper)
- equipment problems (the network dies)
- a power failure or a snow storm (everybody leaves)

Every interruption breaks your train of thought and takes time away from your work. With unplanned interruptions, you lose your place in the work and, when the interruption is over, you have to reconstruct where you were. This also causes errors.

For example, when I am in the middle of a design, I am often working on several things at the same time. While thinking through some logical structure, I realize that a name is misleading, a type must be changed, or an interface is incomplete. If I am interrupted in the middle of this, I often have trouble remembering all these details. While I have been unable to control the interruptions, I have found that maintaining an issue log helps me remember what I was working on when interrupted.

Non-Project Work

Most engineers also spend a lot of time on non-engineering tasks. Examples are

- handling mail
- setting up or updating their computing systems
- going to meetings
- looking for some specification, process, standard, or manual
- assisting other engineers

- attending classes

Few software development groups have adequate support. No one sets up or maintains his or her development system, few have groups to handle product packaging and release, and there is no clerical or secretarial support for mail, phone, or expense accounts. What is more, even when they have such support, many engineers don't know how to use it. This means that most of us spend more time on clerical and support work than on our principal development tasks. And every hour spent on these tasks is an hour we can't spend on development.

Lean and Mean Organizations

Often our organizations pride themselves on having very small support staffs. An almost universally accepted management axiom is that overhead is bad and should be eliminated. In the resulting lean and mean organizations, the engineers do their own clerical work. This is not an effective way to use scarce and expensive software talent.

By cutting overhead, management also eliminates the support staffs that funds in the overhead budget support. While some of these groups are not the least bit interested in supporting the engineers, many are. Eliminating them can have enormous costs. Among these costs is the time every engineer must spend sorting through email, answering the phone, getting supplies, doing expense accounts, and filing mail and documents. In addition to the lost engineering time, this also means that most mail is not answered promptly if at all, phones go unanswered, supplies are wasted or overstocked, and little if anything is properly filed or can be quickly found when needed.

Perhaps most expensive and annoying, every software engineer in such "lean and mean organizations" must set up and maintain his or her personal computing environment. Just because we have the skills to do this doesn't mean we should. Most of us could repair our cars or paint our houses if we chose to, but it would take us longer than using someone who does this for a living. And we have other things to do. Why should we have to handle our own computing support? The principal reasons that engineers spend less than half their time doing the tasks they were trained and hired to do is that, without adequate support, they have to support themselves. What is more, few engineers enjoy or are very good at being part-time clerks and technicians.

Ad-hoc Working and Planning

When no one has taken the time to define and document the organization's practices and methods, they must be maintained informally. When you come to a task that you haven't done before or at least not recently, you look around to see how it should be done. It takes time and a lot of interruptions to find someone with the right experience and get their help. While this is vastly preferable to bulling ahead without exploring prior experience, it does cut into the working week.

A related but slightly different problem concerns planning. When projects don't make detailed plans, and when engineers don't know precisely where they fit into these plans, they must do what I call continuous planning. In continuous planning, the key tool is not the PERT chart or Microsoft Project, it is the coffee machine. When you finish a task, you go to your continuous planning tool to have a cup of coffee. While there you decide what to do next. In the process, you talk to any other engineers who are also doing continuous planning and see what they think. You will usually get some good ideas, but if you don't, you can always interrupt someone.

The common view is that planning takes too much time. By not planning, engineers can immediately start on their programming work. While this immediate progress will feel good, you won't know where you are. Like driving in a strange country without a map, you have to stop at every turn and decide where to go next. All these short stops will take far more total time than a properly thought-out plan would have taken in the first place.

You Also Need an Occasional Break

Finally, creative development is hard work. When designing a product or a system, we need uninterrupted time. But we cannot design complex products for more than a few

hours at a time. The same is true of testing, reviewing, inspecting, coding, compiling, and many other tasks.

One laboratory decided to set up a dedicated group of experts to inspect a large and important product in final test. Every module that had test problems was sent to this group. For a while, they cleaned up a lot of defect-prone modules. Then, one of them later told me, they could no longer see the code they were inspecting. Everything began to blur. They even saw source code in their sleep.

Designing, coding, reviewing, inspecting, and testing are intensely difficult tasks. To have any hope of producing quality products, we must occasionally take breaks. But, to be reasonably efficient, and to do high-quality work, we need to control our own breaks, not take them when the phone rings or when somebody barges into our office or cubicle. Studies show that when engineers spend all their time on their principal job, their performance deteriorates. Some reasonable percentage of time on other tasks such as planning, process improvement, quality analysis, or writing papers can improve engineering performance. You will get more and better work done in the remaining 75% of your time than you would have accomplished in 100% of dedicated time.[1]

So, Keep Track of Your Time

To manage your personal work, you need to know where your time goes. This means you need to track your time. This is not hard, but it does require discipline. I suggest you get in the habit of using the time recording log, shown in Table 1 and Table 2. When doing so, enter the tasks and the times when you start and stop these tasks, and also keep track of interruption times. If you do this, you will soon see where your time goes. Then you can figure out what to do about it.

Manage Interruptions

Next, interruptions are a fact of life, but there are many ways to deal with them. Use "DO NOT DISTURB" signs and establish an ethic where everybody (even the managers) respects them. Forward phone calls or even unplug or turn off the phone. Also consider getting permission to work at home for a day or two a week.

Another way to manage interruptions is to get in the habit of using an issue-tracking log. Then, when you think of something you need to do, make a note of it in the log so you will remember to do it later and you won't forget it when the phone rings. While you will still have to handle these issues, you are less likely to forget them and you can do them at a planned time.

Also, use this same principle with interruptions. When someone calls in the middle of a design problem, tell them you'll get back and then make a note on a sticky so you don't forget.

Learn to Use Administrative Support

Learn how to use support. While few engineers have a support staff to help them, many who do don't know how to use them. If you have a support person, think about every clerical-type task before you do it. Can this person do it for you? Even though it may take longer at first, use them whenever you can. At first the result may need to be redone. But be patient and help the support people understand your problems with their work. It will pay off in the long run.

Plan Every Job

Perhaps most important, learn to plan. Plan your own work and urge your teammates and the project leader to start planning. Proper planning takes time, but it can save much more time than it costs. You will end up planning anyway, but it is much better to do it in an orderly way, and not at the coffee machine.

Vary Your Work

You can do demanding work only for so long. I lose my ability to do intense creative work after an hour and a half to two hours. I need to stop for a break or even to switch to some other kind of work. Further, during these intense sessions, frequent short interruptions offer no relief. It then takes an extra effort to reconstruct my thought

process.

What I suggest is to intersperse various kinds of work throughout your day. Do creative work when you are most fresh and productive and then switch to your email or an administrative task. Then perhaps do a design or code review possibly followed by a process-improvement task or data analysis. By varying the task types, your creative work will be of higher quality and you will actually get more done.

Define and Use a Personal Process

When you regularly make plans, a defined process will save a lot of time. The process provides a framework for gathering historical data and a template for making plans. And, by using historical data, your estimates will be more accurate.

Get and Use Historical Data

Finally, if you don't have administrative or technical support, use your time log to see what this lack costs you. Then tell your managers and show them your data. It might help them see the cost advantages of adequately supporting their engineers. Remember that the amount of work you produce is principally determined by two things:

1. the time the tasks will take
2. how much time you have available for these tasks

To manage your work, you must know where your time goes. Only then can you judge how much work you can do and when you will finish it.

Acknowledgments

In writing papers and columns, I make a practice of asking associates to review early drafts. For this column, I particularly appreciate the helpful suggestions I received from Dan Burton, Alan Koch, and Bill Peterson.

In Closing, An Invitation to Readers

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

Thanks for your attention and please stay tuned in.

Watts S. Humphrey
watts@sei.cmu.edu

Notes

1. For a brief discussion of this issue, see my book *Managing Technical People, Innovation, Teamwork, and the Software Process*, Addison Wesley, 1997, page 186. A more complete discussion is in Donald C. Pelz and Frank M. Andrews, Scientists in *Organizations: Productive Climates for Research and Development*, Wiley, 1966, pp. 56, 65.

2. Recording Log, see my book *A Discipline for Software Engineering*, Addison Wesley, 1995. This log is also discussed in *Introduction to the Personal Software Process*, also by me and published by Addison Wesley in 1997.

Table 1: Time Recording Log

Engineer		Date	
Program		Module	

Date	Start	Stop	Interruption Time	Delta Time	Phase	Comments

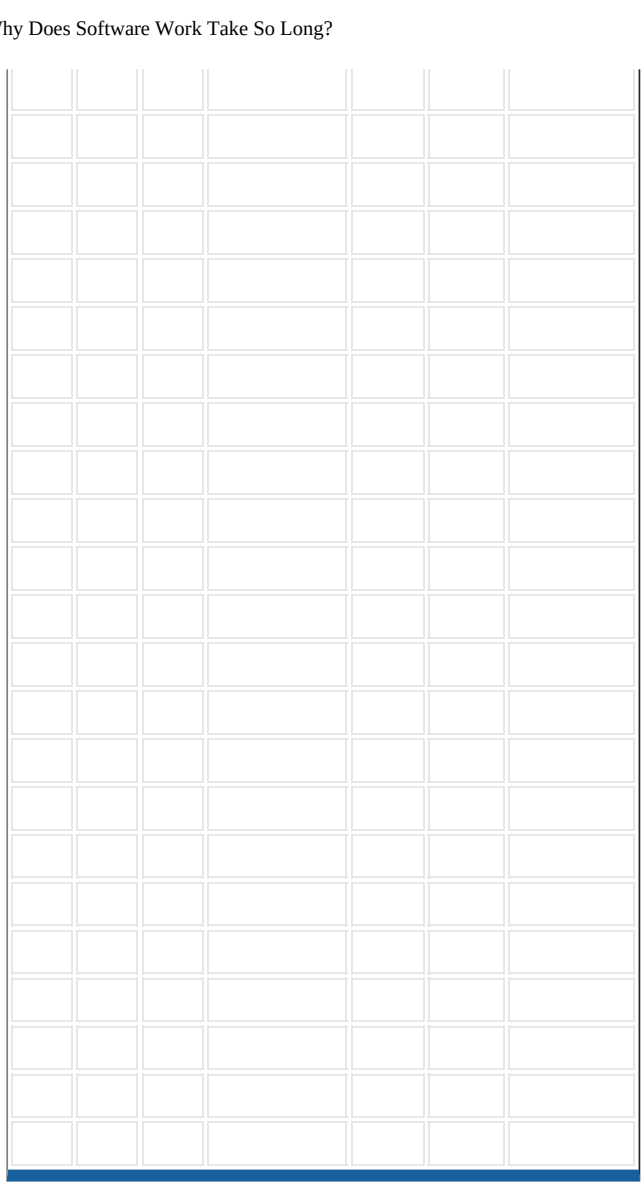


Table 2: Time Recording Log Instructions

Purpose	• Use this form to record the time spent on each project task. • Either keep one log and note the task and product element for each entry or keep separate logs for each major task.
General	• Record all the time you spend on the project. • Record the time in minutes. • Be as accurate as possible. If you need additional space, use another copy of the form.
Header	Enter the following: • your name • today's date • the project name • the name of the program or other product element

	If you are working on a non-programming task, enter the task description in the comments field.
Date	• Enter the date when you made the entry.
Example	• 4/13/98
Start	• Enter the time when you start working on a task.
Example	• 8:20
Stop	• Enter the time when you stop working on that task.
Example	• 10:56
Interruption Time	• Record any interruption time that was not spent on the task and the reason for the interruption. • If you have several interruptions, enter their total time.
Example	• 37 - took a break
Delta Time	• Enter the clock time you actually spent working on the task, less the interruption time.
Example	• From 8:20 to 10:56, less 37 minutes or 119 minutes.
Phase	• Enter the name or other designation of the phase or step you worked on.
Example	• planning, code, test, etc.
Comments	• Enter any other pertinent comments that might later remind you of any unusual circumstances regarding this activity.
Example	• Had a requirements question and had to get help.
Important	• Record all worked time. • If you forget to record the starting, stopping, or interruption time for a task, promptly enter your best estimate.

About the Author

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

For more information

Contact Us

info@sei.cmu.edu

412-268-5800

Security Matters – Doesn't It?

NEWS AT SEI

Authors

James Ellis

Moira West Brown

This article was originally published in News at SEI on: September 1, 1998

Security Matters will focus on the topic of network security and its relationship with and impacts on the network environment on which we all rely so heavily. Security is an underlying but often overlooked factor that touches so many aspects of day-to-day activities on intranets and the Internet that the list of possible topics is virtually endless.

I'll be joined by other members of the SEI Networked Systems Survivability Program wishing to contribute to this column, to discuss and debate network security topics of interest to them. We'd love to hear what topics you'd like us to discuss and debate. So if you're interested in topics such as forming a computer security incident response team, responding to security incidents, secure programming practices, security improvement, the security impacts of emerging technology, or another network security topic, then send me your suggestions in email!

Moira West-Brown

Striking up conversations with new people is always interesting; you never know where the discussion will lead. Invariably though, the question of "What line of business are you in?" will come up. When faced with the reply "Internet security," the next response often takes the form, "That's interesting, but security isn't really something that matters in our organization." It is hard to imagine an organization for which this statement is true. The truth is that many organizations don't think about security until after they have suffered a computer security incident. Then their eyes open to reality – security is a

fundamental foundation that supports business operations.

The Internet

Our daily reliance on technology has changed a great deal in the past decade with the advent and proliferation of the Internet. Just think about your home or office environment 10 years ago and how it differs today. Home computers were out of the price range of most people's pockets, and few of us had dedicated desktop systems. Today, the Internet has become one of the most powerful and widely available communications mediums in the world. Governments, corporations, banks, and schools conduct their day-to-day business over the Internet.

The Internet is easy and cheap to access, but the systems attached to it lack a corresponding ease of administration. As a result, many Internet systems are not securely configured. We know of a case where someone setting up a workstation took a lunch break and by the time he returned, the system had already been compromised! Additionally the underlying network protocols that support Internet communication are insecure, and too few applications make use of the limited security protections that are currently available. The combination of the data available on the network and the difficulties involved in protecting the data securely make Internet systems vulnerable attack targets. As a result, it is not uncommon to see articles in the media referring to Internet intruder activities that result in financial loss, data corruption, loss of public confidence, and potential loss of life.

With such widespread use of the Internet and its inherent insecurity, the information that resides on and flows across the network is increasingly sensitive and at risk. Clearly, security does matter to you as your banking and securities transactions, your medical records, your company's proprietary data, and your personal correspondence are at risk.

Catch-22: keeping pace with security needs

What is more worrying is that Internet security improvements haven't kept pace with the dramatic increase in the use of the network and the need to protect the integrity, authenticity, and privacy of the data that traverses it. How did this imbalance came about in the first place? It boils down to a lack of demand: vendors providing only the level of security that their customers have been willing to pay for. The price was high; vendors didn't just have to address the issue of protecting their own systems and applications, they also had to overcome the inherent insecurity of the underlying Internet protocols by building a secure communications channel on top of the existing infrastructure. So when asked, the vendor community responded that it would include security if enough customers required it in purchases. But most customers didn't know what to ask for and some didn't see the point in asking for security that wasn't available.

Over the years this Catch-22 scenario has perpetuated. The good news is that at last, there are signs that this cycle is now being broken, and there is hope that slowly this will start to redress the imbalance of Internet use and security.

Signs of progress

What happened to break the deadlock? Progress is partly a result of vendors responding to the negative publicity that they received as a result of security vulnerabilities in their products. The advent of electronic commerce on the Web has also played a part. But the biggest factor is the strong push for improved security standards spearheaded by business-to-business communications and commerce needs. Fortune 500 companies are finding that they can save large amounts of money by moving to network-based data communications and storage--indeed they will stop being competitive if they fail to move. The demand for security standards to be established and incorporated into vendor products was a result of their clear understanding of the need for their communications to be secure and protected.

Today the effect is being felt most strongly in the area of data encryption and public-key infrastructures (PKI). These are the areas that will most directly affect business-to-business communications, contracts, and commerce. Over the next couple of years we expect to see encryption and PKI technologies much more widely available and used--for everything from digital signatures on contracts to private email. We hope that this will also focus more attention on other areas of security technology that need similar improvement.

What you can do

How does this affect your ability to secure and protect your data? There are basically two things you can do. First, you can use security as a discriminating factor in system and software procurement. It is now economically feasible to include security considerations during purchase negotiations. You don't need to be a security expert to do this. It is as simple as asking vendors what security they offer. They'll respond with details on the standards they conform to and the integrated security functionality they provide and explain at length the differences between their products and their competition's. Secondly, you can encourage your organization to participate in security standards efforts to be sure they address your business needs.

So the future is looking much healthier for Internet security. We'll soon have the integrated security tools and secure architectures so desperately needed, and doing a good job of securing a system won't require a masters degree or 10 years of on-the-job training! The hope is that one day, systems will be sufficiently secure out of the box that system administrators will be able to take lunch breaks without fear of an intrusion!

But the future hasn't arrived yet! We can't become complacent as we still have to cope with today's problems. This means that we must all be vigilant and do our best with the limited tools and insecure architectures that are available to us. You can take steps to secure your data, but it is neither easy nor cheap. It requires spending time and resources on securing individual systems, communications equipment, and your overall network infrastructure.

In future issues of this column, we will discuss efforts underway in the security community to improve security, the things you need to consider, and what you'll have to do and be aware of to protect your data. A security nirvana may never be attainable; even with the improvements available today and others on the horizon, we have to continue to work at maintaining security.

About the authors

Moira J. West-Brown is a senior member of the technical staff within the CERT® Coordination Center, based at the SEI, where she leads a group responsible for facilitating and assisting the formation of new computer security incident response teams (CSIRTs) around the globe.

Before coming to the CERT/CC in 1991, West-Brown had extensive experience in system administration, software development and user support/liaison, gained at a variety of companies ranging from academic institutions and industrial software consultancies to government-funded research programs. She is an active figure in the international CSIRT community and has developed a variety of tutorial and workshop materials focusing mainly on operational and collaborative CSIRT issues. She was elected to the Forum of Incident Response and Security Teams Steering Committee in 1995 and is currently the Steering Committee Chair. She holds a first-class bachelor's degree in computational science from the University of Hull, UK.

James T. Ellis is a vulnerability analyst at the CERT® Coordination Center located at the SEI. His primary focus is on understanding the root causes of security vulnerabilities, determining how they can be avoided in systems, and transitioning that information to software developers. He is a past general chair for the Symposium on Network and Distributed System Security sponsored by

the Internet Society.

Before joining the SEI, Ellis worked at the Pittsburgh Supercomputing Center, also located at Carnegie Mellon, where he was responsible for system performance and security for the Unicos operating system on a Cray Y-MP/832 supercomputer. Before coming to Carnegie Mellon, Ellis was the manager of computing facilities at the Microelectronics Center of North Carolina (MCNC).

Are Software Architects Like Building Architects?

NEWS AT SEI

Author

Mario R. Barbacci

This library item is related to the following area(s) of work:

Software Architecture

This article was originally published in News at SEI on: September 1, 1998

Welcome to *The Architect*. The purpose of this column is to discuss software architecture and software architecture analysis. We will emphasize practical results that can be used by practicing software designers, developers, and other stakeholders involved in developing the architecture of software systems.

We will feature opinion pieces by SEI staff and guests, techniques, case studies, announcements, etc. Future articles will highlight emerging techniques for architecture analysis developed by the SEI (Software Architecture Analysis Method, Architecture Tradeoff Analysis Method) and their application to systems developed by SEI customers as well as techniques developed by other organizations.

Mario R. Barbacci

For the past several years, the SEI has successfully used its Software Architecture Analysis Method to evaluate software architectures. Why should we care about software architecture analysis? We should care because a software system architecture is not the same as the software system, and a great looking "building" might be impossible to build! The intent of architecture analysis is to detect as early as possible

Related Links

News

SATURN Conference Announces Additional Keynote, Conference Scholarships

Distinguished Speakers, Strong Technical Program Set for SATURN 2014

See more related news »

Training

Big Data - Architectures and Technologies

Documenting Software Architectures - eLearning

See more related courses »

whether we are dealing with an illusion or a feasible design. However, architecture analysis is no panacea—even if the architecture passes the analysis hurdle, the implementation might not be adequate! Positive results are only a temporary "go ahead," and every result must be confirmed by measurements or more detailed analyses at a later phase of the life cycle. Can we make the complementary statement? That is, if an architecture does not pass the hurdle, no amount of hacking will compensate for the deficiency.

Inevitably, the focus on architecture analysis leads us to view the software engineer as a kind of architect of a software system. In this column, I want to suggest that this analogy may not be entirely accurate, and that there may be a more precise analogy available to us in the emerging field of architectural engineering.

Definitions

Webster's [Webster] defines "architect" and "architecture" thus:

> Architect: 1: a person who designs buildings and advises in their construction. 2: a person who designs and guides a plan or undertaking.

> Architecture: 1: the art or science of building; specifically: the art or practice of designing and building structures and especially habitable ones. 2a: formation or construction as or as if the result of conscious act <the architecture of the garden>. b: a unifying or coherent form or structure <the novel lacks architecture>. 3: architectural product or work. 4: a method or style of building. 5: the manner in which the components of a computer or computer system are organized and integrated.

Although the emphasis is on "buildings" and "habitable structures," the software engineering community has adopted the terms and attempted to adapt the definitions to software systems. Consensus definitions are still emerging, but we seem to be comfortable with the terms and use them frequently. We do this in part because architects of buildings have been "architecting" for a very long time and buildings tend to stay up--obviously, they must be doing something right.

It is common for software engineers to mention building architecture as a model to follow, often using Gothic cathedrals as examples! But as Bass, Clements, and Kazman point out [Bass], "Analogies between buildings and software systems should not be taken literally; they break down fairly quickly. Rather, such analogies help us understand that the viewer's perspective is important and that structure can have different meanings depending upon the motivation for examining it."

The analogy is not entirely without merit. In both cases, multiple stakeholders care about the architecture of the "building." The stakeholders (e.g., buyers/sellers, users/producers, builders/maintainers) care about fitness along multiple dimensions (e.g., size, maintainability, strength) and uses (e.g., new and legacy systems, one of a kind, and product lines) throughout the life cycle. But the analogy works only at a superficial level, because we tend to idealize building architectures and the role of the building architect--there is a lot more engineering in modern buildings than architecture purists might care to admit. Perhaps we would do better to model the role of a software architect on an emerging profession that bridges the two camps of architecture and engineering. But first, let's take a short detour on the origin of the problem.

The Encyclopedia Britannica [EB] provides a wealth of material on the topic of architecture, and it is interesting to read the articles under the topic "The Art of Architecture."

> The characteristics that distinguish a work of architecture from other man-made structures are (1) the suitability of the work to use by human beings in general and the adaptability of it to particular human activities; (2) the stability and permanence of the work's construction; and (3) the communication of

> experience and ideas through its form.
>
> All these conditions must be met in architecture. The second is a constant, while the first and third vary in relative importance according to the social function of buildings. If the function is chiefly utilitarian, as in a factory, communication is of less importance. If the function is chiefly expressive, as in a monumental tomb, utility is a minor concern. In some buildings, such as churches and city halls, utility and communication may be of equal importance.

The three characteristics are derived from Vitruvious' s *firmitas*, *utilitas*, and *venustas* (i.e., structural stability, appropriate spacial accommodation, and attractive appearance), although the relative order of the terms (and by implication, which of these characteristics has a primary or supporting role) varies according to the function. (Although this might seem obvious today, the ordering was a matter of controversy for several centuries as different theories of architecture were proposed in which any of these three characteristics had an absolute priority over the others.)

But M. C. Belcher points out [Belcher] that the role of the architect is more ambiguous. Traditionally, the architect was a master in control of all functional, structural, and aesthetic decisions; the method of construction; and the supervision of the building process. This tradition survived until the 19th century, where the complexity of the application of structural steel forced architects to collaborate with steel experts (civil engineers). The primacy of the architect was further eroded in the 20th century by the growth in complexity of building environmental systems (e.g., passenger elevators); architects now had to collaborate with mechanical and electrical engineers as well. Engineers in these disciplines were experts in their subject matter but not on buildings and could not assume the role of the architect. The need for people whose professional focus was on the design of buildings but whose education as engineers allowed them to master the technologies and materials in structural, mechanical, and electrical systems led to the emergence of architectural engineering as a new profession.

What is Architectural Engineering?

> Architectural engineering is the application of engineering principles to the design of technical systems of buildings. The profession of architectural engineering includes practicing engineers designing, managing, and constructing mechanical, electrical, or structural systems for buildings. The profession also includes engineers educated as mechanical, electrical, or civil engineers who practice the application of engineering principles to the design of building systems. Some of the great architectural engineers of past and present are Gustave Eiffel, Buckminster Fuller, Ove Arup, and Santiago Calatrava. [NSAE]

This definition is taken from the National Society of Architectural Engineers (NSAE). This is a young profession -- the NSAE was incorporated only in 1984 and at present there are fewer than 20 accredited architectural engineering programs in the U.S. Nevertheless, things are moving fast in the new profession and, on October 1st, the NSAE will merge with the Architectural Engineering (AE) Division of the American Society of Civil Engineers (ASCE) to form the Architectural Engineering Institute (AEI). (It is an odd feeling to learn of a profession younger than ours--the ACM and the IEEE Computer Society are both over 50 years old!)

Belcher [Belcher] characterizes architecture and engineering as "two dissimilar disciplines which must work together due to the vast array of aesthetic and technical needs of a complex modern building." The dissimilarity stems from the motivations and training of traditional architects and engineers. Architects are trained to think top down, to synthesize a global solution, which may be later refined or abandoned in light of emerging information. Engineers are trained to analyze available data and to solve problems bottom up, following a more systematic process toward a single, "best" solution. Belcher writes,

> Architectural engineers, however, receive training in both analysis and synthesis. They take courses in the theory and practice of aesthetic design side by side with students of architecture; they take courses in calculus, physics, and material sciences right along with students in traditional engineering fields. Thus they understand the creative process and are able to use that mode of thinking when appropriate. They are also taught to analyze, so they know the importance of that mode of thinking and when to use it.

Finally, architects do not think of themselves as engineers. The American Institute of Architects offers career advice to architects and includes suggestions for architects that might be considering a change:

> Sometimes interests and experience lead architects beyond the conventionally defined edges of the profession into other professions and occupations. The aim may be to "function as an architect in the [fill in the blank] field" or to shift disciplines more completely. Very often, a shift in career requires additional education or training, new credentials, or professional registration or certification. While there are no limitations on where or how far an architect may migrate from the field, some migration paths are more common and clearly marked than others.

One of the more common and clearly marked paths? Go back to school and become an engineer!

> The design of buildings often requires the special expertise of civil, structural, mechanical, and electrical engineers. The options for engineers are many, and there is considerable demand for them in industry--from corporate clients and manufacturers of building products--as well as in private practice. Engineers offering design services for building projects typically practice as independent consultants in one of the areas listed or as a specialty engineer (e.g., acoustical, illumination, fire protection). Some engineering firms combine two or more of these areas; some offer architecture, construction management, or design/build services as well.

Conclusion

It might be misleading to say that a software architect is like a building architect. The software architect is more likely to be trained in an engineering or science school and is usually familiar with one or more of the technologies (e.g., security, communications, storage, operating systems) used to build the system. Chances are she came from one of these fields before being promoted to software architect and she could change places with technology/domain experts on a different project. This is different from traditional building architects who come from different schools and have different training than say, steel/concrete/heat, ventilation, and air conditioning engineers.

Is architectural engineering and the motivation and training of an architectural engineer a better analogy for software architecture and the activities of a software architect?

About the Author

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

Barbacci is a fellow of the Institute of Electrical and Electronic Engineers (IEEE), a member of the Association for Computing Machinery (ACM), and a member of Sigma Xi. He was the founding chairman of the International Federation for Information

Processing (IFIP) Working Group 10.2 (Computer Descriptions and Tools) and has served as vice president for Technical Activities of the IEEE Computer Society and chair of the Joint IEEE Computer Society/ACM Steering Committee for the Establishment of Software Engineering as a Profession. He was the 1996 president of the IEEE Computer Society. He is the 1998-1999 IEEE Division V Director.

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

The views expressed in this article are the author's only and do not represent directly or imply any official position or view of the Software Engineering Institute or Carnegie Mellon University. This article is intended to stimulate further discussion about this topic.

Capability Maturity Model Process Improvement

NEWS AT SEI

This library item is related to the following area(s) of work:

Process Improvement

CMMI

This article was originally published in News at SEI on: September 1, 1998

The Software Engineering Institute (SEI) is a federally funded Research and Development Center with the mission to accelerate the most effective technology and practice of modern software engineering. The SEI is funded primarily by the Department of Defense (DoD) but also accepts work from other government organizations as well as the private sector via Cooperative Research and Development Agreements.

The centerpiece product of the SEI has been the Software Capability Maturity Model (CMM) released in 1991. This model has contributed to widespread success in assisting organizations in improving their efficiency in developing quality software products. The success of the Software (SW) CMM spawned other CMMs that address a wide range of subjects.

A CMM provides an organization a conceptual framework within which specific processes, e.g., configuration management and quality, can be optimized to efficiently improve the capability of organizations.

A CMM provides state-of-the-art practices to

- Determine the maturity of an organization's processes.
- Establish goals for process improvement.
- Set priorities for immediate process improvement actions.
- Plan for a culture of product or service excellence.

Please note that current and future CMMI research, training, and information has been transitioned to the CMMI Institute, a wholly-owned subsidiary of Carnegie Mellon University.

Related Links

News

SEI to Co-Sponsor 26th Annual IEEE Software Technology Conference

SEI Cosponsors Agile for Government Summit

Training

See more related courses »

Events

Team Software Process (TSP) Symposium 2014

Nov 3 - 6

By focusing on specific processes, an organization can best leverage the resources for their improvement activities while rallying the organization around specific goals. A CMM can be a road map showing an organization how it can systematically move to more mature levels of performance and do it in more effective and efficient ways. After an objective assessment, an organization can set its goals for increasing the capability of its processes. To the DoD, this translates into more affordable products and services for our war fighters. CMMs can include processes that span the entire lifecycle. Starting with requirements management, they can span the breadth of product development, ensuring quality, lean production concepts, and support to the field. Each individual process includes elements that provide basic practices as well as additional practices that add incremental benefits and maturity.

When these processes are sufficiently matured, the organization increases its performance or maturity. Subsequent to the success of the SW-CMM, other CMMs were developed with SEI support. These CMMs included the Systems Engineering CMM and the Integrated Product Development (IPD) CMM. It became apparent in the development of these and other models that they all contained common processes, e.g., configuration management, quality, and requirements management, supporting the various functional disciplines, software engineering, and systems engineering. Improvements in these common processes could benefit other disciplines. Further, it became apparent that process improvement resources applied to one functional discipline, e.g., software engineering, could be beneficial to another functional discipline. The common elements used in a software CMM appraisal could be used for a systems engineering appraisal, and there would be no need to redo the appraisal of common elements. In addition, improvement efforts based on unique CMMs could result in suboptimization, confusion, and potentially unnecessary expenditure of process improvement resources.

Acquisition reform in the DoD created a significant paradigm shift away from a “how-to” mentality approach to an approach centered on Statements of Objectives and Performance-Based Requirements. The earlier capability models and standards were clearly used in the context of meeting contract requirements. There were even brief attempts to use them as selection criteria or as compliance benchmarks rather than frameworks to identify and define characteristics of good practices that facilitate process improvement. Remember the Requests for Proposals that required an SW-CMM Level 2 or above to propose?

Although DoD Directive 5000 directs we select capable suppliers, it does not direct how it should be determined or set arbitrary levels. DoD has learned over time two important things about maturity levels:

- Many organizations have benefited from the use of CMMs as process improvement tools resulting in delivery of improved products to DoD and government.

- Many projects or products delivered by organizations, purported to be at the SEI Level II or Level III, have not met the customers' requirements.

One of the top-priority projects in the SEI is integration of the CMM products for use in single or multiple functional disciplines. Industry and government along with the SEI now have enough experience in the various functional disciplines to build this framework upon which all present and future CMMs can be based. This will greatly enhance the efforts of CMM users and protect the resources already invested.

Organizations can use their previous CMM process improvement work and tailor their future efforts to their unique organization. The initial common framework effort will be based on the SW-CMM, the SE-CMM, and the IPD-CMM. Other functional disciplines may be added later. To efficiently use the government funds allocated to CMMs, further work on CMMs that are not common framework compliant has been halted. The work accomplished to date in Software CMM, Version 2.0 and the IPD CMM have been

included in the initial CMM Integration (CMMI) baseline.

In building these CMMI products, the needs of industry and government partners must be understood and met. We have had extensive participation in our reviews of the CMMI requirements, and broad collaborative efforts are underway developing the products. We are depending on the functional discipline experts from industry and government to assist in building the products.

In summary, the CMMI project requires a broad collaborative effort to ensure that the best practices are included and process improvement resources are optimized. Industry along with government and the SEI are participating on a team to build the CMMI products. Since many organizations have already made considerable investments in CMM-oriented process improvement efforts, it is important that the products of this project efficiently integrate into these efforts, and that resources are not wasted on a new approach.

About the Author

Mark D. Schaeffer has over 20 years experience in weapons systems acquisition and program management in the Office of the Secretary of Defense, Naval Sea Systems Command, and as congressional staff. He has been the deputy director for systems engineering since November 1994 and is responsible for policy and implementation of systems engineering, technical risk management, design for manufacturing quality, reliability and maintainability, manufacturing, and acquisition logistics.

CMM Integration (CMMI) Framework

NEWS AT SEI

Authors

Roger Bate

Sandra Shrum

This library item is related to the following area(s) of work:

Process Improvement

CMMI

This article was originally published in News at SEI on: September 1, 1998

This article provides a high-level technical description of the CMMI framework, which is currently being developed to accomplish the goals of the CMMI Project. Keep in mind that the CMMI Project is not complete and, thus, the framework and other aspects of the CMMI product suite may change before they are released.

The CMM Integration (CMMI) Project has been underway since February 1998. Representatives from government, industry, and the SEI are combining several existing process improvement models under a single architectural framework. This framework will sort, combine, and arrange process improvement elements to form outputs that serve the individual needs of organizations. These elements include the core building blocks for process management and integration as well as elements specific to certain functional disciplines (such as software engineering or systems engineering).

The CMMI framework is one part of what is called the "CMMI product suite." In addition to the CMMI framework, the product suite consists of capability models, training products, assessment materials, a glossary, and tailoring guidelines. These informational elements are what populate the framework in a raw form and compose the outputs of the framework in their finished form.

The purpose of the CMMI product suite is to serve the users of capability models (CMs) and Capability Maturity Models[sm] (CMMs) better than the independently created CMs and CMMs now available. The framework's integrated approach will simplify the

Please note that current and future CMMI research, training, and information has been transitioned to the CMMI Institute, a wholly-owned subsidiary of Carnegie Mellon University.

Related Links

News

SEI to Co-Sponsor 26th Annual IEEE Software Technology Conference

SEI Cosponsors Agile for Government Summit

Training

See more related courses »

Events

Team Software Process (TSP) Symposium 2014

Nov 3 - 6

process of understanding and using multiple models and provide integrated and tailorable process improvement tools for the development user community.

The process improvement models resulting from the framework will be called “CMs,” not CMMs. Other similar changes in terminology will be necessary to combine existing models and supporting materials into an integrated CMMI framework and product suite.

The CMMI framework development process

As Figure 1 illustrates, framework developers from government, industry, and the SEI are engaged in a project to combine the wealth of information about software engineering (SW), systems engineering (SE), integrated product and process development (IPPD),[1] assessment techniques, and training techniques to develop a CMMI product suite. This suite will be capable of producing CMs tailored to the needs of user organizations.

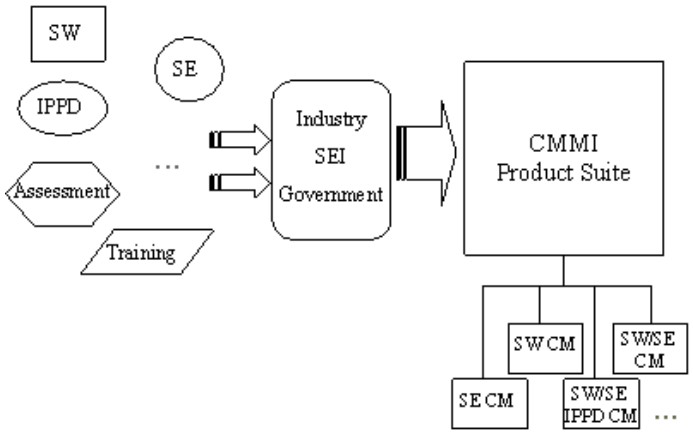


Figure 1: CMMI framework development process

Notice that the outputs at the bottom right of Figure 1 show various combinations of software engineering, systems engineering, and IPPD. Over time, more elements will be added to the mix and thus will add to the available outputs of the framework. These outputs will include CMs covering additional disciplines (e.g., security systems engineering, hardware engineering) and various combinations of these disciplines. These CMs, because they are developed within the CMMI framework, will share common terminology, components, and assessment methods.

Development goals

Before the development team created the framework, they identified the goals that the framework must meet. Overall, the CMMI product suite must be an improvement over the use of independently developed CMs and CMMs.

The goals of the CMMI Project are to eliminate inconsistencies, reduce duplication, and lessen the cost of implementing model-based process improvement. Further goals are to ensure that the products of the framework are easy to understand and use because they use common terminology, have a consistent style, follow uniform construction rules, and share common components among products. Finally, the team is trying to minimize the impact on those who are using existing models, assessment materials, and training materials.

Therefore, developers are striving to maximize commonality, maintain the look and feel of existing materials, provide assets for future models, design a logical organization for assets, and establish and follow configuration-management guidelines.

The vision of the CMMI product suite is that a framework user can generate capability models and their supporting training and assessment materials as needed from the framework's common elements and discipline-specific elements. In other words, if an organization wanted to improve its software engineering, systems engineering, and

integrated product and process development, it could combine these three disciplines into one capability model supported by one set of training materials and assessment materials.

By eliminating unnecessary duplication of common activities, the job of training people to use the models will be simpler. The common terminology used in the models, training materials, and assessment methods will simplify the introduction of process improvement activities based on these models, thereby reducing the costs of adoption.

The structure of the framework

The CMMI framework is designed to provide an internally consistent set of common elements that apply to any discipline and that must be included in any CMMI product. These CMMI products will support process improvement activities, including assessments and training. The CMMI framework currently consists of four parts: the input process, repository, control process, and output process. Figure 2 illustrates the structure of the framework:

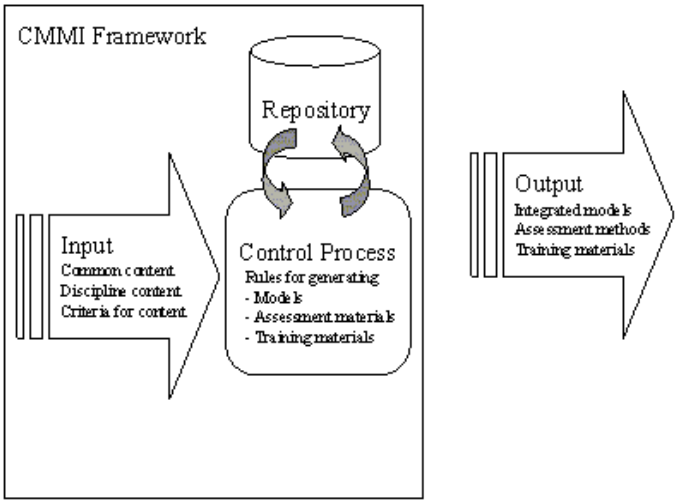


Figure 2: CMMI framework

The repository contains the components of capability models, training materials, and assessment materials, as well as construction rules and the conceptual architecture of the outputs. The control process combines the inputs and the rules for generating capability models, assessment materials, and training materials to produce output that can be applied to an organization's process improvement efforts.

How the framework functions

Imagine that the framework is a capability model generator. As a user of the CMMI framework, you would specify various options based on the needs of your organization, such as disciplines that need to be covered, staged vs. continuous, and inclusion of the IPPD environment. The framework would then generate the capability model that best meets your needs. Figure 3 represents the information that resides in the framework and how it is processed to produce tailored capability models.

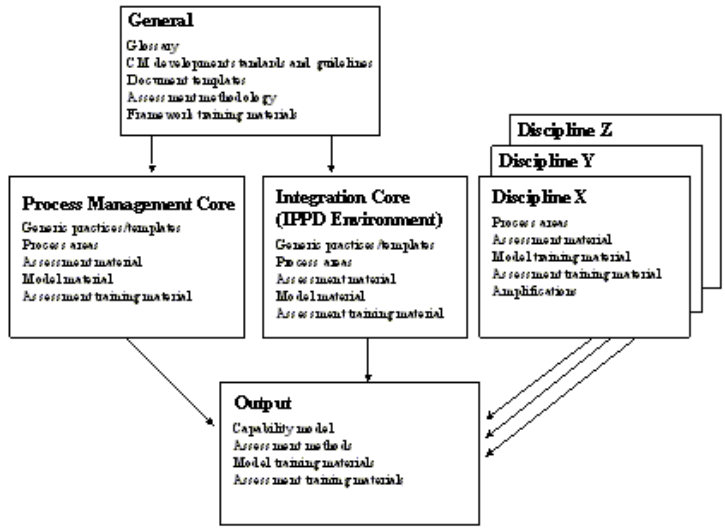


Figure 3: How the framework processes information

The entire middle row of boxes represents the essence of the process improvement information that is eventually used by the user organization. This information includes the process management core, the integration core, and any number of disciplines.

The *process management core* contains process management components that apply to all disciplines and all domains. These components are automatically included in your capability model.

The *integration core* contains information about IPPD, which can be applied in virtually any discipline or domain.

The *disciplines* represent specific information that you can select to include in your capability model. The initial CMMI product suite will include only two disciplines: software engineering and systems engineering. However, the framework is being designed so that it can accommodate new disciplines over time.

Essentially, the framework sorts, combines, and arranges information to make it useful for you and to tailor the information to your needs.

Staged versus continuous representations

In every process improvement model, regardless of representation, the basic building blocks are process areas. How these process areas are presented in the model can be considered its "representation." The two dominant representations used in existing process improvement models are *staged* and *continuous*.

Naturally, framework developers had to decide whether to use staged or continuous representations in the framework's capability models. Each representation has its proponents and detractors in the development community.

In a *staged* representation, process areas are grouped into stages (or maturity levels). Each process area contains practices that, when performed, achieve the purpose of the process area. Within a stage, the institutionalization practices for all constituent process areas must be achieved to successfully achieve the entire stage. Once an organization has achieved the entire stage, it has reached a capability maturity level. The Software Engineering Institute's CMM for Software (SW-CMM) is an example of a staged model.

In a *continuous* representation, process areas also contain practices that, when performed, achieve the purpose of the process area. Generic practices are grouped into capability levels. These practices are added to the practices of each process area to attain a capability level for each process area. Capability levels are achieved process area by process area. Although the order in which process areas are addressed is not required to follow a particular sequence, the order may follow recommended staging. The Electronic Industries Alliance's Interim Standard 731, Systems Engineering

Capability Model (SECM), is an example of a continuous model.

End users of the framework may prefer a CM that has a staged or a continuous representation. Likewise, there are two models that initially compose the input to the framework: one is a staged model (SW-CMM) and one is a continuous model (SECM). Given these existing models, the best way to serve the end user was to provide each model in both staged and continuous representations.

Products resulting from the CMMI framework

The products that result from the CMMI framework include capability models, training materials, and assessment materials.

Capability models

Each capability model consists of process areas, specific practices, generic practices, capability levels, stages, maturity levels, discipline-specific amplifications, and descriptive material. Initially, the CMs that will be available from the CMMI framework are

- · a Software Engineering CM
- · a Systems Engineering CM
- · an Integrated Software and Systems Engineering CM
- · an Integrated Software and Systems Engineering CM that incorporates the principles of IPPD

These CMs will support both the staged and continuous representations.

Training materials

The training package you will receive from the framework will be tailored to the CM that you have chosen. Each training package will contain training for the CM (including both staged and continuous approaches), training for the assessment team, and training for lead assessors.

Assessment materials

The assessment package you will receive from the framework will also be tailored to the CM you have chosen. Each assessment package will contain materials for assessment planning (including requirements and methodology), data collection methods and tools for both staged and continuous approaches (e.g., questionnaires, interviews), analysis methods, and team qualifications.

Other materials

The framework also will provide materials that support development of CMs in other functional disciplines. These materials include a glossary, CM development standards and guidelines, document templates, assessment methodology, and training materials on the use of the framework. Likewise, the core components (process management core and integration core) are available to be incorporated into the new CMs.

How will the CMMI framework be used?

The CMMI framework is designed to be used both by those who use CMs for process improvement in their organizations and by those who develop new CMs.

CM end users are interested in using the tailored capability models, training materials, and assessment materials for process improvement in their organizations. The framework's tailored CMs and supporting materials are used by end users to write policies, conduct process improvement programs, create process documentation, and persuade their organizations to adopt process improvement.

CM developers are interested in developing and maintaining CM products, validating their models, supporting state-of-the-practice reports, or entering new discipline-specific information into the framework. The CMMI framework will be an effective tool

for constructing CMs that use common terminology, common elements, and standard construction rules.

The CMMI framework will provide the ultimate user of these products with several benefits: simple and consistent CMs, common terminology across multiple disciplines, and the ability to spread this useful improvement technology across all of their disciplines. What's more, the framework will ease the complex task of building new CMs and the associated training and assessment materials.

For more information

To learn more about what is happening with the CMMI project and development of the CMMI framework, visit the CMMI Web site.

About the authors

Dr. Roger Bate was the chief architect of the Enterprise Process Improvement Collaboration (EPIC), which developed the Systems Engineering Capability Maturity Model (SE-CMM) and the Integrated Product Development Capability Maturity Model (IPD-CMM). Bate was a Texas Instruments (TI) fellow and the chief computer scientist for TI, where he headed a corporate-wide project to improve the software development process. Bate is also a fellow of the Association for Computing Machinery (ACM) and a fellow of the Society for Design and Process Science.

Sandy Shrum has been a senior writer/editor at the SEI since September 1995. Before working at the SEI, she was a senior information developer at Legent Corporation where she supported business-critical networking systems and maintained technical and user documentation. She has a BS in business administration and marketing from Gannon University and an MA in professional writing from Carnegie Mellon University.

[1] Integrated product and process development (IPPD) is a management technique that integrates all development activities ranging from product concept to product support. The IPPD approach uses multifunctional teams, called integrated product teams (IPTs), to improve the product and its development and sustainment processes. The goal of this improvement is to meet the organization's cost and performance objectives. IPPD evolved from concurrent engineering and is sometimes called integrated product development (IPD).

For more information

Contact Us

info@sei.cmu.edu

412-268-5800

Introduction: Capability Maturity Model Integration

NEWS AT SEI

This library item is related to the following area(s) of work:

Process Improvement

CMMI

This article was originally published in News at SEI on: September 1, 1998

In 1987, the SEI released a software process maturity framework and maturity questionnaire to support organizations in improving their software processes. Four years later, the SEI released the Capability Maturity Model for Software (SW-CMM). Since its release, the SW-CMM has significantly influenced software process improvement worldwide.

However, not all problems in product development and maintenance are caused by lack of attention to the software engineering process. A broader, system-wide view of problems and approaches to process improvement is necessary. In recent years, the SEI became involved in helping to develop additional capability maturity models in other functional disciplines, including systems engineering [The Electronic Industries Alliance's Interim Standard 731, Systems Engineering Capability Model, (SECM)] and integrated product development (IPD-CMM), to establish a common frame of reference for accelerating broader organizational learning.

The need for CMM integration

The SW-CMM is a "roadmap" that describes "evolutionary stages" consisting of key practices that guide organizations in improving their software capability. The SECM, in contrast, shares many of the same principles, but was written to address the needs of a different community: the systems engineering community. This had two consequences. First, the SW-CMM and SECM overlap; for example, both deal with requirements, project management, process definition, etc. The two models provide somewhat different guidance in places where they overlap, but the reason for the difference isn't always clear. Second, the SECM is based on a different representation, one that describes the entire "process area terrain" with less emphasis on exactly how an organization might mature through that terrain. Process areas span levels rather than being defined within a maturity level as in the SW-CMM. This is referred to as a "continuous" representation. The SW-CMM is said to have a "staged" representation.

Please note that current and future CMMI research, training, and information has been transitioned to the CMMI Institute, a wholly-owned subsidiary of Carnegie Mellon University.

Related Links

News

SEI to Co-Sponsor 26th Annual IEEE Software Technology Conference

SEI Cosponsors Agile for Government Summit

Training

See more related courses »

Events

Team Software Process (TSP) Symposium 2014

Nov 3 - 6

Ideally, CMMs should work together harmoniously for the benefit of organizations wishing to apply more than one CMM to improve product quality and productivity. However, the overlap in content and difference in architecture and guidance have made it difficult to apply both models in integrated process improvement. To meet this need, industry, government, and the SEI are now involved in the development of a framework to harmonize and efficiently incorporate the practices of different capability maturity models. Through the CMM integration (CMMI) effort, capability maturity modelers in other engineering communities will be able to develop models that are compatible with those already established.

In a recent article in *CROSSTALK*, Mark E. Schaeffer, Deputy Director for Systems Engineering in the Office of the Secretary of Defense/Acquisition and Technology (OSD/A&T), explains the rationale for CMMI:

It became apparent in the development of these...models that they all contained common processes, e.g., configuration management, quality, and requirements management, supporting the various functional disciplines, software engineering, and systems engineering. Improvements in these common processes could benefit other disciplines. Further, it became apparent that process improvement resources applied to one functional discipline, e.g., software engineering, could be beneficial to another functional discipline. The common elements used in a software CMM appraisal could be used for a systems engineering appraisal, and there would be no need to redo the appraisal of common elements. In addition, improvement efforts based on unique CMMs could result in suboptimization, confusion, and potentially unnecessary expenditure of process improvement resources.

The CMM integration effort

The CMMI Project is a collaborative effort sponsored by the Office of the Secretary of Defense/Acquisition and Technology (OSD/A&T), Systems Engineering, with participation by government, industry, and the SEI. The project's objective is to develop a product suite that provides industry and government with a set of integrated models and related improvement products to support process and product improvement.

The initial outputs from this project will include

- a framework for generating CMMI products
- capability models (CMs) supporting both the staged and continuous representations:
  - a Software Engineering CM
  - a Systems Engineering CM
  - an Integrated Software and Systems Engineering CM
  - an Integrated Software and Systems Engineering CM that incorporates the principles of integrated product and process development (IPPD)[1]
- training products
- assessment materials
- glossary
- tailoring guidelines

The work accomplished to date in SW-CMM Version 2.0, SECM Version 1.0, and the IPD-CMM Version 0.98 (draft) have been included in the initial CMMI baseline.

In this section

For an introduction to the topic of CMM integration, see our Background feature, Capability Maturity Model Process Improvement by Mark E. Schaeffer, Deputy Director for Systems Engineering in the Office of the Secretary of Defense/Acquisition and Technology (OSD/A&T). This article was originally published in *CROSSTALK*, May 1998.

For more detailed information about the framework itself, see the Feature article in this section of *SEI Interactive*, CMM Integration Framework by Roger Bate and Sandy Shrum. This article provides a technical description of what we are currently doing to accomplish the goals of the CMMI Project.

In our Roundtable feature, four members of the CMMI Steering Group—Steering Group Chair Phil Babel, Joan Weszka of Lockheed Martin, Hal Wilson of Litton PRC, and Mike Zsak of OSD—engage in a wide-ranging discussion about the CMM integration work. This article presents the goals and rationale for the CMMI Project and describes how CMM integration will benefit organizations that use CMs.

Our Links feature offers a guide to online information and resources about CMM integration, including links to the CMMI area on the SEI Web site, your most current source of information about the ongoing CMMI Project.

[1] Integrated product and process development (IPPD) is a management technique that integrates all development activities ranging from product concept to product support. The IPPD approach uses multifunctional teams, called integrated product teams (IPTs), to improve the product and its development and sustainment processes. The goal of this improvement is to meet the organization's cost and performance objectives. IPPD evolved from concurrent engineering and is sometimes called integrated product development (IPD).

For more information

Contact Us

info@sei.cmu.edu

412-268-5800

Discussion with Members of the CMM Integration (CMMI) Steering Group

NEWS AT SEI

Author

Bill Pollak

This library item is related to the following area(s) of work:

Process Improvement

CMMI

This article was originally published in News at SEI on: September 1, 1998

Related Links

News

SEI to Co-Sponsor 26th Annual IEEE Software Technology Conference

SEI Cosponsors Agile for Government Summit

Training

See more related courses »

Events

Team Software Process (TSP) Symposium 2014

Nov 3 - 6

Discussion with Members of the CMM Integration (CMMI) Steering Group

Moderated by Bill Pollak

In this article CMMI Steering Group members Philip S. Babel (Aeronautical Systems Center, Air Force Material Command), Joan Weszka (Lockheed Martin Corporation), Hal Wilson ('Net Solutions, Litton PRC),and Michael G. Zsak, Jr. (Office of the Secretary ofDefense, Acquisition & Technology) engage in awide-ranging discussion about the CMMI Project. Theviews expressed in this article are those of theparticipants only and do not represent directly or implyany official position or view of the Software EngineeringInstitute or Carnegie Mellon University. This article isintended to stimulate further discussion about thesetopics.

The systems engineering context

Bill Pollak (moderator): A strong emphasis of the CMMI work is the need to consider software engineering in a systems engineering context. Why do you think this is so important?

Phil Babel: In the services and the OSD, we are involved in the development of defense systems, which involve concurrent hardware, software, and other implementation technologies. Since all of these have to be done in parallel, some people call it concurrent engineering. The systems engineering process is really the bigger process that oversees the rest of the subprocesses, so software engineering

could be thought of as a piece that has to fit inside systems engineering. That's especially important because the front end of the software engineering process is initiated by the systems engineering delivery of a good, solid set of requirements and top-level designs. And on the back end of the development process, the testing and integration phase is really a systems engineering process. So software engineering kind of fits in the middle and runs in parallel with hardware engineering. And I think that's why it's so important that we have an integrated process improvement strategy.

Joan Weszka: I think what we're all trying to achieve is enterprise-level process improvement. And to do that, we need to take advantage of synergies across all the disciplines, including software and systems engineering, in order to maximize the benefits of process improvement. Rather than taking a stovepiped, single-discipline approach, taking an integrated, enterprise-level approach to process improvement should yield much greater efficiency as well as benefit. A first logical step for those involved in software systems development is to adopt an integrated process and improvement strategy for systems and software engineering.

BP: I assume that these conclusions are based on experience with trying to apply the CMM in organizations.

JW: At Lockheed Martin, we have used the individual, discipline-specific models, so we understand the cost and benefits gained. We also have considerable experience in applying integrated product development (IPD), where we deploy integrated product teams having representation from all disciplines across the life cycle, and we have some tremendous success stories in that arena. We expect to see analogous advantages accrue as a result of using the integrated CMMI models since we'll be exploiting commonality and leveraging across the disciplines just as we do with IPD.

MIKE ZSAK: I think if you look at the major change we made with our DoD policy documents back in '96, we brought all of what used to be the stand-alone stovepipes, like software engineering and reliability and maintainability (R&M), under the umbrella of systems engineering. So actually, our policy architecture is now structured so that software engineering is one of the engineering disciplines considered as part of systems engineering. It fits in with configuration management, and R&M, and all of the other disciplines. There are a number of reasons we did this, the first of which is efficiency—we had some experience where folks would work their particular discipline, like R&M or others, and they would optimize their discipline, but the system would be suboptimized. Also, when you look at what goes on in software engineering and many other functional disciplines, you find that they have a lot in common with what is done at the system level—the general thrust and the objectives are very similar. For example, the approach used for risk management in software engineering is very similar to the approach used in systems engineering. You may get into unique implementation methodology, but the basic approach and concepts of risk management are the same in both cases.

BP: So there's an opportunity to exploit those commonalities.

Hal Wilson: Let me shift the perspective to a more general view at the process level. At Litton PRC, we measured our results and we started to look at what went well and what didn't. And we found that for many of the problems that were occurring in very complex implementations, although the problems were primarily software driven, they really stemmed from our having stopped short of a complete, system-wide view. And, from a software perspective, as we went across the boundaries and embraced more of the system-wide considerations—when we brought the systems engineering environment together with our software engineering environment—it became obvious that we needed to have that wider perspective. Mature organizations that measure and verify what's going on and how to improve things find that the stovepiped view of a pure software or pure systems engineering context is limiting. We found that we needed to bring the two together. And that's why we're so supportive of this whole CMMI activity.

PB: The key is that we are primarily in the systems development business, and thus we need integrated systems engineering tools, including process improvement support tools.

Adapting the CMMI framework to an organization's needs

BP: The A-Spec lists eight deliverables from the CMMI framework that include various combinations of staged and continuous capability models (CMs), with and without integrated product and process development (IPPD). How would an organization determine which of these combinations would best meet its needs?

JW: First and foremost, an organization needs to understand what its business is. When we talk about process improvement, we have to consider an organization's goals and business objectives. So for example, if an organization is involved only in software engineering, then the software stand-alone model might suffice. However, if an organization is involved in systems development, then it makes sense to adopt one of the integrated models.

HW: I think that most of our organizations are large and very diverse. There are elements within individual corporations that are going to have different needs. In fact, depending on the type of project your organization executes, you may need to tailor almost on a project basis. You may not have a large enough project to do a full IPPD environment; or you may have a software-only development project and you may want to tailor down your processes from your full corporate set. The CMMI would allow an element of a large organization to select a CMMI product that matches the organization's need. The product-selection technique would allow you to choose and get a product that will allow you to make sure that you're tailoring correctly—especially to make sure that what you chose as a project-specific single-discipline environment is compatible with a multi-discipline version that another part of the organization might have selected. The CMMI product suite would give you a verification of that. As Joan said, you need to know what you're doing, and what your business is, but at the same time, you might have variations within your business that are facilitated by the ability to select products within the CMMI.

Staged vs. continuous representations

BP: What about the choice between the staged and continuous representations?

JW: I think that there will be guidance offered to the community on how they might decide which representation to use. For example, if an organization is using a staged model, such as the SW-CMM, then it might consider using one of the integrated models that is a staged representation. From a cultural perspective, it might be easier to transition to an integrated model having a familiar architecture. The fact is that, as in all technology transition, we have to deal with the cultural issues, which as we all know are the most difficult ones. If an organization has not yet adopted any of the legacy discipline-specific models for process improvement, and is just beginninga process improvement program, then the continuous model may be a logical choice. In the longer term, I think the continuous model has a lot more adaptability, flexibility, and room for growth, and facilitates extending the scope of a process improvement program.

BP: What is behind the decision to preserve both the staged and continuous representations of process development? Wouldn't it be simpler to choose one or the other?

HW: I think there's a characteristic of staged and continuous that has a lot to do with knowing what you're getting into. The continuous model presumes that you know what would be the logical choices for your organization. One of the values of an early adoption of a staged model is the fact that it basically tells you how to get there, and shows you what steps to take. It's very hard for organizations that are just starting to see their way clear to make decisions on "Which of these practices should we do first, how do we approach these, and what level should we be at in each one?" It does require a mature knowledge of what your business practices are, and what you need for your business. And I think it's very comforting for an inexperienced organization to judge risk, particularly for its management when they are making an investment without a lot of experience. As organizations mature, build a process-based culture, and learn howprocesses get adopted and adapted within their organizations, the tendency will be to move from a staged view to a more continuous one. Once the mechanisms become embedded in the organization, they are part of the

organization's culture. So I think that although the tendency has been that organizations new to the game are more comforted by a staged model, those of us who have been at it a while would say that the continuous model has more validity in the long term.

PB: I believe many engineering professional representatives of the medium and large systems development companies, who are familiar with multi-level staged models, see the advantages of continuous representation models. These engineers fully understand the implications of sequenced process improvement and have the experience and knowledge to make the appropriate choices as to what to improve and in what order, consistent with their business objectives. I think another part of the answer to this question is that although it might have been simpler to choose one or the other representation for the CMMI framework, it wouldn't necessarily have been the better or more effective choice. And I think that's where we've had a lot of discussion, a lot of different points of view, to suggest that we should implement and support both. And even though it's more difficult, we would have a lot more to offer in the long run. I think it's the right thing to do.

HW: I think you're right, Phil. And while it would be simpler to choose one or the other, we risk disenfranchising a portion of the industry if we do. That would have a much greater impact than to take the effort to do both.

PB: I think the other point, in case it's not obvious, is that we're bringing together two models, systems engineering and software engineering, which exist in the two different representations, continuous (systems engineering) and staged (software engineering).

Investments in previous CMMs

BP: A major concern of users is likely to be the protection of legacy investments in implementing previous CMMs. How is the CMMI effort responding to these concerns?

HW: I'm not even sure that's the right question in the sense of legacy. I know everybody's concerned with preserving their legacy, but I don't think that means restricting change. At Litton PRC we've been bringing together all of our software and systems engineering into a single integrated set over the last two and a half or so years. We predated the activity of the CMMI, and naturally made some missteps along the way. But the issue, I think, is not so much impacting or protecting the legacy, but rather improving the legacy. And when you begin to look at the process of continuous improvement, you know that some adjustments will be made to gain improvement. Starting as we did with the SW-CMM, which was a staged model, and integrating with it the continuous aspects of the systems engineering model, we had to make choices individually on the various practices. We had to decide how to make that come together just as the product development team (PDT) is now doing on the CMMI effort. Unfortunately, we had to do that without the help of the rest of the industry. So the CMMI effort is really trying to bring that together for organizationsto keep the impact of having to make those decisions to a minimum—so when the PDT does make those decisions, companies won't be at odds with the rest of the industry. Companies will have a way to validate that they are indeed doing the correct things. The reason it takes so long, when you're doing it from your own perspective, is that you're really trying to determine where this will be in the future. When no one else is giving you input, it's very difficult to do it by yourself. That's the value of a technical legacy, because if you're going to improve your environment and your current disciplines, you're going to have to make some choices. If you're not confident that the choices you're making are in the general direction that the industry is going, you may be in a backwater before you know it. So the CMMI is actually the way to protect your legacy and make sure that your organization will proceed the way that the industry is going.

MZ: We've heard this question a lot, and I still struggle with it. You have to be careful about your definition of "protecting legacy." If your definition is that what you've done in the past is going to be just as applicable in the future, that there's not going to be any change, then the answer is, we're not protecting the legacy, because there *is* going to be change. Now what we've tried to do is to take into consideration what changes are going to occur, and make sure that the changes are worthwhile. We've

been very clear in saying that when we come out with the software version of the CMM, it will not be identical to Version 2.0 that was in the draft stage. We are taking steps to provide the audit trail we've taken from the old Version 2.0 to the version where we eventually wind up. But, if protecting legacy means not making any changes, then we're not going to protect legacy. If you're going to improve, you're going to make changes.

JW: I think one of the key enablers to protecting legacy investments is the decision that we made to provide both the continuous and staged model representations. I certainly agree with the point that, if an organization is currently using the SW-CMM or one of the systems engineering models, there will likely be changes involved when a CMMI model is adopted. The changes may be due to additional or expanded practices in the CMMI model. The community has come to expect that there will be evolutionary changes and model improvements, and expects explanations and mappings to trace what has changed over the previous versions. You will find the same representations or architectures used in the CMMI models that were used in the CMMI source models (the SW-CMM, SECM, and IPD-CMM). Thus, organizations will be able to adopt the CMMI models in the representation (staged or continuous) they're using. This will allow legacy investment to be preserved. Over time, when an organization is ready to transition from whatever representation it's using to another, benchmarking can be used to facilitate the transition. We've said that the CMMI assessment method will provide consistent results across the two representations. So, if you're currently using a staged representation, and you conduct an appraisal, you could conduct another appraisal using a continuous model and compare the results. So, as you transition from one representation to another, the impact, if any, will be very clear. An important point to note is that the input models to the CMMI effort are the existing legacy models contained in the source documents of the A-Spec. Had the CMMI project started with a clean sheet of paper, there would have been no notion of preserving legacy investment in process improvement. I think our A-Spec approach requiring use of legacy models is another indication of what we're trying to do to preserve legacy investment.

HW: I think that's an excellent point. It's the assumption that we've made through all of this. When we speak to the legacy, my concept of legacy is really the organization's processes and practices that they have built their organization and their maturity upon. That's the legacy that you consistently and constantly improve. So, in that sense, you expect to change and migrate as you go forward. But you don't want to throw everything away. I think that's been the underlying issue in protection—we start with things that people are already doing, that have been well established and that they're comfortable with, and that the industry itself has formed a consensus about. And that's the protection. The improvement is natural, and that's where I think the distinction has been made between what you start with as a legacy and what you would like to retain in terms of concept, procedure, and discipline—your organizational heritage—and then move forward with that. The big concern that everyone had was whether the CMMI would diverge completely from where they were, and I think the answer to that is "no"—it's going to start with where they were and move forward.

BP: What about legacy investments in training and tools?

HW: I think if you look at the way organizations incorporate practice and extend maturity ... as they move from levels of maturity forward, they modify their internal training materials. They don't necessarily use a standard set; they improve. And when you look at what the CMMI is doing, particularly in the core areas, they're really bringing together separate training elements that might have to be trained from a different perspective into one cohesive set. As you move forward, you pick a consolidated systems and software engineering approach. If you pull one of those products from the CMMI, whether it be staged or continuous, you will have a consolidation of the essential elements of both, rather than having two totally separate training activities, as you would today. And I think if anything, it should minimize the impact. If you are truly going forward and incorporating the two disciplines, you should see the benefit immediately. If you're going to stay within just one discipline, then you should not see a significant difference. But, in either case, you will get the essential elements. And I think if there is an impact on training, it will be on the

organizations themselves. They will have to decide how they take and improve their internal training processes to bring the software-specific elements and move them slightly into the background, and to move the core elements forward within the total view. That, I think, is a tremendous value improvement, rather than a detriment. But the fact is, you still have to go through it. There is going to be some cost to move forward, even in going from one level to the next in the current model.

JW: I think we have to keep our eye on the return on investment from a longer term perspective. So, yes, there may be some impacts near term, to consolidate training, to integrate tools, and possibly to make some changes to tools if in fact they don't integrate. But the real objective is to realize long-term savings. To achieve the return on investment, which we expect will be achieved, we may need to invest in the near term. However, we'd expect to see tremendous payoff in the longer term.

CMMs for other functional disciplines

BP: What are some other disciplines not currently incorporated into the CMMI framework that you anticipate being incorporated in the future?

PB: As you know, what we're trying to do is develop a framework and an architecture that allows us to add additional development-related and enterprise-related disciplines. A couple have been mentioned in some of our forums. The ones that we're actually going to add will come from the real development users and the technology and operational needs, as they advance and grow. But I think potentially, some of them might be artificial intelligence or the expert systems engineering area, complex hardware development activities, security engineering/trusted systems engineering, safety-critical systems, where extra processes have to be overlaid on the baseline processes... All of those relate to technical engineering activities. But if you go to a broader development perspective, we could even think about broader coverage of the basic program management activities. It's too early to say now how this is all going to come together. It's going to depend on what the industry and what the actual developers believe as they begin the process.

JW: Another area for future CMMI model expansion that has been suggested by the user community is systems acquisition.

PB: And in that sense, if you look at the full life cycle of engineering systems, you might say thatmaintenance and supportability, for example, might be appropriate disciplines.

Relationship with international standards

BP: I note that the A-Spec includes a requirement that the CMMI product suite be compatible with ISO 15504. How is the relationship with international standards evolving?

MZ: There's certainly a sensitivity to make sure that what we do here takes into consideration what's happening in the international arena. We don't want to develop something that isolates the companies in the U.S. from the international marketplace. A number of folks on the team—on the Steering Group or the product development teams or stakeholder reviewers—are involved at the international level with ISO/IEC documents. I haven't counted them, but I know there's a fair number involved with the U.S. TAG (Technical Advisory Group), to JTC1/SC7 which is the international group under ISO/IEC that's responsible for 15504, 12207, 15288 and other related documents. So I think that the folks who are working in both arenas are involved, which gives the CMMI effort first-hand knowledge of what's happening and a view of what the draft documents are like. That, I think, is one of the main reasons for having it in the spec—so that we make sure we don't ignore what's happening on the international arena. We want to make sure that we don't put our industry in a position of being unable to compete in a world marketplace.

HW: I think that sums up the whole issue. One of the things we want to make sure of is that the term "evolving" is really the key term. Even with a statement of compatibility, we have to recognize that over time, even in a relatively short time, each of these activities—on the international level, the national level, and the CMMI level—will be operating somewhat independently, and probably out of sync. You can

expect that something will occur in one element that will affect the other, and they will adapt. And what you eventually would like to do—and what the intent of the A-Spec is—is to make sure that international standards are considered. What we don't want is that, once the stake is driven into the ground, it isn't allowed to move simply because we placed it there. It should move to the logical place that industry consensus will take it. It's more important to recognize how to remain competitive, how to make sure you accommodate the things in our organizations that drive how we implement and develop. This is especially important because most of our organizations are international.

MZ: Hal brought up a very important point. Most of the things we're talking about are evolving. 15504 is not an international standard; it's a technical report that has not yet gotten to the status of an international standard. The systems-level life-cycle model, 15288, is still at the working draft level. That document hasn't been circulated outside of the working group yet for review and comment. So the only document we have to look at in terms of a full standard in the international arena is 12207. And there's inconsistency between the international documents. So the challenge is not only to make sure we're aligned with the international standards, but also to resolve the conflicts that exist within 12207 and 15504, for example. It's not a very easy thing to deal with.

Input from the user community

BP: What efforts are you making to keep the user community informed of the continuing evolution of the work? How are you gathering stakeholder review comments and adjustments?

PB: What we've done is gone out in a number of forums and explained, at the time, what the project was all about and given our status. We've put a number of those briefings and a set offrequently asked questions (FAQs) on the SEI Web site to inform the community about what is going on with the CMMI effort. We've established a stakeholder review group, which is a group of folks who are going to review as we go through the development effort. Members of the stakeholder review group represent and spread the word to their very large organizations. So we're anticipating a lot of input, and that's another way of informing the community of what's going on with the project.

BP: How about commercial industry? Any special efforts going on to involve commercial industry?

PB: The commercial industry raised this concern in Chicago a while back, and we took names of those who were really interested. Our lead industry person, Bob Rassa, then went through some effort to reach out to these folks and invite them to participate, and we did get some participation in the stakeholder review group.

HW: Bob Rassa also asked the commercial arm of the Electronic Industries Alliance (EIA) to go forward and see if there was any interest in commercial industry to participate. And I think the initial reaction was that those organizations that were heavily involved and felt that it was in their best interest have chosen to do so. In fact, there have been several offers for individuals to participate on the product development teams.

JW: I think that one of the things we did to ensure that we did reach out to the commercial community was take a hard look at the distribution channels we were using to disseminate information on CMMI. Specifically, we talked with the SEI about the distribution lists they had used in the past for their various correspondence groups and advisory boards, and we included those people in the distribution for CMMI information. In particular, we included them in the invitation to participate in the CMMI effort by providing product development team members. We realized that we needed to expand our communication channels and our distribution list. Hopefully, with the additional outreach, everyone who has been involved with capability models in the past is now aware of the CMMI Project. And, we would like to think that everyone is tuned in to the CMMI page on the SEI Web site, so they can keep up to date in the future on what's happening with the project.

Role of the Steering Group

BP: What role does the Steering Group play in guiding this effort?

PB: The Steering Group has an important role in this development effort. For example, we decided to take a systems engineering approach in developing the CMMI product suite. As a result, we have written the functional/performance requirements in the form of a formal specification (A-Spec). The Steering Group is also responsible for configuration control of the A-Spec and the top-level design products as they evolve. We have a role between the sponsor and the users, and we're representing the users in tracking progress and approving products, so we're providing an insight/oversight kind of steering function to the product development teams. We're resolving issues that come up, and right now, we're in the midst of planning transition—how we are going to sustain, maintain, and support these product-suite-based products once they're developed. And we're making efforts to disseminate information about these developments as they occur.

MZ: From the very beginning, this was defined as a collaborative effort between government, industry, and the SEI. The Steering Group exists as a forum for having that collaborative effort to make sure that the voices of the services, DoD, and industry are heard and help drive and direct this effort.

HW: If you look at the way that development organizations are set up, you find that most of those organizations have separate elements that are concerned with each of the disciplines, and these separate elements have perspectives that are self sustaining. The organizations that acquire and implement systems or utilize systems within the DoD and within industry have a lot of the same characteristics. So it's pretty obvious when you bring a complex set of environments and put them all together that there has to be some coordinating and even controlling element that keeps the industry and government needs and constraints in perspective. You could create tremendous impacts if you just went willy-nilly. As Mike said, the Steering Group provides a forum and a mechanism for not only bringing that discussion together, but also bringing a consensus on what is really necessary to meet the needs of all the constituents. We haven't really had a means to do that in the past, and part of the reason there were so many CMMs was that there wasn't a means to consider the needs of all the constituents in the past. So if we had not brought the Steering Group into existence, the result would be far different.

PB: We have both industry and government representation on the Steering Group, along with systems engineering and software engineering perspectives. So in this sense, we have brought together these backgrounds and representations, which I feel is very important. And from all of the members of both industry and government, we reach into a number of standards bodies and associations within the industry that are oriented toward particular activities, and if we weren't able to draw on this diversity, we wouldn't be able to bring together a consensus model.

Effect of CMMI on assessments

BP: In what ways might the CMMI be beneficial in terms of how if will affect assessments?

HW: Industry and government organizations have performed assessments, either internally or externally, to benchmark where they are. Generally, you want to go out and get someone other than yourself to let you know if your self-assessment is valid. Our management certainly does—they don't believe that an organization claiming a particular level should be the one making that assessment. For a third-party assessment, the issue is what mechanism do you use? And I think that the way the CMMI is going is to bring together a unified assessment approach.

JW: That's an excellent point. I think the key is that the mechanism and underlying model an organization is using to assess itself internally, doing self-assessments, is consistent with the mechanism and model used for external benchmarking.

HW: Today it would be impossible for an organization to bring together software engineering, systems engineering, and IPD in a single assessment approach on their own. They would have no way of arbitrating the differences in the models to gain any

result that industry could validate. What the CMMI is going to provide is a single, unified, consensus approach. And that's valuable. That's a major cost savings. The difficulty in bringing together the different CMMs in an organization ahead of time is you're always forced to go back and look to see if you can still pass any one of these divergent CMMs or Software Capability Evaluations (SCEs) from a different perspective. The CMMI will eliminate that. That's one of the problems of being an early adopter—having to make sure that you're not jeopardizing your evaluation in any independent SCE against one CMM versus another. Because the characteristics are slightly different—in this case, they're somewhat divergent—one element gets more credence in one SCE than in another. The training of the individual and which SCE is being used, for which CMM, can greatly affect a good basic practice. The CMMI will eliminate a lot of that. This is one of the greatest advantages of a unified model.

JW: If the same model and the same appraisal method are used in both internal and external appraisals, then you can achieve comparable results. If you use different appraisal methods or different models, then comparison of results may be difficult. As long as everyone is aligned from a model and method perspective, then the outcomes of internal and external appraisals can synergistically support an organization's process improvement program.

HW: One of the inherent and underlying concerns about legacy has been in the rating area. Many people have expressed the concern of "How do we know if we'll keep the rating we have?" If you're only dealing with one of the CMMs now, particularly software, you'll have a rating based on a current model. If you continue to operate within a staged model, while there will be an improvement within the content of the software CM product of the CMMI, there would be some stability in that process, but there would be the same impact on your rating as if you were going to the next level of a release. Companies that are already moving to bring together software and systems engineering realize that the costs involved in trying to accommodate both models today are enormous. And that's really the issue. Most companies that are doing complex operations realize that they are doing systems engineering as well as software engineering, and they need to address both in order to be mature; so the movement toward a common assessment model is critical. Today you can't have a combined rating for software and systems engineering maturity. The CMMI is the only thing that's going to give you that.

Maintenance of CMMI products

BP: What are the plans for maintenance of the CMMI products? What guarantee is there that this will not be a one-time product?

HW: It's an issue that is being addressed by the Steering Group. We recognize that it's a Steering Group responsibility to define a process for maintenance and improvement of the CMMI. And that will be one of the things coming out of this in the future.

About the moderator

Bill Pollak is a senior writer/editor, member of the technical staff, and team leader of the Technical Communication team at the SEI. He is the editor and co-author of *A Practitioner's Handbook for Real-Time Analysis: Guide to Rate Monotonic Analysis for Real-Time Systems* (Kluwer Academic Publishers, 1993) and has written articles for the *Journal of the Association of Computing Machinery (ACM) Special Interest Group for Computer Documentation (SIGDOC)*, *CROSSTALK*, and *IEEE Computer*.

About the panelists

Philip S Babel is Technical Advisor for Embedded Computer Systems Software, Aeronautical Systems Center (ASC), Air Force Material Command (AFMC), Wright-Patterson Air Force Base Ohio. His responsibilities include technical leadership for the application of computer systems and software technology to aeronautical systems acquisition and development. He defines and establishes policies, processes, practices, and methods for the engineering of embedded computer systems software. He has a BS in electrical engineering from the University of Detroit and an MS in

computer and information systems from the Ohio State University.

Joan Weszka has over 25 years of experience in software and systems engineering, and program management of computer systems. At IBM, she held management and technical positions in systems development of commercial and government large-scale, real-time, systems, and was Process Program Manager at IBM Federal Systems Company Headquarters. She is currently Manager of Process and Program Performance at the Lockheed Martin Enterprise Information Systems' Software & Systems Resource Center, a service organization and source of expert resources to Lockheed Martin companies in areas of consultation, process improvement, training, and technology transition. Weszka is Chairperson of the Enterprise Process Improvement Collaboration Steering Group. She previously served as Chairperson of the SEI's SW-CMM Advisory Board and was a member of the SEI's Software Acquisition CMM Steering Group, the SCE Advisory Board and the CMM-Based Appraisal Advisory Board.

Hal Wilson has been designing and implementing computer information and computer-assisted communications systems for over 30 years. He holds the position of vice president and general manager, 'Net Solutions, the focal point for Internet/Intranet-related customer support activities within Litton PRC. Since joining PRC in 1984, Wilson has directed the design and implementation of two major systems integration programs and has directed the design, competitive selection, and development of two large system programs. He also created and led the Systems and Process Engineering organization that developed process and engineering policy and procedures for systems and software engineering within Litton PRC. Wilson is currently the chairman of the Systems Engineering Committee (G47) of the Electronic Industries Alliance, which is responsible for the creation of two new systems engineering standards, ANSI/EIA 632, Processes for Engineering a System, and EIA IS 731, Systems Engineering Capability Model. He also serves as the vice chair of the National Defense Industrial Association Systems Engineering Committee, chartered by the Systems Engineering Directorate of the Office of the Under Secretary of Defense for Acquisition and Technology. Wilson holds a Bachelor of Science degree in physics from St. John's University in New York. He also is a graduate of the Western Electric Graduate Engineering Education program. Wilson was awarded Federal 100 award by Federal Computer Week Magazine in 1993. He co-holds a patent for an Alarm Scanning Mechanism designed for the Washington Metro Communications Control System. The views expressed in this article are the author's only and do not represent directly or imply any official position or view of the Software Engineering Institute or Carnegie Mellon University. This article is intended to stimulate further discussion about this topic.

Michael G. Zsak, Jr. is a Systems Engineer for the Office of the Secretary of Defense (Acquisition & Technology). He is responsible for providing technical support to the Deputy Director, Systems Engineering and the Director, Test, Systems Engineering, and Evaluation. In addition to his normal duties, he serves as the DoD advisor to the International Symposium on Product Quality & Integrity (RAMS), the DoD representative to the U.S. Technical Advisory Group for ISO/IEC JTC 1/SC7, Chairman of the Reliability Analysis Center Steering Committee, and Vice Chair of the Society of Automotive Engineers International Division on Reliability, Maintainability, Logistics, and Supportability. He is also a guest lecturer at the University of Maryland and the Defense Systems Management College.

Contact Us
info@sei.cmu.edu
412-268-5800

Life in the Digital Subscriber Lane

NEWS AT SEI

Author

Scott R. Tilley (Florida Institute of Technoloy)

This article was originally published in News at SEI on: September 1, 1998

In the first *Net Effects* column, I talked about the era of net-centric computing (NCC) and its profound and far-reaching effects in the office and elsewhere. Recall that the underlying principle behind NCC is a distributed environment where applications and data are downloaded from network servers on an as-needed basis. NCC relies on portable applications than run on multiple architectures ("write once, run anywhere"), high bandwidth (for downloading applications on demand), and low-cost thin clients such as the network computer (NC), the NetPC, and Windows-based terminals (WBT).

However, there is much more to NCC than just thin clients. NCC is not a technology *per se*. Rather, it is a collection of technologies that, taken together, characterize this new paradigm. Table 1 illustrates some of the separate technologies (and specific instances) that together constitute net-centric computing. As with most things in computing, terminology related to NCC is full of acronyms. From Table 1, Orb refers to a CORBA (Common Object Request Broker Architecture) server, DOT stands for Distributed Object Technology, CBS means Component-Based System, XML is the eXtensible Markup Language, and TPS is a Transaction-Processing System. There are of course other technologies involved in net-centric computing; this is just a representative sample.

Technology	***client***	***server***	***object***	***data***	***infrastructure***	***control***
Instance	fat, thin, lean	Web, Orb	DOT, CBS	XML	Ne	

Table 1: NCC Technologies

The first column focused on the client aspects of NCC, in particular on thin clients. This column focuses on some of the infrastructure aspects of NCC, in particular on my experiences with one type of high-speed Net access technology that is becoming widely available at relatively low cost: digital subscriber lines (DSL). In early June, I had US

West's "Megaline" DSL service installed in my home office. To paraphrase Austin Powers, "*It's fast baby, yeah!*"

High-speed Net access

There are now several high-speed Net access options. Table 2 summarizes some of the more common methods available to consumers. The most widely used technique for Net access is still analog modems running at either 28.8K or 33.6K. (Note: Throughout this article, when used in the context of Net speeds, the suffix 'K' refers to thousands of bits per second and the suffix 'M' refers to millions of bits per second.) The more recent deployment of 56K modems has given users the capability to connect at speeds comparable to single-band ISDN (Integrated Services Digital Network). Note that 56K modems operate in an *asymmetrical* fashion: the download speeds can (theoretically) reach 56K, but the upload speeds are limited to 33.6K. Asymmetric bandwidth is common for consumer-oriented high-speed Net access technologies. This is due primarily to the limitations of the so-called "last mile" of the network, from the external infrastructure to inside the home. Asymmetric access is usually sufficient, given the nature of Net use. Most information is coming downstream from the Net (or the Web) to the client; comparatively little is sent back upstream, except with bandwidth-intensive applications such as video conferencing.

Net Connection	***Upload Speed***	***Download Speed***
Dial-up modem	33.6K	56K
Bonded modems	64K	128K
ISDN	64K/128K	64K/128K
Satellite dish	33.6K	400K-10M
Power lines	1M	1M
DSL	256K-1M	256K-7M
T-1	1.5M	1.5M
Cable modem	1M-2M	1M-10M

Table 2: High-Speed Net Access Options

Modem bonding is an esoteric development that allows two modems to be coupled to operate as one virtual modem. This setup requires two telephone lines and an Internet service provider (ISP) that supports this type of multi-link connection. When it works, it offers speeds comparable to dual-band ISDN (128K). ISDN itself has a checkered

history of high cost and notoriously difficult setup. Although it has been available for some time, it is being supplanted by newer technologies.

Some of the newer technologies are satellites, digital subscriber lines, and cable modems. These technologies provide connection speeds that are equal to or better than the traditional speeds of dedicated and expensive T-1 lines. Satellite hookups operate in an asymmetric fashion, using the satellite dish to download data at speeds ranging from 400K to 10M. Data uploads are still done using a regular analog modem. The wide discrepancy in speed for satellites reflects the different types of technology used. Some satellite systems receive information from space-based satellites in stationary orbit. Others receive signals from ground-based transmitters. Although the ground-based setups can offer much higher connection speeds, the receiver needs to be in a line-of-sight with the transmitter. For flat areas such as Phoenix, this works very well. For hilly areas such as Pittsburgh, the use is more limited. For most people, affordable and high-speed two-way satellite communication is still a few years away. Satellite constellations like Iridium or Teledisic may change that.

Recently, Nortel and Norweb collaborated to offer 1M Net connections over standard power lines. This means one could literally plug the computer into the wall socket and voila!—high-speed network access without any extra wiring. For the moment, this technology is limited to trials in the UK, where transformer setups are different from those in North America. However, it is expected that the technology will become available here as it matures. The allure of powerline-based networking to homes and small offices is obvious: no changes in the infrastructure are required to go online. Even now, you can buy smaller versions of powerline-based networks for the SOHO (single office, home office) use, allowing computers and printers to be networked using small adapters plugged into wall sockets.

For higher speed access, cable modems are becoming more widespread. They currently offer up to 10M connection speeds downstream and up to 2M upstream. They have the advantage of using the existing wiring from television cable companies, thereby reducing the infrastructure requirements. This is one of the main reasons that AT&T recently purchased the cable giant TCI. The disadvantage of cable modems is that the connection is shared, or pooled, among users in the neighborhood. Although there may be 10M of bandwidth available from the cable company, if 10 people are all using the service at the same time, each will receive about 10% of that bandwidth, reducing throughput to 1M—still quite fast, but not as fast as one might like.

Digital subscriber lines

The main competition to the cable modems offered by the cable companies is digital subscriber line (DSL) technology, offered by the local telephone companies. The appeal of DSL is that it offers users high bandwidth (up to 7M) without requiring any new wiring; it uses the existing copper-based infrastructure, also known as POTS (Plain Old Telephone Service). In fact, the same line can be used for simultaneous Net access and voice or fax communication. This is because the DSL system uses a different part of the telephone line's frequency for different types of data. Unlike cable modems, the DSL line is not shared with other users; you have all the bandwidth to yourself.

DSL is also asymmetric, moving data at up to 1.5M upstream and up to 7M downstream. For this reason, DSL is sometimes called ADSL. In fact, there are several variants of DSL, so it is also known as *x*DSL. The biggest stumbling block to widespread adoption of DSL is (again) the existing infrastructure. Because DSL uses the existing copper telephone lines for the last mile, special equipment must be installed between the phone company's central office and local neighborhoods. This equipment creates “access loops” for customers. Unfortunately, the loop's length is limited to a couple of miles. This means a significant portion of U.S. phone customers live too far away from their local phone offices to get DSL.

My experience with DSL

Unlike ISDN, which requires extensive knowledge on the part of the consumer and the installer, installation of DSL is relatively simple. For me, it involved a short visit by a representative from US West's Megaline service. He installed a “modem” by NetSpeed (a subsidiary of Cisco Systems) that connects to the regular telephone line. I already had the other software they offer, such as an Internet browser, installed. So all I needed was some standard ISP information, such as the name of the mail server, and that was it. The entire installation took about 15 minutes.

The NetSpeed device is a SpeedRunner 202. It's about the size of a hardcover book, turned on its end. Calling it a modem is really a misnomer; it's actually an Ethernet-to-ADSL router. It uses the ATM (asynchronous transfer mode) message-packaging format over an ADSL physical layer to support PPP (point-to-point protocol) networking. I

already had a second telephone line that I was using for 56K access, so I just switched that modem line into the back of the SpeedRunner, connecting the device to US West's WAN. The SpeedRunner also has a 10/100Base-T interface, into which I plugged my computer's Ethernet card. With two simple connections, I was online with DSL.

I should point out that DSL is just the physical layer of the Net connection. You still need an ISP for the usual services, such as domain name resolution, email delivery, and so on. Moreover, with DSL you need an ISP that supports DSL. This is similar to finding an ISP that supports your type of 56K modem access, x2 or K56Flex. In my case, I chose uswest.net, since it was the easiest combination and they provided service in my area.

The cost for all the new technology is relatively small. Installation was $110. The cost of the DSL line is $40 per month. This includes the concurrent use of the line as a regular telephone or fax line. The ISP cost is extra, currently $19.95 a month for unlimited access. Thus, for about $60 a month I have high-speed Net access and simultaneous use of a second phone line. So far, I've been very pleased with the performance. I have the lowest DSL speed offered, which is nominally 256K both upstream and downstream. However, the SpeedRunner has a diagnostic mode that lets you measure actual throughput, and I seem to be connected at about 640K downstream and 256K upstream.

From the end-user perspective, I can say that using this DSL connection feels almost the same as sitting on a 10M corporate local area network (LAN). Web pages load almost instantaneously, files download very rapidly using ftp, and large email attachments are no longer a problem. A side benefit of the SpeedRunner router is that it can be plugged into a hub, so I could share the Net connection with other computers on my office LAN if I so chose.

Implications of wide-spread deployment of DSL

If the use of DSL for SOHO access to the Net becomes more common, the implications may be significant. (The same could be said for other high-speed Net access options, such as cable modems.) DSL can support a variety of high-bandwidth applications, such as Web access, telecommuting, virtual private networking, and streaming multimedia content for on-demand video or teleconferencing. These services were either not possible to support or were ineffectively supported by conventional dial-up data delivery technologies such as 33.6K analog modems. Since DSL is based on existing standards (PPP, ATM, and so on), a competitive marketplace is being created to offer a variety of networking solutions. Besides offering high-speed Net access, the “always on” feature is very attractive: there is no need to dial to connect to your ISP; your Net connection is always ready.

For office use, DSL lends itself to branch office connectivity, effectively replacing expensive leased lines. As mentioned before, most business PC applications (e.g., file access, email, terminal emulation, Web browsing) perform asymmetric communication, making DSL a suitable technology to connect a SOHO to the enterprise. With more organizations exploring the benefits of telecommuting, the importance of high-speed Net access technologies like DSL will increase.

Although there are significant security issues that need to be addressed, technologies such as DSL can provide the infrastructure for virtual private networks (VPN), also known as “Extranets.” A VPN is a secure extension of an Intranet to existing global network backbones, such as the Internet. Nodes on the Extranet can use DSL to connect at high speeds through corporate firewalls using secured tunneling protocols such as PPTP (point-to-point tunneling protocol). Eliminating geographical distances between points on the network, and hence fostering collaboration between people, is just one way DSL can be used to improve the net-centric computing experience.

About this column

The focus of the *Net Effects* column is the impact of net-centric computing on a wide variety of issues, including computer science, information technology (IT), and software engineering.

About the author

Scott Tilley is a visiting scientist with the Software Engineering Institute at Carnegie Mellon University, an assistant professor in the Department of Computer Science at the University of California, Riverside, and principal with *S.R. Tilley & Associates*, a strategic and tactical information technology consulting boutique.

The views expressed in this article are the author's only and do not represent directly or imply any official position or view of the Software Engineering Institute or Carnegie Mellon University. This article is intended to stimulate further discussion about this

topic.

For more information

Contact Us

info@sei.cmu.edu

412-268-5800

Your Date or Mine?

NEWS AT SEI

Author

Watts S. Humphrey

This library item is related to the following area(s) of work:

Process Improvement

This article was originally published in News at SEI on: September 1, 1998

Congratulations, you've just been promoted. You get to lead the new project we just won. You will have six engineers, but two of them are half time for a month or two. You can hire four more. The delivery date is nine months. What do you say? Most engineers would say, "Gee thanks boss, I've always wanted to run a project and this sounds like a great opportunity. I'll give it my best shot, but the date looks awfully tight." If that's your answer, you lose!

Who owns the nine-month date? When your boss offered you the promotion, whose date was the nine months? It was the boss's date. But, when you said, "Boy that's a tough date, I'll do my best to meet it," whose date was it then? Yours! And don't ever forget it! You have just bought the ranch. Even though you had no intention of doing so, and you didn't even have time to think about it, bang, it hit you out of the blue. And there you are, the proud owner of a budding disaster.

So what else could you do? It turns out there is plenty you could do.

Getting Into Trouble

Projects usually get in trouble at the very beginning. They start with impossible dates, and nobody has time to think, let alone do creative or quality work. All that counts is getting into test, and the rush to test invariably produces a hoked-up product, a poor quality program, a late delivery, an unhappy customer, and disgruntled management.

While the promotion looks good at first glance, it is only good news if you handle it right. To have any chance of a successful project, there are things you must do right now. Before we talk about what to do, however, let's discuss the causes and

Related Links

News

SEI to Co-Sponsor 26th Annual IEEE Software Technology Conference

SEI Cosponsors Agile for Government Summit

Training

See more related courses »

Events

Team Software Process (TSP) Symposium 2014

Nov 3 - 6

consequences of this all-too-common situation.

Pressure

The only way to manage development work is with pressure. Managers know that relaxed projects rarely succeed. Projects can get into trouble by endlessly studying alternatives, not making decisions, or loading up a product with nice features. These all seem like good ideas at the time, but without pressure months can go by and nobody notices.

So expect pressure. Managers know they must push, and you can expect them to keep pushing, at least if they are awake and doing their jobs. Their only questions are

- What dates will they push for?
- How hard should they push?

The Problems With Pressure

Schedule pressure causes all kinds of counterproductive behavior. People don't plan their work, they rush through their designs, and they don't review their products. The big push is to demonstrate progress to management, and the only thing management recognizes as progress is getting into test.

Unfortunately, few managers really understand that this is the worst possible thing they could push for. Most software engineers know that racing to throw a product into test is a mistake but they don't know how to fight the pressure. Then they feel compelled to rush through requirements and design and to skip everything else but code and test. When they do, they know what will happen. And of course it does.

While you can always say management was unreasonable, you will be responsible for being late and producing a poor-quality product. Everyone can easily blame somebody else, but do you really want to spend your life this way?

I Told You So

You know intuitively when a schedule is too aggressive. You have a queasy feeling in the pit of your stomach. You can sense the dates are wildly optimistic, and you know this is a disaster just waiting to happen. So you tell the manager, "That schedule looks awfully tight to me," and the manager responds, "But that is the date in the contract, or that's the date the customer demands, or that's the date the boss committed." So you say, "Ok boss, if you say so, I'll try but don't be surprised if it takes longer."

Now that you told the boss, if there are problems, you can always say I told you so. Unfortunately, that won't help. The minute you walk out of the room, it is your date. You may have been worried, but you took the job didn't you? Why did you take the job if you didn't think you could do it? If you don't settle the issue right now, you will be the goat, no matter what you say about the date.

Negotiating Schedules

So you must take a stand. Most engineers are so focused on the job that they don't think about what the manager is saying. When managers say the delivery date is nine months, they are making a bid. And you bought it without a counter offer. You'd never buy a house or a car or a boat this way. You'd debate the number.

Think about it this way. Management has just said, "We have this key project, and the best date we think you can make is nine months." If you don't counter with another date, they will hold you to nine months. Unfortunately, if you just guess a later date, they will ask you why. And if you don't have a good answer, they will either ignore your date or get somebody who won't argue.

Management doesn't know how long the job will take, and neither do you. If you knew, and if you could convince them you knew, they would accept your date. If the project really will take 12 months, the last thing most managers want is a commitment to deliver in 9 months. You work for them, and they will also be held accountable for your schedule. They could easily lose their jobs if your project fails, and all you would lose is a chance to have another disaster. That would probably be a relief, at least after this project.

Handling Pressure

You must start by convincing management that you know how long the project will take. To do this, however, you must know yourself. This, it turns out, is not very difficult. It takes time and some hard work, but when engineers make careful estimates, and when they use historical data to make these estimates, they are generally pretty accurate.

The way you determine the date is to make a plan, and to make a very detailed plan. Since this can take a lot of work, and since you want your team to be part of this planning process, you need to get your new team to help you. This actually is the best possible approach. It will not only produce the best plan, but the team will then be in a far better position to do the work. They will also be committed to the schedule.

Then, when you have the plan, go back to management and tell them what the date really is. When they argue with you, as they will, take them through your plan. Show them as much detail as they will sit still for. Walk them through the *numbers*, and the *task lists*, and the *historical data* you used for comparison. Talk about *product sizes* and *productivity rates*. Show them enough to really convince them that you know what you are talking about.

Be Flexible But Firm

Once you have made the sale, and management accepts your plan, stop selling and move on to the next subject. They will probably want to talk about alternatives. What would the date be with more staff, or reduced requirements, or a phased-version delivery plan? They may want you to present the plan to higher management, or to the customer. In fact, you will probably have to walk more people through the plan, so keep it dusted off, and make sure your backup is solid. Expect people to find any chinks or inconsistencies. Remember though, these are estimates. So tell them what you think and why, but remember that no one knows as well as you do how long the job will take.

If anyone can convince you that your estimates are off, be willing to make adjustments. As long as they have actual historical data to back up their opinions, and as long as these data are relevant to your project, consider the new facts. Under no conditions, however, make any such decisions on the spot. Any schedule change requires careful study and team agreement. So don't change your estimates without data and the time to review them with your team. Remember, almost all initial plans are tight, so don't cut your schedule without a very good story that the team agrees with.

Answering Management

So, when management says the date is nine months, the way to answer is to say, "That looks like a great opportunity boss. Let me get the team together and we'll take a look. We'll make the best plan we can, and be back to you in a couple of days."

If you think the schedule will be longer than management wants, don't argue about it right now. You don't have the ammunition to win that argument, and all that would do is convince management that you have a bad attitude. If they think you are out to prove their date is wrong, they won't let you make the plan. Remember, they don't believe you can give them a good date. After all, nobody has made good schedules before, so why should you be first?

So start with a positive attitude, and really drive for a better date. In fact I once had a team come back with a five-week better schedule than management had asked for, and they're still holding to their plan. So give it an honest try, but then get the data to defend it.

How Does This Work

I have coached many development teams on how to do this, and it always works. Of course, we have started by softening up management, and we have been there to help at the beginning. But teams are surprised at how well this approach works and, after a good start, they can usually continue working this way. Management also quickly discovers that informed and committed teams do vastly better work. While most plans come in with longer dates than management wanted and management always asks lots

of tough questions, when teams have good plans they can defend them. And when they defend their plans, they always convince management. Best of all, they end up working to their schedule not management's.

So addressing the schedule problems up front really pays off, both for the engineers and for management. While it takes management a few days to get over the shock, they will end up with a software team that knows what they are doing. These teams can report their progress against realistic plans, deliver on the agreed schedules, and produce fine products.

The Next Steps

So you need to know how to make a plan, how to present this plan to management, and then how to defend the plan when it is attacked. Then, once you have done all this, all you have to do is develop the product. But at least you will have a date that you and your team agree you can meet. And, most important, you will have a well-thought-out plan to guide the work.

While these methods are not difficult, they are not obvious. If you need help in how to do this, one place to look is at the various Personal Software Process (PSP) references that explain how to make individual plans.(The PSP references are my two books *A Discipline for Software Engineering* and *Introduction to the Personal Software Process*, both published by Addison-Wesley. Additional references are in the February, March, and April 1998 issues of *Crosstalk*.)

The SEI teaches PSP courses, and there are a growing number of university and commercial courses available. We are also introducing the Team Software Process (TSP) which shows teams of PSP-trained engineers how to handle this planning and commitment process. TSP introduction also walks teams through a launch process that produces their detailed plans and negotiates their schedules with management.

Final Comments

While this all sounds logical and simple, everybody's situation is different. If you remember two basic principles, however, you should be able to handle almost any case. First, you are paid to do what management wants you to do, so don't refuse a direct order. Second, always be willing to make a "best effort" but don't commit to a date without a plan. If management appears totally unreasonable, however, read up on negotiating strategies before you get in over your head. (Probably the best reference on this subject is *Getting to Yes*, by Roger Fisher and William Ury, Houghton Mifflin, 1981. I also summarize some key points about negotiating programming issues in Chapter 12 (Power and Politics) of my book, *Managing Technical People, Innovation, Teamwork, and the Software Process*, Addison-Wesley, 1997.)

Most managers will be reasonable and respect your desire to make a plan, but occasionally one won't listen. While he or she could be totally unreasonable, it is more likely that higher level managers are applying heavy pressure that your manager is unwilling to buck. You should certainly try to turn such situations around, but that is normally very difficult. If you can't make headway pretty soon, get out from under your gutless manager as quickly as you can.

Acknowledgements

In writing papers and columns, I make a practice of asking associates to review early drafts. For this column, I particularly appreciate the helpful suggestions from Linda Gates, Alan Koch, Warren Morrison, Mark Paulk, and Bill Peterson.

In Closing, An Invitation to Readers

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

Thanks for your attention and please stay tuned in.

Watts S. Humphrey

watts@sei.cmu.edu

About the Author

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

The views expressed in this article are the author's only and do not represent directly or imply any official position or view of the Software Engineering Institute or Carnegie Mellon University. This article is intended to stimulate further discussion about this topic.

Security of the Internet

NEWS AT SEI

Authors

James Ellis

Howard F. Lipson

Thomas A. Longstaff

Linda Pesante

Derek Simmel

This article was originally published in News at SEI on: December 1, 1998

This article was first published in *The Froehlich/Kent Encyclopedia of Telecommunications vol. 15*, pp. 231-255, New York: Marcel Dekker, 1997.

Overview of Internet security

As of 1996, the Internet connected an estimated 13 million computers in 195 countries on every continent, even Antarctica. The Internet is not a single network, but a worldwide collection of loosely connected networks that are accessible by individual computer hosts in a variety of ways, including gateways, routers, dial-up connections, and Internet service providers. The Internet is easily accessible to anyone with a computer and a network connection. Individuals and organizations worldwide can reach any point on the network without regard to national or geographic boundaries or time of day.

However, along with the convenience and easy access to information come new risks. Among them are the risks that valuable information will be lost, stolen, corrupted, or misused and that the computer systems will be corrupted. If information is recorded electronically and is available on networked computers, it is more vulnerable than if the same information is printed on paper and locked in a file cabinet. Intruders do not need to enter an office or home, and may not even be in the same country. They can steal or tamper with information without touching a piece of paper or a photocopier. They can create new electronic files, run their own programs, and hide evidence of their unauthorized activity.

Basic security concepts

Three basic security concepts important to information on the Internet are confidentiality, integrity, and availability. Concepts relating to the people who use that information are authentication, authorization, and nonrepudiation.

When information is read or copied by someone not authorized to do so, the result is known as *loss of confidentiality*. For some types of information, confidentiality is a very important attribute. Examples include research data, medical and insurance records, new product specifications, and corporate investment strategies. In some locations, there may be a legal obligation to protect the privacy of individuals. This is particularly true for banks and loan companies; debt collectors; businesses that extend credit to their customers or issue credit cards; hospitals, doctors' offices, and medical testing laboratories; individuals or agencies that offer services such as psychological counseling or drug treatment; and agencies that collect taxes.

Information can be corrupted when it is available on an insecure network. When information is modified in unexpected ways, the result is known as *loss of integrity*. This means that unauthorized changes are made to information, whether by human error or intentional tampering. Integrity is particularly important for critical safety and financial data used for activities such as electronic funds transfers, air traffic control, and financial accounting.

Information can be erased or become inaccessible, resulting in *loss of availability*. This means that people who are authorized to get information cannot get what they need.

Availability is often the most important attribute in service-oriented businesses that depend on information (e.g., airline schedules and online inventory systems). Availability of the network itself is important to anyone whose business or education relies on a network connection. When a user cannot get access to the network or specific services provided on the network, they experience a *denial of service*.

To make information available to those who need it and who can be trusted with it, organizations use authentication and authorization. *Authentication* is proving that a user is whom he or she claims to be. That proof may involve something the user knows (such as a password), something the user has (such as a "smartcard"), or something about the user that proves the person's identity (such as a fingerprint). *Authorization* is the act of determining whether a particular user (or computer system) has the right to carry out a certain activity, such as reading a file or running a program. Authentication and authorization go hand in hand. Users must be authenticated before carrying out the activity they are authorized to perform. Security is strong when the means of authentication cannot later be refuted—the user cannot later deny that he or she performed the activity. This is known as *nonrepudiation*.

Why care about security?

It is remarkably easy to gain unauthorized access to information in an insecure networked environment, and it is hard to catch the intruders. Even if users have nothing stored on their computer that they consider important, that computer can be a "weak link," allowing unauthorized access to the organization's systems and information.

Seemingly innocuous information can expose a computer system to compromise. Information that intruders find useful includes which hardware and software are being used, system configuration, type of network connections, phone numbers, and access and authentication procedures. Security-related information can enable unauthorized individuals to get access to important files and programs, thus compromising the security of the system. Examples of important information are passwords, access control files and keys, personnel information, and encryption algorithms.

Judging from CERT Coordination Center (CERT /CC) data and the computer abuse reported in the media, no one on the Internet is immune. Those affected include banks and financial companies, insurance companies, brokerage houses, consultants, government contractors, government agencies, hospitals and medical laboratories,

network service providers, utility companies, the textile business, universities, and wholesale and retail trades.

The consequences of a break-in cover a broad range of possibilities: a minor loss of time in recovering from the problem, a decrease in productivity, a significant loss of money or staff-hours, a devastating loss of credibility or market opportunity, a business no longer able to compete, legal liability, and the loss of life.

History

The Internet began in 1969 as the ARPANET, a project funded by the Advanced Research Projects Agency (ARPA) of the U.S. Department of Defense. One of the original goals of the project was to create a network that would continue to function even if major sections of the network failed or were attacked. The ARPANET was designed to reroute network traffic automatically around problems in connecting systems or in passing along the necessary information to keep the network functioning. Thus, from the beginning, the Internet was designed to be robust against denial-of-service attacks.

The ARPANET protocols (the rules of syntax that enable computers to communicate on a network) were originally designed for openness and flexibility, not for security. The ARPA researchers needed to share information easily, so everyone needed to be an unrestricted "insider" on the network. Although the approach was appropriate at the time, it is not one that lends itself to today's commercial and government use.

As more locations with computers (known as *sites* in Internet parlance) joined the ARPANET, the usefulness of the network grew. The ARPANET consisted primarily of university and government computers, and the applications supported on this network were simple: electronic mail (email), electronic news groups, and remote connection to other computers. By 1971, the Internet linked about two dozen research and government sites, and researchers had begun to use it to exchange information not directly related to the ARPANET itself. The network was becoming an important tool for collaborative research.

During these years, researchers also played "practical jokes" on each other using the ARPANET. These jokes usually involved joke messages, annoying messages, and other minor security violations. Some of these are described in Steven Levy's *Hackers: Heroes of the Computer Revolution*. It was rare that a connection from a remote system was considered an attack, however, because ARPANET users comprised a small group of people who generally knew and trusted each other.

In 1986, the first well-publicized international security incident was identified by Cliff Stoll, then of Lawrence Berkeley National Laboratory in northern California. A simple accounting error in the computer records of systems connected to the ARPANET led Stoll to uncover an international effort, using the network, to connect to computers in the United States and copy information from them. These U.S. computers were not only at universities, but at military and government sites all over the country. When Stoll published his experience in a 1989 book, *The Cuckoo's Egg*, he raised awareness that the ARPANET could be used for destructive purposes.

In 1988, the ARPANET had its first automated network security incident, usually referred to as "the Morris worm. " A student at Cornell University (Ithaca, N.Y.), Robert T. Morris, wrote a program that would connect to another computer, find and use one of several vulnerabilities to copy itself to that second computer, and begin to run the copy of itself at the new location. Both the original code and the copy would then repeat these actions in an infinite loop to other computers on the ARPANET. This "self-replicating automated network attack tool" caused a geometric explosion of copies to be started at computers all around the ARPANET. The worm used so many system resources that the attacked computers could no longer function. As a result, 10% of the U.S. computers connected to the ARPANET effectively stopped at about the same time.

By that time, the ARPANET had grown to more than 88,000 computers and was the primary means of communication among network security experts. With the ARPANET effectively down, it was difficult to coordinate a response to the worm. Many sites

removed themselves from the ARPANET altogether, further hampering communication and the transmission of the solution that would stop the worm.

The Morris worm prompted the Defense Advanced Research Projects Agency (DARPA, the new name for ARPA) to fund a computer emergency response team, now the CERT Coordination Center, to give experts a central point for coordinating responses to network emergencies. Other teams quickly sprang up to address computer security incidents in specific organizations or geographic regions. Within a year of their formation, these incident response teams created an informal organization now known as the Forum of Incident Response and Security Teams (FIRST). These teams and the FIRST organization exist to coordinate responses to computer security incidents, assist sites in handling attacks, and educate network users about computer security threats and preventive practices.

In 1989, the ARPANET officially became the Internet and moved from a government research project to an operational network; by then it had grown to more than 100,000 computers. Security problems continued, with both aggressive and defensive technologies becoming more sophisticated. Among the major security incidents were the 1989 WANK/OILZ worm, an automated attack on VMS systems attached to the Internet, and exploitation of vulnerabilities in widely distributed programs such as the sendmail program, a complicated program commonly found on UNIX-based systems for sending and receiving electronic mail. In 1994, intruder tools were created to "sniff" packets from the network easily, resulting in the widespread disclosure of user names and password information. In 1995, the method that Internet computers use to name and authenticate each other was exploited by a new set of attack tools that allowed widespread Internet attacks on computers that have trust relationships with any other computer, even one in the same room. Today the use of the World Wide Web and Web-related programming languages create new opportunities for network attacks.

Although the Internet was originally conceived of and designed as a research and education network, usage patterns have radically changed. The Internet has become a home for private and commercial communication, and at this writing it is still expanding into important areas of commerce, medicine, and public service. Increased reliance on the Internet is expected over the next five years, along with increased attention to its security.

Network security incidents

A *network security incident* is any network-related activity with negative security implications. This usually means that the activity violates an explicit or implicit security policy. Incidents come in all shapes and sizes. They can come from anywhere on the Internet, although some attacks must be launched from specific systems or networks and some require access to special accounts. An intrusion may be a comparatively minor event involving a single site or a major event in which tens of thousands of sites are compromised. (When reading accounts of incidents, note that different groups may use different criteria for determining the bounds of an incident.)

A typical attack pattern consists of gaining access to a user's account, gaining privileged access, and using the victim's system as a launch platform for attacks on other sites. It is possible to accomplish all these steps manually in as little as 45 seconds; with automation, the time decreases further.

Sources of incidents

It is difficult to characterize the people who cause incidents. An intruder may be an adolescent who is curious about what he or she can do on the Internet, a college student who has created a new software tool, an individual seeking personal gain, or a paid "spy" seeking information for the economic advantage of a corporation or foreign country. An incident may also be caused by a disgruntled former employee or a consultant who gained network information while working with a company. An intruder may seek entertainment, intellectual challenge, a sense of power, political attention, or financial gain.

One characteristic of the intruder community as a whole is its communication. There are electronic newsgroups and print publications on the latest intrusion techniques, as

well as conferences on the topic. Intruders identify and publicize misconfigured systems; they use those systems to exchange pirated software, credit card numbers, exploitation programs, and the identity of sites that have been compromised, including account names and passwords. By sharing knowledge and easy-to-use software tools, successful intruders increase their number and their impact.

Types of incidents

Incidents can be broadly classified into several kinds: the probe, scan, account compromise, root compromise, packet sniffer, denial of service, exploitation of trust, malicious code, and Internet infrastructure attacks.

Probe

A probe is characterized by unusual attempts to gain access to a system or to discover information about the system. One example is an attempt to log in to an unused account. Probing is the electronic equivalent of testing doorknobs to find an unlocked door for easy entry. Probes are sometimes followed by a more serious security event, but they are often the result of curiosity or confusion.

Scan

A scan is simply a large number of probes done using an automated tool. Scans can sometimes be the result of a misconfiguration or other error, but they are often a prelude to a more directed attack on systems that the intruder has found to be vulnerable.

Account compromise

An account compromise is the unauthorized use of a computer account by someone other than the account owner, without involving system-level or root-level privileges (privileges a system administrator or network manager has). An account compromise might expose the victim to serious data loss, data theft, or theft of services. The lack of root-level access means that the damage can usually be contained, but a user-level account is often an entry point for greater access to the system.

Root compromise

A root compromise is similar to an account compromise, except that the account that has been compromised has special privileges on the system. The term *root* is derived from an account on UNIX systems that typically has unlimited, or "superuser," privileges. Intruders who succeed in a root compromise can do just about anything on the victim's system, including run their own programs, change how the system works, and hide traces of their intrusion.

Packet sniffer

A packet sniffer is a program that captures data from information packets as they travel over the network. That data may include user names, passwords, and proprietary information that travels over the network in clear text. With perhaps hundreds or thousands of passwords captured by the sniffer, intruders can launch widespread attacks on systems. Installing a packet sniffer does not necessarily require privileged access. For most multi-user systems, however, the presence of a packet sniffer implies there has been a root compromise.

Denial of service

The goal of denial-of-service attacks is not to gain unauthorized access to machines or data, but to prevent legitimate users of a service from using it. A denial-of-service attack can come in many forms. Attackers may "flood" a network with large volumes of data or deliberately consume a scarce or limited resource, such as process control blocks or pending network connections. They may also disrupt physical components of the network or manipulate data in transit, including encrypted data.

Exploitation of trust

Computers on networks often have trust relationships with one another. For example, before executing some commands, the computer checks a set of files that specify which other computers on the network are permitted to use those commands. If attackers can forge their identity, appearing to be using the trusted computer, they may be able to gain unauthorized access to other computers.

Malicious code

Malicious code is a general term for programs that, when executed, would cause undesired results on a system. Users of the system usually are not aware of the program until they discover the damage. Malicious code includes Trojan horses, viruses, and worms. Trojan horses and viruses are usually hidden in legitimate programs or files that attackers have altered to do more than what is expected. Worms are self-replicating programs that spread with no human intervention after they are started. Viruses are also self-replicating programs, but usually require some action on the part of the user to spread inadvertently to other programs or systems. These sorts of programs can lead to serious data loss, downtime, denial of service, and other types of security incidents.

Internet infrastructure attacks

These rare but serious attacks involve key components of the Internet infrastructure rather than specific systems on the Internet. Examples are network name servers, network access providers, and large archive sites on which many users depend. Widespread automated attacks can also threaten the infrastructure. Infrastructure attacks affect a large portion of the Internet and can seriously hinder the day-to-day operation of many sites.

Incidents and Internet growth

Since the CERT Coordination Center began operating in 1988, the number of security incidents reported to the center has grown dramatically, from fewer than 100 in 1988 to almost 2,500 in 1995, the last year for which complete statistics are available as of this writing. Through 1994, the increase in incident reports roughly parallels the growth of the size of the Internet during that time.

Incident Trends

In the late 1980s and early 1990s, the typical intrusion was fairly straightforward. Intruders most often exploited relatively simple weaknesses, such as poor passwords and misconfigured systems, that allowed greater access to the system than was intended. Once on a system, the intruders exploited one or another well-known, but usually unfixed, vulnerability to gain privileged access, enabling them to use the system as they wished.

There was little need to be more sophisticated because these simple techniques were effective. Vendors delivered systems with default settings that made it easy to break into systems. Configuring systems in a secure manner was not straightforward, and many system administrators did not have the time, expertise, or tools to monitor their systems adequately for intruder activity.

Unfortunately, all these activities continue; however, more sophisticated intrusions are now common. In 10 years of operation, the CERT Coordination Center has seen intruders demonstrate increased technical knowledge, develop new ways to exploit system vulnerabilities, and create software tools to automate attacks. At the same time, intruders with little technical knowledge are becoming more effective as the sophisticated intruders share their knowledge and tools.

Intruders' technical knowledge

Intruders are demonstrating increased understanding of network topology, operations, and protocols, resulting in the infrastructure attacks described in the previous section on Internet infrastructure attacks.

Instead of simply exploiting well-known vulnerabilities, intruders examine source code to discover weaknesses in certain programs, such as those used for electronic mail. Much source code is easy to obtain from programmers who make their work freely available on the Internet. Programs written for research purposes (with little thought for security) or written by naive programmers become widely used, with source code available to all. Moreover, the targets of many computer intrusions are organizations that maintain copies of proprietary source code (often the source code to computer operating systems or key software utilities). Once intruders gain access, they can examine this code to discover weaknesses.

Intruders keep up with new technology. For example, intruders now exploit vulnerabilities associated with the World Wide Web to gain unauthorized access to systems.

Other aspects of the new sophistication of intruders include the targeting of the network infrastructure (such as network routers and firewalls) and the ability to cloak their behavior. Intruders use Trojan horses to hide their activity from network administrators; for example, intruders alter authentication and logging programs so that they can log in without the activity showing up in the system logs. Intruders also encrypt output from their activity, such as the information captured by packet sniffers. Even if the victim finds the sniffer logs, it is difficult or impossible to determine what information was compromised.

Techniques to exploit vulnerabilities

As intruders become more sophisticated, they identify new and increasingly complex methods of attack. For example, intruders are developing sophisticated techniques to monitor the Internet for new connections. Newly connected systems are often not fully configured from a security perspective and are, therefore, vulnerable to attacks.

The most widely publicized of the newer types of intrusion is the use of packet sniffers. Other tools are used to construct packets with forged addresses; one use of these tools is to mount a denial-of-service attack in a way that obscures the source of the attack. Intruders also "spoof" computer addresses, masking their real identity and successfully making connections that would not otherwise be permitted. In this way, they exploit trust relationships between computers.

With their sophisticated technical knowledge and understanding of the network, intruders are increasingly exploiting network interconnections. They move through the Internet infrastructure, attacking areas on which many people and systems depend. Infrastructure attacks are even more threatening because legitimate network managers and administrators typically think about protecting systems and parts of the infrastructure rather than the infrastructure as a whole.

In the first quarter of 1996, 7.5% of 346 incidents handled by the CERT Coordination Center involved these new and sophisticated methods, including packet sniffers, spoofing, and infrastructure attacks. A full 20% involved the total compromise of systems, in which intruders gain system-level, or root, privileges. This represents a significant increase in such attacks over previous years' attacks, and the numbers are still rising. Of 341 incidents in the third quarter of 1996, nearly 9% involved sophisticated attacks, and root compromises accounted for 33%.

Intruders' use of software tools

The tools available to launch an attack have become more effective, easier to use, and more accessible to people without an in-depth knowledge of computer systems. Often a sophisticated intruder embeds an attack procedure in a program and widely distributes it to the intruder community. Thus, people who have the desire but not the technical skill are able to break into systems. Indeed, there have been instances of intruders breaking into a UNIX system using a relatively sophisticated attack and then attempting to run DOS commands (commands that apply to an entirely different operating system).

Tools are available to examine programs for vulnerabilities even in the absence of source code. Though these tools can help system administrators identify problems, they also help intruders find new ways to break into systems.

As in many areas of computing, the tools used by intruders have become more automated, allowing intruders to gather information about thousands of Internet hosts quickly and with minimum effort. These tools can scan entire networks from a remote location and identify individual hosts with specific weaknesses. Intruders may catalog the information for later exploitation, share or trade with other intruders, or attack immediately. The increased availability and usability of scanning tools means that even technically naive, would-be intruders can find new sites and particular vulnerabilities.

Some tools automate multiphase attacks in which several small components are

combined to achieve a particular end. For example, intruders can use a tool to mount a denial-of-service attack on a machine and spoof that machine's address to subvert the intended victim's machine. A second example is using a packet sniffer to get router or firewall passwords, logging in to the firewall to disable filters, then using a network file service to read data on an otherwise secure server.

The trend toward automation can be seen in the distribution of software packages containing a variety of tools to exploit vulnerabilities. These packages are often maintained by competent programmers and are distributed complete with version numbers and documentation.

A typical tool package might include the following:

- network scanner
- password-cracking tool and large dictionaries
- packet sniffer
- variety of Trojan horse programs and libraries
- tools for selectively modifying system log files
- tools to conceal current activity
- Tools for automatically modifying system configuration files
- tools for reporting bogus checksums

Internet vulnerabilities

A vulnerability is a weakness that a person can exploit to accomplish something that is not authorized or intended as legitimate use of a network or system. When a vulnerability is exploited to compromise the security of systems or information on those systems, the result is a security incident. Vulnerabilities may be caused by engineering or design errors, or faulty implementation.

Why the Internet is vulnerable

Many early network protocols that now form part of the Internet infrastructure were designed without security in mind. Without a fundamentally secure infrastructure, network defense becomes more difficult. Furthermore, the Internet is an extremely dynamic environment, in terms of both topology and emerging technology.

Because of the inherent openness of the Internet and the original design of the protocols, Internet attacks in general are quick, easy, inexpensive, and may be hard to detect or trace. An attacker does not have to be physically present to carry out the attack. In fact, many attacks can be launched readily from anywhere in the world—and the location of the attacker can easily be hidden. Nor is it always necessary to "break in" to a site (gain privileges on it) to compromise confidentiality, integrity, or availability of its information or service.

Even so, many sites place unwarranted trust in the Internet. It is common for sites to be unaware of the risks or unconcerned about the amount of trust they place in the Internet. They may not be aware of what can happen to their information and systems. They may believe that their site will not be a target or that precautions they have taken are sufficient. Because the technology is constantly changing and intruders are constantly developing new tools and techniques, solutions do not remain effective indefinitely.

Since much of the traffic on the Internet is not encrypted, confidentiality and integrity are difficult to achieve. This situation undermines not only applications (such as financial applications that are network-based) but also more fundamental mechanisms such as authentication and nonrepudiation. As a result, sites may be affected by a security compromise at another site over which they have no control. An example of this is a packet sniffer that is installed at one site but allows the intruder to gather information about other domains (possibly in other countries).

Another factor that contributes to the vulnerability of the Internet is the rapid growth and use of the network, accompanied by rapid deployment of network services involving complex applications. Often, these services are not designed, configured, or

maintained securely. In the rush to get new products to market, developers do not adequately ensure that they do not repeat previous mistakes or introduce new vulnerabilities.

Compounding the problem, operating system security is rarely a purchase criterion. Commercial operating system vendors often report that sales are driven by customer demand for performance, price, ease of use, maintenance, and support. As a result, off-the-shelf operating systems are shipped in an easy-to-use but insecure configuration that allows sites to use the system soon after installation. These hosts/sites are often not fully configured from a security perspective before connecting. This lack of secure configuration makes them vulnerable to attacks, which sometimes occur within minutes of connection.

Finally, the explosive growth of the Internet has expanded the need for well-trained and experienced people to engineer and manage the network in a secure manner. Because the need for network security experts far exceeds the supply, inexperienced people are called upon to secure systems, opening windows of opportunity for the intruder community.

Types of technical vulnerabilities

The following taxonomy is useful in understanding the technical causes behind successful intrusion techniques, and helps experts identify general solutions for addressing each type of problem.

Flaws in software or protocol designs

Protocols define the rules and conventions for computers to communicate on a network. If a protocol has a fundamental design flaw, it is vulnerable to exploitation no matter how well it is implemented. An example of this is the Network File System (NFS), which allows systems to share files. This protocol does not include a provision for authentication; that is, there is no way of verifying that a person logging in really is whom he or she claims to be. NFS servers are targets for the intruder community.

When software is designed or specified, often security is left out of the initial description and is later "added on" to the system. Because the additional components were not part of the original design, the software may not behave as planned and unexpected vulnerabilities may be present.

Weaknesses in how protocols and software are implemented

Even when a protocol is well designed, it can be vulnerable because of the way it is implemented. For example, a protocol for electronic mail may be implemented in a way that permits intruders to connect to the mail port of the victim's machine and fool the machine into performing a task not intended by the service. If intruders supply certain data for the "To:" field instead of a correct email address, they may be able to fool the machine into sending them user and password information or granting them access to the victim's machine with privileges to read protected files or run programs on the system. This type of vulnerability enables intruders to attack the victim's machine from remote sites without access to an account on the victim's system. This type of attack often is just a first step, leading to the exploitation of flaws in system or application software.

Software may be vulnerable because of flaws that were not identified before the software was released. This type of vulnerability has a wide range of subclasses, which intruders often exploit using their own attack tools. For readers who are familiar with software design, the following examples of subclasses are included:

- race conditions in file access
- non-existent checking of data content and size
- non-existent checking for success or failure
- inability to adapt to resource exhaustion
- incomplete checking of operating environment
- inappropriate use of system calls
- reuse of software modules for purposes other than their intended ones

By exploiting program weaknesses, intruders at a remote site can gain access to a victim's system. Even if they have access to a nonprivileged user account on the victim's system, they can often gain additional, unauthorized privileges.

Weaknesses in system and network configurations

Vulnerabilities in the category of system and network configurations are not caused by problems inherent in protocols or software programs. Rather, the vulnerabilities are a result of the way these components are set up and used. Products may be delivered with default settings that intruders can exploit. System administrators and users may neglect to change the default settings, or they may simply set up their system to operate in a way that leaves the network vulnerable.

An example of a faulty configuration that has been exploited is anonymous File Transfer Protocol (FTP) service. Secure configuration guidelines for this service stress the need to ensure that the password file, archive tree, and ancillary software are separate from the rest of the operating system, and that the operating system cannot be reached from this staging area. When sites misconfigure their anonymous FTP archives, unauthorized users can get authentication information and use it to compromise the system.

Improving security

In the face of the vulnerabilities and incident trends discussed above, a robust defense requires a flexible strategy that allows adaptation to the changing environment, well-defined policies and procedures, the use of robust tools, and constant vigilance.

It is helpful to begin a security improvement program by determining the current state of security at the site. Methods for making this determination in a reliable way are becoming available. Integral to a security program are documented policies and procedures, and technology that supports their implementation.

Security policy, procedures, and practices

Security policy

A policy is a documented high-level plan for organization-wide computer and information security. It provides a framework for making specific decisions, such as which defense mechanisms to use and how to configure services, and is the basis for developing secure programming guidelines and procedures for users and system administrators to follow. Because a security policy is a long-term document, the contents avoid technology-specific issues.

A security policy covers the following (among other topics appropriate to the organization):

- high-level description of the technical environment of the site, the legal environment (governing laws), the authority of the policy, and the basic philosophy to be used when interpreting the policy
- risk analysis that identifies the site's assets, the threats that exist against those assets, and the costs of asset loss
- guidelines for system administrators on how to manage systems
- definition of acceptable use for users
- guidelines for reacting to a site compromise (e.g., how to deal with the media and law enforcement, and whether to trace the intruder or shutdown and rebuild the system)

Factors that contribute to the success of a security policy include management commitment, technological support for enforcing the policy, effective dissemination of the policy, and the security awareness of all users. Management assigns responsibility for security, provides training for security personnel, and allocates funds to security. Technological support for the security policy moves some responsibility for enforcement from individuals to technology. The result is an automatic and consistent enforcement of policies, such as those for access and authentication. Technical options that support policy include (but are not limited to)

- challenge/response systems for authentication
- auditing systems for accountability and event reconstruction
- encryption systems for the confidential storage and transmission of data
- network tools such as firewalls and proxy servers

There are many books and papers devoted to site security policies, including requests for comments RFC 1244 (6) and RFC 1281 (7), guidelines written by the Internet Engineering Task Force.

Security-related procedures

Procedures are specific steps to follow that are based on the computer security policy. Procedures address such topics as retrieving programs from the network, connecting to the site's system from home or while traveling, using encryption, authentication for issuing accounts, configuration, and monitoring.

Security practices

System administration practices play a key role in network security. Checklists and general advice on good security practices are readily available. Below are examples of commonly recommended practices:

- Ensure all accounts have a password and that the passwords are difficult to guess. A one-time password system is preferable.
- Use tools such as MD5 checksums (8), a strong cryptographic technique, to ensure the integrity of system software on a regular basis.
- Use secure programming techniques when writing software.
- Be vigilant in network use and configuration, making changes as vulnerabilities become known.
- Regularly check with vendors for the latest available fixes, and keep systems current with upgrades and patches.
- Regularly check online security archives, such as those maintained by incident response teams, for security alerts and technical advice.
- Audit systems and networks, and regularly check logs. Many sites that suffer computer security incidents report that insufficient audit data is collected, so detecting and tracing an intrusion is difficult.

Security technology

A variety of technologies have been developed to help organizations secure their systems and information against intruders. These technologies help protect systems and information against attacks, detect unusual or suspicious activities, and respond to events that affect security. In this section, the focus is on two core areas: operational technology and cryptography. The purpose of operational technology is to maintain and defend the availability of data resources in a secure manner. The purpose of cryptography is to secure the confidentiality, integrity, and authenticity of data resources.

Operational technology

Intruders actively seek ways to access networks and hosts. Armed with knowledge about specific vulnerabilities, social engineering techniques, and tools to automate information gathering and systems infiltration, intruders can often gain entry into systems with disconcerting ease. System administrators face the dilemma of maximizing the availability of system services to valid users while minimizing the susceptibility of complex network infrastructures to attack. Unfortunately, services often depend on the same characteristics of systems and network protocols that make them susceptible to compromise by intruders. In response, technologies have evolved to reduce the impact of such threats. No single technology addresses all the problems. Nevertheless, organizations can significantly improve their resistance to attack by carefully preparing and strategically deploying personnel and operational technologies. Data resources and assets can be protected, suspicious activity can be detected and assessed, and appropriate responses can be made to security events as they occur.

One-Time Passwords. Intruders often install packet sniffers to capture passwords as they traverse networks during remote login processes. Therefore, all passwords should at least be encrypted as they traverse networks. A better solution is to use one-time passwords because there are times when a password is required to initiate a connection before confidentiality can be protected.

One common example occurs in remote dial-up connections. Remote users, such as those traveling on business, dial in to their organization's modem pool to access network and data resources. To identify and authenticate themselves to the dial-up server, they must enter a user ID and password. Because this initial exchange between the user and server may be monitored by intruders, it is essential that the passwords are not reusable. In other words, intruders should not be able to gain access by masquerading as a legitimate user using a password they have captured.

One-time password technologies address this problem. Remote users carry a device synchronized with software and hardware on the dial-up server. The device displays random passwords, each of which remains in effect for a limited time period (typically 60 seconds). These passwords are never repeated and are valid only for a specific user during the period that each is displayed. In addition, users are often limited to one successful use of any given password. One-time password technologies significantly reduce unauthorized entry at gateways requiring an initial password.

Firewalls. Intruders often attempt to gain access to networked systems by pretending to initiate connections from trusted hosts. They squash the emissions of the genuine host using a denial-of-service attack and then attempt to connect to a target system using the address of the genuine host. To counter these address-spoofing attacks and enforce limitations on authorized connections into the organization's network, it is necessary to filter all incoming and outgoing network traffic.

A firewall is a collection of hardware and software designed to examine a stream of network traffic and service requests. Its purpose is to eliminate from the stream those packets or requests that fail to meet the security criteria established by the organization. A simple firewall may consist of a filtering router, configured to discard packets that arrive from unauthorized addresses or that represent attempts to connect to unauthorized service ports. More sophisticated implementations may include bastion hosts, on which proxy mechanisms operate on behalf of services. These mechanisms authenticate requests, verify their form and content, and relay approved service requests to the appropriate service hosts. Because firewalls are typically the first line of defense against intruders, their configuration must be carefully implemented and tested before connections are established between internal networks and the Internet.

*Monitoring Tools.*Continuous monitoring of network activity is required if a site is to maintain confidence in the security of its network and data resources. Network monitors may be installed at strategic locations to collect and examine information continuously that may indicate suspicious activity. It is possible to have automatic notifications alert system administrators when the monitor detects anomalous readings, such as a burst of activity that may indicate a denial-of-service attempt. Such notifications may use a variety of channels, including electronic mail and mobile paging. Sophisticated systems capable of reacting to questionable network activity may be implemented to disconnect and block suspect connections, limit or disable affected services, isolate affected systems, and collect evidence for subsequent analysis.

Tools to scan, monitor, and eradicate viruses can identify and destroy malicious programs that may have inadvertently been transmitted onto host systems. The damage potential of viruses ranges from mere annoyance (e.g., an unexpected "Happy Holidays" jingle without further effect) to the obliteration of critical data resources. To ensure continued protection, the virus identification data on which such tools depend must be kept up to date. Most virus tool vendors provide subscription services or other distribution facilities to help customers keep up to date with the latest viral strains.

*Security Analysis Tools.*Because of the increasing sophistication of intruder methods and the vulnerabilities present in commonly used applications, it is essential to assess periodically network susceptibility to compromise. A variety of vulnerability

identification tools are available, which have garnered both praise and criticism. System administrators find these tools useful in identifying weaknesses in their systems. Critics argue that such tools, especially those freely available to the Internet community, pose a threat if acquired and misused by intruders.

Cryptography

One of the primary reasons that intruders can be successful is that most of the information they acquire from a system is in a form that they can read and comprehend. When you consider the millions of electronic messages that traverse the Internet each day, it is easy to see how a well-placed network sniffer might capture a wealth of information that users would not like to have disclosed to unintended readers. Intruders may reveal the information to others, modify it to misrepresent an individual or organization, or use it to launch an attack. One solution to this problem is, through the use of cryptography, to prevent intruders from being able to use the information that they capture.

Encryption is the process of translating information from its original form (called *plaintext*) into an encoded, incomprehensible form (called *ciphertext*). Decryption refers to the process of taking ciphertext and translating it back into plaintext. Any type of data may be encrypted, including digitized images and sounds.

Cryptography secures information by protecting its confidentiality. Cryptography can also be used to protect information about the integrity and authenticity of data. For example, checksums are often used to verify the integrity of a block of information. A checksum, which is a number calculated from the contents of a file, can be used to determine if the contents are correct. An intruder, however, may be able to forge the checksum after modifying the block of information. Unless the checksum is protected, such modification might not be detected. Cryptographic checksums (also called message digests) help prevent undetected modification of information by encrypting the checksum in a way that makes the checksum unique.

The authenticity of data can be protected in a similar way. For example, to transmit information to a colleague by email, the sender first encrypts the information to protect its confidentiality and then attaches an encrypted digital signature to the message. When the colleague receives the message, he or she checks the origin of the message by using a key to verify the sender's digital signature and decrypts the information using the corresponding decryption key. To protect against the chance of intruders modifying or forging the information in transit, digital signatures are formed by encrypting a combination of a checksum of the information and the author's unique private key. A side effect of such authentication is the concept of nonrepudiation. A person who places their cryptographic digital signature on an electronic document cannot later claim that they did not sign it, since in theory they are the only one who could have created the correct signature.

Current laws in several countries, including the United States, restrict cryptographic technology from export or import across national borders. In the era of the Internet, it is particularly important to be aware of all applicable local and foreign regulations governing the use of cryptography.

Information warfare

Extensive and widespread dependence on the Internet has called new attention to the importance of information to national security. The term *information warfare* refers to the act of war against the information resources of an adversary. Like warfare on land or in the air, information warfare is one component of a range of attack strategies for dominating an adversary in order to gain or maintain an objective.

Information warfare is divided into two categories: offensive and defensive. The purpose of offensive information warfare is to attack the information resources of an adversary to gain dominance. Defensive information warfare is the protection of your information assets against attack.

Information assets can take many forms, from messages sent by courier in diplomatic bags to the computers used to analyze enemy positions based on satellite data. In computer security, information assets include digital information, the computers that

process them, and the networks that transmit the digital information from place to place. Computer security is a key element for protecting the availability, integrity, and confidentiality of all these information assets.

Internet security protects information assets consisting of computers, information, and networks that are part of the Internet. Internet security is related to information warfare when the Internet contains information assets that are important to the information warfare objective. For example, if an adversary can use the Internet to access battle plans, the Internet is being used for information warfare.

Internet security is important to both offensive and defensive information warfare because the Internet is a global and dependable resource on which many countries rely. Historically, military networks and computers were unreachable by nonmilitary participants. The Internet, however, provides a cost-effective way for military and government units to communicate and participate in achieving objectives. Use of the Internet means that individuals, multinational companies, and terrorist organizations all can gain access to important information resources of governments and military forces. Thus, it is important to address Internet security concerns as a key component of defensive information warfare.

Because the Internet is global, it can be an avenue of attack for offensive information warfare by many governments. One of the battlefields for a future military offensive could very well involve the Internet. Intruder technology could be used by a government as a weapon against information resources, or used randomly by a terrorist organization against civilian targets.

In the study of information warfare, there are many new problems to solve that are not evident in other forms of warfare. These problems include identifying the enemy, responding without making your systems vulnerable to attack, and gathering intelligence on the Internet about preparations for a military exercise. These and other problems are likely to be the subject of discussion and investigation for some time to come.

The future

Research and development efforts are underway to allow critical applications to operate in the future in a more secure environment than exists today.

Internetworking protocols

Most of the network protocols currently in use have changed little since the early definitions of the ARPA research and education network when trust was the norm. To have a secure foundation for the critical Internet applications of the future, severe weaknesses must be addressed: lack of encryption to preserve privacy, lack of cryptographic authentication to identify the source of information, and lack of cryptographic checksums to preserve the integrity of data (and the integrity of the packet routing information itself). New internetworking protocols are under development that use cryptography to authenticate the originator of a packet and to protect the integrity and confidentiality of data.

The IETF (Internet Engineering Task Force) Proposed Standard for the Next Generation Internet Protocol (IPng) is being designed to cope with the vastly increased addressing and routing needs associated with the exponential growth of the Internet. IPng provides integral support for authenticating hosts and protecting the integrity and confidentiality of data.

The first release of IPng is officially termed IPv6 (Internet Protocol version 6). Since it is impractical to replace the existing protocol instantly and simultaneously throughout the Internet, IPv6 is designed to coexist with the current version of IP, allowing for a gradual transition over the course of years. Implementations of IPv6 for many routers and host operating systems are underway.

In the future, authentication protocols will increasingly be supported by technology that authenticates individuals (in the context of their organizational or personal roles) through the use of smart cards, fingerprint readers, voice recognition, retina scans, and so forth.

Protocol design, analysis, and implementation will be the subject of continued research. A primary goal is 100% verifiably secure protocols (that is, protocols as provably secure as the cryptographic algorithms supporting them), but researchers are nowhere near attaining this goal.

Intrusion detection

Research is underway to improve the ability of networked systems and their managers to determine that they are, or have been, under attack. Intrusion detection is recognized as a problematic area of research that is still in its infancy. There are two major areas of research in intrusion detection: anomaly detection and pattern recognition.

Research in anomaly detection is based on determining patterns of "normal" behavior for networks, hosts, and users and then detecting behavior that is significantly different (anomalous). Patterns of normal behavior are frequently determined through data collection over a period of time sufficient to obtain a good sample of the typical behavior of authorized users and processes. The basic difficulty facing researchers is that normal behavior is highly variable based on a wide variety of innocuous factors. Many of the activities of intruders are indistinguishable from the benign actions of an authorized user.

The second major area of intrusion detection research is pattern recognition. The goal here is to detect patterns of network, host, and user activity that match known intruder attack scenarios. One problem with this approach is the variability that is possible within a single overall attack strategy. A second problem is that new attacks, with new attack patterns, cannot be detected by this approach.

Finally, to support the needs of the future Internet, intrusion detection tools and techniques that can identify coordinated distributed attacks are critically needed, as are better protocols to support traceability.

Software engineering and system survivability

Current software engineering methods and practice have had only limited success in managing the intellectual complexity of designing and implementing software. Moreover, in the design of software systems, security concerns are typically an afterthought (addressed through add-ons and software patches) rather than being an integral part of the overall design. This means that software systems of any significant size and complexity are likely to have exploitable security flaws. Because managing the intellectual complexity of software is difficult, up-front security design in products is rare, and detailed knowledge about systems is widespread, systems will be breached in spite of our best efforts to make them invulnerable. Therefore, the concept of information systems security must encompass the specification of systems that exhibit behaviors that contribute to survivability in spite of intrusions. Only then can systems be developed that are robust in the presence of attack and are able to survive attacks that cannot be completely repelled.

System survivability is the capacity of a system to continue performing critical functions in a timely manner even if significant portions of the system are incapacitated by attack or accident. We use the term *system* in the broadest possible sense, which includes networks and large-scale "systems of systems."

Although the concepts and practices associated with system survivability are embryonic, they include (but are not limited to) traditional areas of software engineering and computer science such as reliability, testing, dependability, fault tolerance, verification of correctness, performance, and information system security. Promising research in survivability encompasses a wide variety of research methods in software engineering. Inoculation tools may be developed that will automate the distribution of security fixes, throughout an entire network infrastructure, to provide comprehensive protection from a newly discovered security flaw. The concept of inoculation may be further generalized to encompass adaptive networks, which consist of distributed cooperative network elements that exchange information on security problems and actively change and adjust in response to security threats.

Web-related programming and scripting languages

Downloading interesting, informative, or entertaining "content" from a remote site to a user's local machine is central to the activity of Web browsing (or "net surfing"). The content getting the most attention from Web users and the greatest concern from security experts is executable content, code to be executed on the local machine on download. This executable content may provide live audio of a conference in progress, a jazz tune, three-dimensional (3-D) animation effects, or hostile code that destroys the local file system. Executable code is authored using one or more Web-related programming or scripting languages designed specifically for the production of platform-independent executable content. Languages in this category include JAVA and ActiveX. Executable content is called an "applet" in JAVA and a "control panel" in ActiveX.

Web-related programming languages pose new security challenges and concerns because code is downloaded, installed, and run on a user's machine without a review of source code (the recommended practice for secure use of publicly available software). These activities can be triggered by following any hypertext link or opening any page while browsing. A user may not even be aware that code has been downloaded and executed. Some Web-related programming languages, most notably JAVA, have built-in security features, but security experts are concerned about the adequacy of these features.

As executable content makes Web browsing even more alluring, further research in software engineering and greater user awareness will be necessary to counter security risks. Presently, the security of executable content depends upon the correctness of multiple vendors' implementations, the inherent security of platform-independent "virtual machines," and the safety of the source code that is executed. In the foreseeable future, users need to be educated about the risks so they can make informed choices about where to place their trust.

Intelligent autonomous agents - a new computing paradigm

The future Internet environment is likely to be increasingly dependent on an agent-based model of computing, with significant implications for Internet security. Agents are executable software objects with executions that are not tied to any specific host or computing resource or to any geographical or logical network location. Agents perform computation and communication defined by a user, but the execution platforms are typically outside the user's administrative control (and outside the administrative control of the user's organization). The conceptual model of agent operation is one in which an intelligent agent, at the request of a user, goes to one or more remote hosts to perform a computation or gather information and then returns to the user with the result. An agent's mode of operation may range from partially to fully autonomous, and the degree to which an agent is autonomous may vary throughout the life of that agent.

A future agent-based computing environment may include features such as these:

- Agents share information and cooperate to complete the user's task.
- Agents protect themselves with intrinsic security mechanisms but also depend on some measure of extrinsic security provided by the infrastructure and cooperating agents.
- Since most of an agent's activity takes place outside the user's domain of administrative control (and hence outside any firewall designed to protect the user), the traditional firewall has little to contribute to security.
- Replication and agent diversity provide increased survivability while under attack and under conditions of degraded or uncertain infrastructure support.
- Agents communicate to enhance the detection of threats. Specialized sensor agents are specifically designed to detect particular types of threats, and groups of diverse sensor agents provide the entire agent "collective" with a comprehensive profile of current threats.
- The agent-supported infrastructure protects itself and takes defensive action without user intervention.

List of acronyms

ARPA - Advanced Research Projects Agency
CERT /CC - CERT Coordination Center
DARPA - Defense Advanced Research Projects Agency
FIRST - Forum of Incident Response and Security Teams
FTP - File Transfer Protocol
IETF - Internet Engineering Task Force
IP - Internet Protocol
IPng - Next Generation Internet Protocol (official name is IPv6)
IPv6 - Internet Protocol version 6 (also informally called IPng)
NFS - Network File System
RFC - request for comments

Bibliography

Caelli, W., Longley, D., and Shain, M. *Information Security Handbook*. New York: Stockton Press, 1991.

Chapman, D. B., and Zwicky, E. D. *Building Internet Firewalls*. Sebastopol, CA: O'Reilly & Associates, 1995.

Cheswick, W. R., and Bellovin, S. M. *Firewalls and Internet Security: Repelling the Wily Hacker*. New York: Addison-Wesley, 1994.

Comer, D. E. *Internetworking with TCP/IP*, 3 vols. Englewood Cliffs, NJ: Prentice-Hall, 1991 and 1993.

Garfinkel, S. *PGP: Pretty Good Privacy*. Sebastopol, CA: O'Reilly & Associates, 1995.

Garfinkel, S., and Spafford, G. *Practical UNIX and Internet Security*, 2d ed. Sebastopol, CA: O'Reilly & Associates, 1996.

Kaufman, C., Perlman, R., and Speciner, M. *Network Security: Private Communication in a Public World*. Englewood Cliffs, NJ: PTR Prentice-Hall, 1995.

McGraw, G., and Felten, E. W. *Java Security*. New York: John Wiley & Sons, 1996.

National Research Council. *Computers at Risk: Safe Computing in the Information Age*. Washington, DC: National Academy Press, 1991.

Schneier, B. *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, 2d ed. New York: John Wiley & Sons, 1996.

References

1. *Network Wizards Internet Domain Survey*.
2. Levy, S. *Hackers: Heroes of the Computer Revolution*. Garden City, NY: Anchor Press/Doubleday, 1984.
3. Stoll, C. *The Cuckoo's Egg: Tracking a Spy Through the Maze of Computer Espionage*. New York: Doubleday, 1989.
4. Denning, P. J., (ed.). *Computers Under Attack: Intruders, Worms, and Viruses*. New York: ACM Press, Addison-Wesley, 1990.
5. CERT Coordination Center. CERT advisories and other security information. CERT /CC, Pittsburgh, PA.
6. Internet Engineering Task Force, Site Security Policy Handbook Working Group. *Site Security Handbook*, RFC 1244. Note: RFC 1244 has been replaced by RFC 2196.
7. Internet Engineering Task Force, Network Working Group. *Guidelines for the Secure Operation of the Internet*, RFC 1281.
8. Internet Engineering Task Force, Network Working Group. *The MD5 Message-Digest Algorithm*.

For more information

Contact Us

info@sei.cmu.edu

412-268-5800

CERT Coordination Center Celebrates Ten Years

NEWS AT SEI

This article was originally published in News at SEI on: December 1, 1998

Ten years ago this October, a young college student created a "worm" program that resulted in the first Internet security incident to make headline news. (For more about the worm program, see our Background feature, "Security of the Internet.") It was the wake-up call for network security. In response, the CERTCoordination Center (CERT/CC) was established at the SEI. Its charter was to work with the Internet community to respond to computer security events, to raise the community's awareness of computer security issues, and to prevent security breaches.

The need for the CERT/CC has grown along with the growth of the Internet. In 1988, there were about 88,000 computers on the Internet. In 1998, as the CERT/CC celebrates its 10th anniversary, there are an estimated 50,000,000 users on the Net. Even more significantly, the U.S. government and U.S. commerce grow increasingly dependent on networked systems every year. For example, the Department of Defense (DoD) is moving away from expensive custom software to commercial-off-the-shelf software and commercially supported networking products. Federal mandates for online electronic records with remote access, along with the low costs of using the Internet, are accelerating the DoD's Internet use.

Along with the rapid increase in the size of the Internet and its use for critical functions, there have been progressive changes in intruder techniques, increased amounts of damage, increased difficulty of detecting an attack, and increased difficulty of catching the attackers. In December 1988, the CERT/CC responded to its first incident report on the first day of operation. Weeks later, it published its first security alert. While continuing to respond to security incidents and publish advisories, the CERT/CC has expanded its role over the years to better meet the needs of the Internet community. Its early work now provides the foundation for the SEI Survivable Systems Initiative.

Current areas of focus for the Survivable Systems Initiative include

- *incident response*: The CERT/CC provides assistance to computer system administrators in the Internet community who report security problems. When a security breach occurs, CERT/CC staff members help the administrators of the affected sites to identify and correct the vulnerabilities that allowed the incident to

occur. The CERT/CC staff also coordinates the response with other sites affected by the same incident.

- *survivable network management:* Members of the SEI Survivable Systems Initiative address problems in operational practice through four related products:

    1. security practices that provide concrete, practical guidance that helps organizations improve the security of their networked computer systems. These practices are published as security improvement "modules" that focus on best practices in network security
    2. an information security evaluation method that organizations can use to identify vulnerabilities in their networked systems and keep up with changes over time
    3. adaptive security management, a process for organizations to improve the security of their networked systems by changing their software engineering practices
    4. training courses in using the evaluation method and the adaptive management process, along with recommended curricula to address the needs of information security professionals

- *survivable network technology*: The Survivable Systems Initiative is concentrating on the technical basis for identifying and preventing security flaws and for preserving essential services if a system is penetrated and compromised.

In this section

In this release of *news@sei*, we celebrate the 10th anniversary of the CERT/CC by inviting staff members from the SEI Survivable Systems Initiative to tell the story of the past 10 years--and speculate about the outlook for the future--in their own words.

For a detailed introduction to the topic of network security, see our Background feature, Security of the Internet. This article was written by seven members of the CERT/CC staff and was first published in *The Froehlich/Kent Encyclopedia of Telecommunications vol. 15*, pp. 231-255, New York: Marcel Dekker, 1997.

In our Spotlight feature, we present an interview with Richard D. Pethia, manager of the Survivable Systems Initiative and first manager of the CERT/CC. In the interview, Rich looks back at the past 10 years and discusses topics that include intruders and how they have changed, the need to protect the critical infrastructure, the need for R&D and training in network survivability, and what organizations can do to help themselves.

In our Roundtable feature, Thomas A. Longstaff and David Fisher turn their attention to the future in a wide-ranging discussion about The Next Ten Years. Topics include the current state of the Internet; the trend toward complex interconnected systems; adaptive, self-correcting systems; new approaches in survivability research; defense in depth and decentralization; and survivability: the future of security technology.

(< 5 minute) survey.

Find Us Here

Share This Page

For more information

Contact Us

info@sei.cmu.edu

412-268-5800

Interview with Richard D. Pethia

NEWS AT SEI

Author

Bill Pollak

This article was originally published in News at SEI on: December 1, 1998

Richard D. Pethia is manager of the SEI Survivable Systems Initiative and first manager of the CERT® Coordination Center (CERT/CC). In this interview, Rich looks back at the past 10 years and discusses topics that include

- history
- CERT /CC principles
- Internet security in the news
- intruders and how they have changed
- serious incidents
- protecting the critical infrastructure
- R&D and training in information security
- security of software products
- what organizations can do
- the future of Internet security

History

Bill Pollak (interviewer): Tell me how the CERT®Coordination Center (CERT/CC) began.

Rich Pethia: We started the CERT /CC in November of 1988. At that time the SEI wasn't doing any work in information security. There were only about 88,000 computers connected to the Internet.

BP: How about now?

RP: There are an estimated 50 million users on the Net, doubling in size every 10 to 12

months. On November 2, 1988, there was a graduate student at Cornell University who let loose a "worm" on the Internet, and it basically crippled the network. If I remember correctly, it was released at night, and people began to see some strange things happening by the next morning. Systems all over the place were failing to operate because the Internet was clogged up with pieces of the worm sending messages to other pieces of the worm. It basically used all the bandwidth of the Internet. There were a lot of people who tried to solve the problem. The Internet has no formal structure, but research scientists at Berkeley, MIT, and Purdue, some of the people at the Defense Advanced Research Projects Agency (DARPA), some of the people on campus here at Carnegie Mellon, some of the computing facilities people here at the SEI, and people at a number of other schools around the country all began to try to understand what was going on. They started informally communicating with one another, captured a piece of the worm, and then began to reverse engineer it. All told, it took a couple of days to understand what was happening and get corrections out to the world so that they could get this thing out of the network.

BP: So this happened on a grassroots level; there was no organization to the response.

RP: Exactly. Then there was a series of meetings later in Washington, DC. They were called by the National Computer Security Center (NCSC), a public arm of the National Security Agency (NSA). They called together a lot of participants and asked for ideas on what might be put in place so that if this kind of problem ever happened again, the response would be more effective.

BP: Did they contain the problem in a couple of days?

RP: Yeah. It's pretty amazing; people really worked effectively together to contain the problem. There was lots of debate in Washington as to whether or not there should be a centralized approach or a decentralized one, and if industry or government should do it. And finally, the director of DARPA, Craig Fields, said "Well, in lieu of all this discussion, which may never end ... let's just do something." So they called up Larry Druffel, who was then the director of the SEI, and said, "We want to start a computer emergency response team." We talked to them for a couple of days, so by this time it was the middle of November--November 17, 1988.

BP: At this point, what were you doing at the SEI?

RP: I was beginning to put together a customer service group and helping put some more structure into the way we went about marketing the SEI's products and services. When Larry got the call from DARPA, he called together his managers and tried to decide whether or not the SEI should do this. There was a lot of internal debate; a lot of people thought that this type of function didn't belong in an R&D facility. But he decided to go ahead with it. Somehow, my name was tossed onto the table as the person who ought to start it. So I met with Larry, got myself unhooked from other activities, formed a small advisory group of people inside the SEI as well as a small group of people outside the SEI, and over the next two weeks, we put together a strategy on how we would start such a center. Larry and DARPA had a handshake over money, and on December 7, 1988, DARPA sent out a press release that announced our existence. The phone rang for the first time that night. We had six hours of idle time before we got the first call!

On December 7, 1988, DARPA sent out a press release that announced our existence. We had six hours of idle time before we got the first call!

BP: What was the first call?

RP: The first call was from some folks at a research lab on the West Coast that had a hacker in their system. They were trying to trace him back to wherever he was, they were getting very little support from anybody, and they needed some help.

BP: What were the other options available to an organization like this at the time?

RP: They were trying to call the investigative organizations--the Secret Service, the FBI--and they couldn't find anybody interested in what was happening to them. And that's really all they could do. At the time, there really was no place else to call. So we ended up working with this guy for the next two weeks, until we effectively got to the

point where it didn't look like--because the hacker was offshore and no investigators were interested--anyone would be able to make an arrest. But we finally talked him out of beating up the systems on the West Coast.

BP: So you actually got on the phone with him?

RP: Yeah. In the early days, we did a lot of that, because nobody would investigate. So the best we could do was to try to convince these people to stop breaking into the systems in the U.S.

BP: Are there now better laws in place than there were then?

RP: There are currently better laws in place and more legal precedent as well. And that's true both nationally and internationally. There has been legislation that has laid down law in this area**,** and the investigative organizations have built up their capabilities as well.

CERT/CC principles

RP: So that's how we got started. We didn't really have a good feeling for what we would do up front. But there were a couple of principles that we put in place early on that we still hold to today. One of them was the principle of confidentiality: When people call us and tell us about their problems, we promise them that we won't divulge that information to anyone else without their permission. The only exception to this is if we get a subpoena, which of course we honor. Another principle was that we didn't want to set ourselves up to be the organization that would solve all these problems, but rather the organization that would facilitate the response. And that's why we called ourselves the CERT® Coordination Center.

BP: By "facilitate," you mean to help people to help themselves?

RP: Yes. We recognized the excellent work that was done in resolving the problem of that original Internet worm. We didn't want to cut out big pieces of the community whose talents were obviously important in solving these kinds of problems. So we tried to be very careful to work with the technical community--technologists and researchers--and with the vendors, rather than to set ourselves up as an organization that would somehow replace them. I think that's another principle we've held to, and it's served us well ever since. We tried to keep a focus on the technical issues rather than get stretched out into investigations and law enforcement, and all the other things that go along with this problem. We were concerned that we would spread ourselves too thin. Very early on, we decided that what the country really needed was a distributed approach, that there need to be multiple response teams, each one focused on a particular community of interest. If you look across the whole Internet and you think about all of the different organizations that are connected, you realize that government or military organizations are very different from universities, from for-profit corporations, and so on.

BP: So your principle was that the incident response teams had to map to these domains?

RP: Right. To give effective response to a particular domain, you really have to understand it, its culture, and the laws and regulations that govern how it operates...

BP: Now, is that, in fact, what distinguishes the various incident response teams?

RP: To a large extent, yes. Many of the teams are focused on individual organizations. Penn State has an incident response team for just Penn State. Boeing has its own team too. Some of them have a national focus; there's an Australian team, a Japanese team, a German team, and one in the Netherlands. We had originally thought that some of them might be focused on a particular class of technology--one for Unix systems, one for IBM mainframes, and so on--but that never really happened. What's happened instead, however, is that the product vendors are now participating. So, for example, Sun Microsystems and Cisco each have response teams that address their internal corporate security needs, plus they each have a team that addresses security vulnerabilities reported in their own products.

Internet security in the news

BP: This whole issue has been raised into the public consciousness. You see it increasingly in TV commercials: the IBM commercial about the hackers who break into an organization and find the payroll records comes to mind. It seems that the world is coming around to something you have been concerned about for many years.

RP: Yes, in a big way, especially in the last couple of years. From our perspective, that's kind of heartening. Others are now beginning to look at these problems as something they really need to pay attention to.

BP: How do you account for the growing interest in these kinds of problems?

RP: I think there are several factors that are driving the increased awareness. One is organizations like ours, who have been out trying to raise awareness of such issues. In fact, raising awareness was one of the things initially identified in our charter with DARPA. DARPA wanted us to spend a significant amount of time getting this problem out from under the table and putting it out where people could see it.

BP: This relates to your education and training activities?

RP: Right. And at all of the conferences we go to, we give presentations and birds-of-a-feather sessions. We've worked to cultivate a relationship with the media. Another thing, I think, is that, as people have begun viewing the Internet as more than just a hobby, but as essential to the way they conduct business, they've suddenly begun to realize that they are putting real business assets online. That's increased the awareness of security risks dramatically. Within the Department of Defense (DoD), about 95% of the network traffic goes over the Internet. The DoD has become very dependent on the Internet and has become very aware of what the risks are, through both a series of attacks that have been launched at them from the outside, as well as their own internal exercises to gauge their own vulnerability. Over time, the importance of these issues has been going up the management chain, so now the most senior levels of the DoD and other parts of the government really recognize this as a problem. I think there are two other things that are going to drive this. One is the potential for liability, the other is the insurance industry. We haven't seen a lot of major lawsuits yet...

Within the Department of Defense (DoD), about 95% of the network traffic goes over the Internet. The DoD has become very dependent on the Internet and has become very aware of what the risks are.

BP: So the insurance industry has a lot of interest in network security.

RP: Yes. The insurance companies--and we've talked to several of them--are now beginning to see this area of insuring intangible information assets, or the integrity of the computer operation, as a new area of business. Historically, they spent a lot of time insuring equipment, but not so much time insuring operations or data. They're now beginning to see this as a new opportunity. As they begin to get involved in this area, they start to set rates based on some assessment of risk. I think that management is going to start to pay attention to security because their business insurance is going to cost more if they choose to ignore it. I think that will motivate people as well.

Intruders and how they have changed

BP: Tell me about the intruders that you've seen. How have they changed? First, let's start with the original guy who made the worm. What happened to him?

RP: A jury found him guilty of violating the 1986 Computer Fraud and Abuse Act and sentenced him to three years of probation, 400 hours of community service, a fine of $10,050, and the costs of his supervision.

BP: Was he just fooling around, or were his intentions more sinister?

RP: I think he was just experimenting. There's lots of speculation on what he was trying to do. He certainly wrote a program that was designed to spread itself around the Internet by exploiting vulnerabilities. I think some speculation is that it was a research project that got out of hand. I also think that he may have wanted to prove a point, that he had been telling people for some time that this network was very

vulnerable, and the worm was a demonstration of how vulnerable it was. Also, it's believed that the impact was as broad and as noticeable as it was because of a bug in the code, not by design. So his original intent may have been to demonstrate something, but because of this bug, the program propagated itself all over the place.

BP: In a way, we should probably be indebted to him that his intentions were as benign as they were, right?

RP: I hate to say that anybody ought to be able to break the law for any reason. But it certainly was a wake-up call, and it got a lot of people's attention. Some of the other incidents that occurred over the years have done the same--in some cases, because they were large incidents with widespread impact, and in other cases because they caught the attention of the press.

BP: What percentage of incidents can be attributed just to smart kids who are fooling around?

RP: I think a lot of it is that. My guess is that 80% of what we see, if not more, is from the "recreational hacker," people who are out there testing their skills, trying to explore networks with no real malicious intent in mind other than seeing how far they can go and what they can get away with. The problem is the side effects that can occur that can accidentally cause a lot of damage. There are some trends in the incidents that we've seen, in terms of skill level and sophistication of the hackers. Over time, the attacks have become more and more sophisticated. The hackers understand more and more about operating systems, networking software, and network topology.

Over time, attacks have become more and more sophisticated. The hackers understand more and more about operating systems, networking software, and network topology.

They have really investigated the network protocols and have taken advantage of the flaws in them; they've learned how to "chain" vulnerabilities together. If they can't get from where they are to where they want to be in one step, they've learned how to take multiple steps to get there. And they've automated their attacks. When we started, we saw a lot of people typing in commands on the keyboard; now, almost all the time, we see things written up in scripts or pieces of software that execute the attacks. This gives hackers a lot more leverage. We've seen more and more use of network scanners, which will basically scan through a whole series of systems on the Internet looking for those that are vulnerable. We've seen more use of network sniffers--planting a piece of software in some system on the network where there's a lot of traffic going by to capture information. Even today, a lot of the logons happening on networks are still going through clear text with no encryption. The system ID, account name, and password are all traveling through clear, plain text. Sniffers can just grab that information; people are giving away the keys to their systems. On the other hand, given the widespread ability of these tools ...

BP: Because they're all available on the Net?

RP: Right. You can get a nice burglar's toolkit right off the network without working very hard at all. A few Net searches, and that's that.

BP: And that's legal?

RP: Right. Nothing illegal about it. But because of that, there's a whole new generation of not-so-sophisticated hackers who can simply use the tools and see how far they can go just by pressing buttons. And that just raises the annoyance level. We now get about 35 reports each daythat are real incidents, because so many people are out there probing systems, and so many more people are out there not protected. So these guys are being successful.

BP: A lot of this seems to me to be like the natural escalation of a game--like chess or any other game--in which over time, the players develop expertise. It seems that this escalation has no limits. As you get better at your responses, the hackers get better at thwarting your responses.

RP: It's going to keep escalating. I think that's exactly right. And I think what the good

guys have to do is understand that they can't depend so much on the static kinds of defenses they used in the past. Their protection strategies are going to have to be much more dynamic. They're going to have to realize that any defense they put into place is going to degrade over time, because the hackers are going to find a way to penetrate it sooner or later. And so defensive people are going to have to counter that, and be much more dynamic in their responses than they are today.

The good guys have to understand that they can't depend so much on the static kinds of defenses they used in the past. Their protection strategies are going to have to be much more dynamic.

Serious incidents

BP: Let's for a minute put aside the recreational hacker and talk about the real bad guys.

RP: Over the years we've seen much more theft of intellectual property and valuable information. Industrial espionage. People stealing things of value that they can go and make a buck with. Extortion is on the rise too. We've seen more denial-of-service attacks, especially over the last two and a half years, where it seems pretty obvious that someone is either trying to put someone else out of business or to do some serious damage to the business. A lot of these attacks are aimed at Internet service providers (ISPs) to put their systems out of commission for some number of hours or days. You can really hurt an ISP's business that way, because the customers are going to go someplace else. We have seen much more of that. Certainly there have been some big notable things in the news, like the Citibank case a couple of years ago, where a couple of million dollars was lifted out of the Citibank vaults by electronic funds transfers. All but a couple hundred thousand of that was recovered, but that's still a very big deal. Those things make the news periodically. The suspicion is that there's a lot more of this that goes on that people aren't willing to talk about, although there is little hard data to support that. We've seen attacks that could be potentially life threatening, as people have begun to attack hospitals and clinics. The case we often bring up in this regard was the one reported by New Scotland Yard several years ago, where a hacker broke into a medical facility and changed the test results on cancer tests from negative to positive, so that for several days, several women thought they had cancer when they didn't. Another case was where a hacker broke in and changed data in a CAT scan. Scans had been done that were to be used for brain surgery, and the surgery had to be postponed until the system was rebuilt. As organizations put more and more of this sensitive information online, they're putting whatever's attached to the information--like people--at risk.

As organizations put more and more sensitive information online, they're putting whatever's attached to the information--like people--at risk.

Protecting the critical infrastructure

BP: This brings us to critical infrastructure protection. Can you talk about what's going on with that, and what the SEI's role is?

RP: The Presidential Commission on Critical Infrastructure Protection was established about two years ago to assess how vulnerable the United States was to terrorist attacks.

BP: And a potential target of terrorist attacks is the critical infrastructure.

RP: Yes. The commission itself had a broad charter to look at all forms of terrorism, not just information infrastructure security ... I think their study lasted about 18 months. The report that they produced is out there on the Web.We sent them a white paper, as did many other people. It was interesting that the area that they finally focused on was primarily information security. They found that the problem was pervasive throughout the eight critical infrastructures that they looked at. They found that the information infrastructures were becoming increasingly interconnected, and that an attack against any one of them would potentially have a ripple effect on the others. And for the most part, with the exception of the financial community, the infrastructure operators really weren't all that aware of their potential vulnerability or the seriousness of it. The commission recommended that a number of things be put in place, and some of that is actually on its way in the federal government right now. One of them was the establishment of the National Infrastructure Protection Center (NIPC), which is now part of the FBI. As part of their charter, the NIPC has an incident

response and an investigation role, and also a preventive role. In fact, we're meeting with the FBI to see how we can work with them to support their effort. There is also something called the CIAO (Critical Infrastructure Assurance Office), which was established by Presidential Decision Directive 63 (PDD-63), and they are responsible for coordination of the infrastructure-assurance activities across the federal government.

BP: What else is the government doing to protect the infrastructure?

RP: Each federal agency is required to have a plan for how it is going to improve its security, and those plans are due right about now. The idea was that the federal government would take the lead in trying to mount a national effort to improve security, and one way is by improving itself. The second way is that one federal agency has been assigned for each of the critical infrastructures, to work with that industry group to help improve the information-security aspects of that industry. For example, the Department of Treasury has been appointed to lead the financial sector, the Department of Transportation has the transportation sector, the Department of Commerce has the telecommunications world, and so on. Each of the lead agencies, in conjunction with industry leaders from each of the industry groups, is tasked with putting together a strategy for how that critical infrastructure is going to work with the federal government to improve. I have yet to see any outputs from any of those efforts. I think that discussions are underway, but it's too early to tell where all of that is going to go.

BP: It sounds a bit top-heavy given the requirements of the situation.

RP: Well, my concern is that because it's been spread across so many agencies, it may start to lose focus.

BP: What sort of coordination is there?

RP: That's what the CIAO is there to do; to keep all of those efforts coordinated.

R&D and training in information security

RP: The other activity within the federal government is the proposed increase in R&D in information security. There was a significant research component to the agenda as well. And we'll have to see what Congress decides. Our participation in this so far has been working with the FBI at the NIPC. In fact, we have one activity with them now to look at ways to analyze incident data, to try to spot trends. We have been working and have had several discussions with Jeffrey Hunker, who's the director of the CIAO. He's been talking to us about R&D agendas as well as awareness and training. How can you build an understanding of this problem, but also, how can you begin to build skilled practitioners who are capable of working within these problem areas? That's one area that we're especially concerned about. Over the years, as we've talked to people on the phone, our perspective is that the average level of ability of system administrators has been declining, given the explosion in the use of the technology.

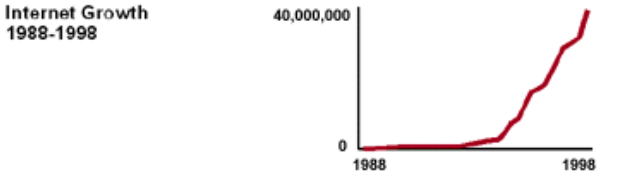
The average level of ability of system administrators has been declining, given the explosion in the use of the technology. The systems today aren't any easier to administer than the systems of 10 years ago, but we're asking people with less and less technical skill to take on that task.

BP: That's understandable.

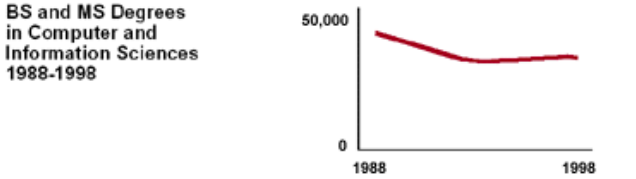
RP: A lot of people are being pressed into service as system administrators who don't really have the technical foundations.

BP: And modern systems make it a lot easier to hide the user from the implementation details of the system. You lose that deep technical expertise.

RP: I have a chart that relates that information. The top chart is the growth of the Internet, and the bottom chart is the number of people getting Bachelor's and Master's degrees in the information sciences over the same period of time.



Source: Internet Domain Survey by Network Wizards, http://nw.com/zone/WWW/



Source: Digest of Education Statistics 1997, US Office of Educational Research and Improvement, Washington DC, publisher: US Government Printing Office, Superintendent of Documents, 1997

RP: What's interesting to me is looking at this gap, which is getting wider over time. To me, that means increased vulnerability. I think that's what we're seeing out there right now. This whole area of training--not just awareness, but building real skills--I think is critically important. The systems today aren't any easier to administer than the systems of 10 years ago. But yet, we're asking people with less and less technical skill to take on that task.

Security of software products

BP: Is there now more security built into systems?

RP: Yes and no. At the architecture and design level, I think many systems are better... I hate to make this broad generalization, but I think there's more attention at the architecture and design level being paid to security than there was 10 years ago. The problem that I see is that at the implementation level--the code that's going out today is just as buggy as the code that went out 10 years ago. About 90% of the vulnerabilities we see aren't caused by errors in design, but by weak implementation. And that's not getting any better at all; in fact, I think it's getting worse. Vendors are being driven by the marketplace. Time to market is critical for them: "Get the product out the door, and if it's accepted, fix it later, and if it's not accepted, forget the product and go with something else." I think we're seeing that in many of the products.

BP: So the software problem hasn't disappeared.

RP: No. It's just gone to a different place.

What organizations can do

BP: What would you recommend for a company to do to help itself? A company that has very sensitive data in its system, for example.

RP: Oh, a couple of things. First of all, I'd encourage them to take several views of the problem. One of them is to just assume that sooner or later, they're going to have a problem, and to get ready for it. They need to have their own policies, procedures, and incident-response capabilities in house. If something happens, who are you going to call, what are you going to do? Are you going to decide to investigate or not; if so, what law-enforcement organization are you going to call in? What senior management officers are going to be notified? What authorizations do system administrators have so that they can begin to immediately investigate the problem? Does that include the ability of the system administrator to look at any piece of data on the system? All those decisions need to be made in advance, so that when something does happen, the response can be quick. Incidents tend to get worse over time, so the longer an intruder has access to a system, the more deeply and broadly the intruder can penetrate an operation.

Incidents tend to get worse over time, so the longer an intruder has access to a system, the more deeply and broadly the intruder can penetrate an operation.

You want to be able to deal with it quickly, and that means having things in place ahead of time. The next area is prevention: beginning to develop real strategies and putting things in place to help prevent these incidents before they occur. We're

currently working on a security self-evaluation method that we hope to have available to customers in a year or so. It will give them guidance on how they should think about the problem, how they should analyze their own operations ... A lot of this involves understanding what's important to them. We see a lot of people who do get interested in security who make the mistake of trying to do "all or none." Typically in most organizations, it's only a small percentage of their operation or a small percentage of their data that's critically important. We encourage people to understand what that is, and to invest a lot of time and energy there and not so much in other places. People should also get plugged into information sources like the CERT®/CC and CERT® advisories, and other things on the Net, so that they can stay current on vulnerabilities and threats. They also need an active program to fix systems when vulnerabilities are found ...

BP: Before something happens.

RP: Right ... Understanding the kinds of defensive things they can put in place--firewalls, policies, procedures--that will give them protection against at least everyday hackers, and in those critical areas, protection against people who will invest more time and energy trying to get access to that data. As part of that, we encourage people to not think of intruders in only one way--to think about different kinds of intruders who would be interested in them for different reasons. The recreational hacker who manages to stumble across you while searching the Net might be interested in you because you're somehow highly visible to them and "look juicy"; the criminal is interested in you because he or she is trying to make money by stealing information from your organization or by putting you out of business; and the terrorist wants to make a big political statement and is looking for a place to have a big, visible impact. Putting defensive strategies in place is another important thing that we think those organizations should do, and making sure that they have at least some of their staff focused on that problem, rather than having it be assigned as part of a secondary or tertiary duty, which we see happening in a lot of places.

BP: Is the SEI an example of an organization that "looks juicy" to a recreational hacker?

RP: Certainly. The CERT® /CC gets pounded on all the time. People have been trying to get into our systems for years. So we're under a sort of constant attack. I think any organization that's publicly visible, certainly any organization that's visible and claims to do security work, is an interesting target.

The future of Internet security

BP: Where do you see the Internet going in the next 10 years?

RP: Much of traditional information security is based on a model of a bounded system--a system where you have defined who has access to what, where the endpoints of the network are, and what's attached to the endpoints. For every person who uses the network, you have some way to authenticate who they are and know where they are. Traditional security depends on having this very structured network. The Internet, on the other hand, looks less and less like this every day. It has been growing at a furious pace, and I think we're going to see this network continue to grow dramatically in both size and function. Low-cost network-access devices will make access affordable to almost everyone, and almost everyone will want to use it because of the applications that the network will support. For example, there will be a blurring between the Internet and the cable networks--entertainment over the network or networking over the cable systems; they will eventually be integrated. More information will be available to more people, and there will be no need for people to identify who they are. There will be more electronic commerce, and there will be some fundamental changes in the way we do business. For example, real-estate brokers will behave very differently in a world where buyers have quick, direct access to property listings and to the sellers themselves. We will see a demand for the equivalent of cash transactions over the Internet--transactions where it's not important that the buyers and sellers authenticate themselves to each other, only that there is a fair exchange of value.

So a lot of concepts that we've used for years won't apply as the networks evolve. You're seeing more and more use of various kinds of networking techniques to make it

less and less possible to know who you're dealing with. It's currently easy with today's protocols to simply "spoof" the protocol, to disguise your point of origin and make it look like you're attacking from someplace you're really not. Wireless networking will just exacerbate that. In fact, I saw a neat little gadget last week. It's a cellular modem, about the size of a cell phone, plugged into the back of a portable PC, and there's a parallel cell phone system that's exclusively for these devices. So now you can go anywhere in the country...

BP: And put a modem on one of these.

RP: Yes. And remotely get access to anything from anywhere. So the whole concept of being able to identify who this person is, being able to track them back, is just going to be more and more difficult to do. I think that's one of the problems we will face in the future. Another one is that I expect to see more and more real full-scale network computing; more cases where the application depends on the network being in place. In fact, the application is distributed broadly across the network.

BP: The Java model.

RP: Right. And so, this whole idea of being able to say, "I can restrict my organization so that only certain people or systems have access to my computers, and have the people in my organization restricted to only having very limited access to external computers," that's beginning to go away. And with that, the whole concept of a firewall begins to disappear, because the firewall is nothing more than a filter that enforces the policies for restricting access that you build into it. And as those policies break down, the whole concept of the firewall goes with it. The problem is, we don't have anything in place to replace those concepts, and that's going to be one of the next big challenges. In this world of truly distributed computing, with things ubiquitously available, with a lot of anonymous people running around, what are the security models we will need to have in place to deal with that world, and then what technology are we going to need to support it all?

What security models will we need in a world of distributed computing, and what technology will we need to support it all?

BP: That seems like an entirely new frontier for security.

RP: And I think a lot of people are concerned about it.

BP: It sounds as if you will be very busy in the next 10 years.

RP: I don't see any end in sight. It's going to continually be a challenge to keep things updated. Again, as any sort of defenses are set up, the bad guys are going to be right there. It's going to be a footrace. One side is battling against another. But then on top of that is this whole changing technology that is going to drive an endless series of new business opportunities and new online applications that allow people to do business in new ways. And each of those changes is going to require a change in security, policy, and technology.

BP: What about the problem that so many organizations are having these days finding qualified professionals to staff their needs?

RP: There is a critical need for more people who are skilled and work in the area of information security, especially at the technical level. The other big need that we see out there is for organizations that provide training and education. If you look at the number of schools in the country that have anything that's in any significant way focused on information security ... there are only a handful, maybe five or six. Given the need for these people out there, that's nowhere near enough. The job market is just tremendous in this area for people who have strong skills in technology.

BP: Is it difficult for you to find qualified staff members?

RP: It's very difficult. And everybody who I know of working in this area is trying to increase their staff. The private sector consulting firms are trying to grow in major ways to keep up with the increased demand. I hope that the universities pay attention to that, because there's going to be an increasing need from government and industry

for trained professionals in this area. Even junior colleges, to help some of the younger folks aim in the right direction, would be a step forward.

The other issue that's going to be kind of nasty to deal with over the next few years is that, given this big increase in demand on the one hand and a lack of qualified professionals on the other, the marketplace is going to be susceptible to a bunch of snake-oil salesmen. I think we're going through a risky period right now, where some of the organizations and people who offer security products and services don't really understand the security issues or technologies, they take a surface-level look, and I don't think they know enough to know that the services that they're offering aren't sufficiently robust.

BP: And decision makers under pressure are susceptible to taking actions that are ultimately not in their best interests.

RP: I think the market has to be very wary of that for the next five years, until this balance between demand and trained professionals settles out. I'm really worried that a lot of people are going to invest a lot of money and time in the belief that they're going to implement real solutions, when essentially they're just as vulnerable as if they had done nothing.

About Richard D. Pethia

Richard D. Pethia manages the Survivable Systems Initiative at the Software Engineering Institute (SEI). The Survivable Systems Initiative is focused on improving trust in network-connected systems by: developing and transitioning system administration tools and techniques that improve the security of existing systems; exploring, developing, and transitioning software engineering practices that lead to systems with improved security characteristics; and, through the CERT Coordination Center, conducting security incident-response activities and fostering the development of incident-response infrastructures that lead to rapid correction of vulnerabilities and resolution of incidents.

Before coming to the SEI, Pethia was the director of engineering at Decision Data Computer Company in Philadelphia, Pennsylvania, where he was responsible for engineering functions (software engineering, hardware engineering, mechanical engineering, drafting, and documentation) and resources (engineering laboratories, engineering systems, and development tools) required to support new product development. The product line of this company is personal computers, workstations, printers, and high-performance telecommunications controllers. Pethia also was manager of operating systems development for Modular Computer Corporation in Fort Lauderdale, Florida. Product development focused on real-time operating systems and other system software to meet the needs of industrial and government clients in the application areas of industrial automation, process control, data acquisition, and telecommunications. Pethia has additional experience in the areas of time-sharing system development, telecommunications, and networking. He has 30 years' experience in both technical and managerial positions.

For more information

Contact Us

info@sei.cmu.edu

412-268-5800

Representing Software Architecture

NEWS AT SEI

Author

Rick Kazman

This library item is related to the following area(s) of work:

Software Architecture

This article was originally published in News at SEI on: December 1, 1998

In the previous release of *news@sei*, Mario Barbacci compared software architecture with more traditional architectural and engineering practices. In this column, I will both take the analogy a little bit farther and also begin to discriminate software architecture and architectural engineering from "traditional" forms of architecture and engineering in terms of the specific needs for representing software and system architectures.

It is first useful to consider the question: Why should one bother to design and use a software or system architecture?[1]

Fundamentally, there are three reasons why software architecture is important to large, complex, software-intensive systems [1]:

1. *Communication among stakeholders*. Software architecture represents a common high-level abstraction of a system that most if not all of the system's stakeholders can use as a basis for creating mutual understanding, forming consensus, and communicating with each other.

2. *Early design decisions*. The software architecture of a system is the earliest artifact that enables the priorities among competing concerns to be analyzed, and it is the artifact that manifests the concerns as system qualities. The tradeoffs between performance and security, between maintainability and reliability, and between the cost of the current development effort and the cost of future

Related Links

News

SATURN Conference Announces Additional Keynote, Conference Scholarships

Distinguished Speakers, Strong Technical Program Set for SATURN 2014

See more related news »

Training

Big Data - Architectures and Technologies

Documenting Software Architectures - eLearning

See more related courses »

developments are all manifested in the architecture.

3. *Transferable abstraction of a system*. Software architecture constitutes a relatively small, intellectually graspable model for how a system is structured and how its components work together—this model is transferable across systems; in particular, it can be applied to other systems exhibiting similar requirements and can promote large-scale reuse.

If we believe in these reasons, then it is immediately obvious that the way that we *represent* the software architecture is of crucial importance, because the representation will affect how well the architecture serves as a communication vehicle, how well it conveys early design decisions, and how accurately it describes the system's key abstractions. Thus, the *clarity*, *completeness*, and *precision* of the architectural representation are of key importance.

Clarity is important to support communication. Completeness and precision are needed to support analysis.

We have now introduced a new concept to the picture: analysis. Engineering representations, in any field, serve to support analysis. Structural engineers, for example, might analyze the strength and bending properties of the materials that they use to construct a building, to determine if a roof will withstand an anticipated snow load, or if the walls will crumble in an earthquake.

So too with software architecture. As architects, we need to be able to understand whether a system, as it is designed, will support the anticipated volume of user requests within some acceptable latency bounds. We need to be able to know that anticipated changes to the system can be made within reasonable time and budget constraints. And so forth. Consequently, we must understand what it is that we need to analyze before we can decide what to represent.

Software has many possible representations. In the software architecture community, these representations are known as *views* or *structures*. Some of the most common and useful are

- *conceptual/logical*: The units of this view are abstractions of the system's functional requirements. These abstractions are related by the "shares data with" relation. This view is useful for understanding the interactions among entities in the problem space, and their variation.

- *process*: This structure deals with the dynamic aspects of a running system. The units of the view are processes or threads. Relations represented by the links include "synchronizes with," "can't run without," "can't run with," "preempts," or any of several other relations dealing with process synchronization and concurrency.

- *physical*: This structure shows the mapping of software onto hardware. It is particularly of interest when dealing with distributed or parallel systems. The components are hardware entities (processors, sensors) and the links are communication pathways (typically networks). Relations between the processors are "communicates with." This view allows an engineer to reason about performance, availability, and security.

- *module*: The units are work assignments and have products (such as interface specifications, code, test plans, etc.) associated with them. They are linked together by the "is a submodule of" relation. Module structure is used to allocate a project's labor and other resources during maintenance as well as development.

- *uses*: The units of this view are procedures or modules; they are linked by the "assumes the correct presence of" relation. Uses structure is used to engineer systems for which it is easy to create extensions or subsets. One might want to do this to create different-"strength" versions of a product for different markets or to make use of an incremental-build approach to integration.

- *calls*: The units of this structure are procedures or methods; they are related by the "calls" or "invokes" relation. The relations are characterized by their interfaces. The calls structure is used to trace the flow of execution in a program.

- *data flow*: The units of this view are programs or modules; the relation is "may send data to." The links are labelled with the name of the data transmitted. The data-flow view is most useful for requirements traceability.

- *control flow*: The units of this view are programs, or modules; the relation is "activates" or "becomes active after." This view is useful for verifying the functional behavior of the system as well as for understanding timing and ordering properties. If the only mechanism for transferring control is procedure call, then this view is identical to the calls structure.

- *class*: Units of this structure are classes. The most common relation between the classes is "inherits from." This view supports reasoning about collections of similar behavior (i.e., the classes that other classes inherit from), and parameterized differences from the core, which are captured by sub-classing.

Note that we have used both terms "structure" and "view" in the above list. This is intentional. But we do not use them interchangeably. A structure defines a unique set of components and connectors. A view is simply that: a particular view of some structure or structures. An analogy can be found with relational databases. There is some information that can be found in the tables of the databases. These are the database's structures. One can then select and compose any number of views on top of this structure, to extract from the database particular information that is of importance.

Now, what is the connection between these views and structures and the analysis of software and system architectures? Quite simply, the views must carry sufficient information to support analysis. Another way of saying this is that they must be appropriately annotated to support analysis. For example, to support a performance analysis, the process view of a system must include information about process priorities, scheduling policies, queuing policies, and processing times. In addition, information about CPU speeds, network bandwidths, sensor rates, memory sizes, and the allocation of processes to hardware will be needed from the physical view to complete this analysis. At the SEI, we are currently working on a set of architectural-representation requirements—with an eye to supporting analysis—and we expect to publish this as a technical report in early 1999.

One final point: given that it is important to represent software architectures and that there is so much information to be represented, a natural question is "What tools exist to aid in this representation?" While many tools—both academic and commercial—exist, none of them is currently satisfactory in that they cannot represent even the incomplete list of views and structures that we presented here. Some partial solutions do exist however. UML (Unified Modeling Language) is currently becoming an industry standard for software representation, and there is great interest in extending UML to support architectural concepts. And commercial tools are beginning to extend themselves to incorporate architectural notions; so there is hope.

References

[1] L. Bass, P. Clements, R. Kazman, *Software Architecture in Practice*, Addison-Wesley, 1998.

1 While we say software architecture we are almost always talking about system architecture for software-intensive systems.

About the Author

Rick Kazman is a senior member of the technical staff at the SEI, where he is a technical lead in the Architecture Tradeoff Analysis Initiative. He is also an adjunct professor at the Universities of Waterloo and Toronto. His primary research interests within software engineering are software architecture, design tools, and software visualization. He is the author of over 50 papers and co-author of several books, including a book recently published by Addison-Wesley entitled *Software Architecture in Practice*. Kazman received a BA and MMath from the University of Waterloo, an MA from York University, and a PhD from Carnegie Mellon University.

The views expressed in this article are the author's only and do not represent directly or imply any official position or view of the Software Engineering Institute or Carnegie Mellon University. This article is intended to stimulate further discussion about this topic.

COTS Evaluation in the Real World

NEWS AT SEI

Author

David J. Carney

This library item is related to the following area(s) of work:

System of Systems

This article was originally published in News at SEI on: December 1, 1998

Introduction

In my last column, I covered the overall topic of evaluating commercial off-the-shelf (COTS) products. To summarize the major points that I made in that column:

- "COTS evaluation" can have various meanings. For our purposes, the intended meaning is that COTS evaluation is a process to decide whether to select one product for use in a given context.
- The evaluation process is pervasive. That is, evaluation is not restricted simply to the moment when an assessment of a product is made, but is operative at many points, e.g., when we define our selection criteria, when we perform vendor surveys or market research, and so forth.

I also suggested that there are three large-scale tasks that occur when any COTS evaluation is performed:

1. Plan the evaluation.
2. Design the evaluation instrument.
3. Apply the evaluation instrument.

In this column, I will examine two actual organizations that carry out COTS evaluation processes, to see how this abstract notion of COTS evaluation fits with real-world experiences.

The Central Idea: The Importance of Context

Before I do so, however, I must return to a thought that was touched on in the last column, and whose importance I will now stress. Central to COTS evaluation—in the particular sense of it as a process rooted in decision making—is the importance of *context*. By context I mean all of the factors and constraints (functional, technical, platform, business issues, and so forth) that exist before a COTS product is chosen. "Context" includes everything with which and against which the COTS product must harmonize, conform, and operate; it is the basis on which we develop evaluation criteria to assess the product.

This notion seems rather simple, and perhaps obvious, but it has some rather far-reaching implications. For one thing, in using COTS products in complex systems, it is difficult to continue with our traditional requirements-driven processes. COTS products are typically written to the vendor's own predictions and expectations of what will have market success. The requirements of our particular system, i.e., the one that will incorporate those products, are not known to the vendor (nor would they necessarily be of interest to the vendor even if known).

So instead of a set of hard "must haves" by which we will judge a product, we have a much more fluid collection of features, some mandatory, some strongly desirable, some merely "nice to have"; in short, this collection of features provides the *context*, the source of the criteria by which we will decide whether a given COTS product is sufficient for our needs.

Another implication is that this context can be very wide, containing both technical and business-oriented issues, and the kind of tradeoffs we will make will necessarily mix apples and grapefruit. We must therefore find ways in the evaluation process to reconcile constraints that compete along different axes of interest. For instance, we will often need to make the choice between product A, which has wonderful throughput but whose vendor seems to be moving out of this particular market niche; and product B, whose vendor we know and trust, but whose product is slower (not disastrously slow, perhaps, but certainly creaky by comparison). So should we buy from trusted vendors, but with near-obsolete products? Or should we choose a bleeding edge of technology, knowing that it comes from three guys in a basement?

Finally, since this evaluation context will in some ways replace a complete set of requirements, it is very possible that novel or unexpected features of COTS products will actually change our idea of the system as it is being built. There are numerous instances, one of which I will describe presently, in which the features of emerging products can bring about changes in the design, or even the overall architecture, of a COTS-intensive system. This is indeed a radical situation for people who consider that an architecture (or design, for those who think they are the same) must be established at the beginning and that COTS products must be found to conform to it.

These implications are quite real. If we make a significant policy decision to use pre-existing pieces (i.e., if we mandate that our systems make extensive use of COTS products), then we must relinquish our grip on the details, be willing to suffer the (sometimes chaotic) fluctuations of the marketplace, and be content to let some parts of our systems be fashioned by the commercial forces that drive that marketplace.

A Glimpse at Current Practice

We now will consider how two very different organizations deal with the problem of evaluating COTS products. One organization ("A") is a major contractor that has provided numerous systems for the government. The other organization ("B") is a private corporation in the domain of business services and financial transactions. While the two use very different approaches to COTS evaluation, we still observe our abstract process ideas underlying both. And we also learn that these two different approaches corroborate our notion that whatever the actual process used, it is driven by context.

Organization A

This organization is building a large information system for a government agency. Lest *large* be misunderstood, the system will incorporate over 60 commercial components, many of them based on such evolving technologies as Web browsers, Java-based products, and other middleware products. The system, now partially fielded, serves

several thousand users and is deployed worldwide on a variety of platforms.

The domain of this system, that of Web-based information systems using browsers, Java, and its derivative technologies, is scarcely half a decade old and is still growing and evolving with lightning speed. And as fast as the marketplace of products is changing, the marketplace of ideas is changing even faster. So competition among companies (some barely deserving of the name) is cutthroat, with product releases occurring rapidly. In addition, there is little stability even regarding which products should perform which functions (i.e., these products often lack a condition called "product integrity").

For this organization, and in the context of this system, there is a fundamental need for flexibility. Since it is known that the system makes use of evolving technologies, it is understood that new elements will be refreshed and replaced often, and that design decisions (and thus COTS product choices) are subject to frequent reconsideration. Therefore, COTS evaluation for this organization is flavored accordingly.

The context for evaluation is primarily technical: the aggregate collection of factors (e.g., interfaces, standards) that permit or prevent interoperation among components. Thus, while individual products are assessed for their individual functionalities, they are assessed as much to see if they interoperate and cooperate with the other COTS components in the system.

The following indicates how our abstract process steps are instantiated for COTS evaluation by this organization:

- *Plan the evaluation*. The planning for evaluation is generally not rigorous or exhaustive, since the expected lifetime of any specific product within the system is relatively short. Planning is therefore opportunistic. Given the number of COTS products used and the number of potential candidates, a careful, methodical approach is simply not feasible.
- *Design the evaluation instrument*. What we have abstractly termed the "evaluation instrument" is weighted with criteria about compatibility. In fact, the "evaluation instrument" is really the existing system with which the candidate product must operate.
- *Apply the evaluation instrument*. Assessing products is generally done through prototyping and through installation of the product into the existing system context.

The evaluation process itself is as flexible as possible, and serendipity is not considered a bad thing. The unexpected appearance of new capabilities (whether within a product or through more general technology trends) can trigger reassessment of decisions made earlier.

One thing we observe in this style of system development (other than that it is frightening to many people!) is the mutual influences that evaluation and system design have on each other in a COTS-intensive system. I will consider this topic in greater detail in my next column.

Organization B

This organization is a large financial institution that purchases a large number of COTS products each year. The business processes of this industry are relatively stable, and while heavily dependent on data processing, are usually not influenced by any rapidly changing technological trends.

COTS evaluation, as performed by this organization, is most often done to choose products to modernize existing capabilities. While there is interaction among all of the organization's systems, new COTS products are not perceived as components within a single large system. Hence, while compatibility with the existing platforms and infrastructure is significant, new products might be considered even if they depart from the existing technical infrastructure.

The following shows how our abstract process steps are instantiated:

Plan the evaluation. The organization has a large set of procedures about how to prepare evaluation plans, how to conduct product searches, how to conduct vendor assessments, and so forth. Strict guidelines exist that require particular planning activities depending on business importance of the product, expected cost, and similar factors.

- *Design the evaluation instrument*. Choices about evaluation criteria are strongly weighted toward business factors: vendor stability, reports about vendor's market share, and satisfaction reports from other users of the product
- *Apply the evaluation instrument*. There is little prototyping. Often the actual assessment of product capabilities is limited to the vendor's presentation. The hands-on technical evaluation done in house is largely focused on platform compatibility with existing systems in the organization.

Last Thoughts

The ways that these two organizations perform COTS evaluation are radically different, which should not be surprising given how different their contexts are. Organization A's evaluation plans are necessarily opportunistic, while B's are rooted in method. Organization B's "evaluation instrument," especially in its choices about evaluation criteria, is weighted toward business factors, while A's are weighted toward technical compatibility. And when actually assessing products, Organization A does extensive prototyping while B does comparatively little in-house product examination.

These differences reflect very different circumstances, and in no way imply rightness or wrongness of either instantiated process. But both are examples of how a commercial bias will affect the evaluation process, and ultimately the entire process of system construction.

In the next column, I will focus on the kind of evaluation process found in Organization A, since it is here that we can observe the point noted above, namely, that there is a very strong mutual influence between designing a COTS-based system and evaluating candidate products for that system. Stay tuned.

About the Author

David Carney is a member of the technical staff in the Dynamic Systems Program at the SEI. Before coming to the SEI, he was on the staff of the Institute for Defense Analysis in Alexandria, Va., where he worked with the Software Technology for Adaptable, Reliable Systems program and with the NATO Special Working Group on Ada Programming Support Environment. Before that, he was employed at Intermetrics, Inc., where he worked on the Ada Integrated Environment project.

Distance Learning Grows by Degrees

NEWS AT SEI

Author

Scott R. Tilley (Florida Institute of Technoloy)

This article was originally published in News at SEI on: December 1, 1998

Distance learning is an emerging phenomenon that promises to alter fundamentally the nature of traditional education and training. As more universities begin to offer online versions of their courses, asynchronous study and synchronous lectures consisting of streaming multimedia will become more prevalent. Much of the growth in the distance learning industry can be attributed to the increasingly pervasive nature of the Net and the Web, and the collaborative infrastructure provided by net-centric computing.

In the previous *Net Effects* column, I talked about some of the infrastructure aspects of net-centric computing (NCC). One of the most important infrastructure issues is Net access, especially for consumers connecting from their homes or home offices. Without inexpensive high-speed access options, NCC will be limited to institutions such as business and universities that have dedicated connections to the Internet backbone or an internal Intranet.

I've now had the opportunity to use three different types of digital Net access technologies. These consumer-oriented high-speed connections are becoming widely available at relatively low cost. The first technology I looked at was a 56K modem. My experiences of just over a year ago with a 3COM x2-based modem are documented in the Volume 1, Number 20 issue of *SIGPC*, "56K Modems" (http://www.srtilley.com/SIGPC/V1N20.htm).

The second technology I tried was a Digital Subscriber Line (DSL). In early June, I had US West's "Megaline" DSL service installed in my home office. DSL has the advantage of offering relatively high-speed connections using existing telephone wiring and allowing the simultaneous use of the same line for either voice or fax traffic and the Net connection. My experiences with DSL are documented in Volume 1, Issue 2 of *SEI Interactive*, "Life in the Digital Subscriber Lane" (http://www.sei.cmu.edu/interactive/Columns/Net_Effects/1998/September/

Net_Effects.sept98.htm).

More recently, I've been using a third technology: a cable modem. In September, I had Charter Communications' "Pipeline" cable modem service installed in my new home office. Cable modems have the advantage of offering high-speed connections using existing television cabling. My experiences with cable modems are documented in the Volume 2, Number 7 issue of *SIGPC*, "Cooking With Cable" (http://www.srtilley.com/SIGPC/V2/N7.htm).

These three types of consumer-oriented Net access technologies all have different characteristics. For example, their connection speeds differ greatly from one another, from 53K at the low end using a 56K modem to 10M at the high end using a cable modem. However, they all can play a similar role: aiding distance learning.

The changing nature of distance learning

The notion of distance learning is certainly not new. Correspondence courses have existed for a long time. CD-ROM–based training material has been used in corporate settings for several years. Computer-based training packages have also been used for a while, for example in children's educational programs. All of these techniques are types of independent learning: The material is experienced by the student without any live interaction with the course instructor.

What is new is how distance learning can now be experienced. Using the Net and the Web, course material can be delivered to users in streaming multimedia format. The high-speed Net access methods I referred to above make this form of education possible for people who are geographically removed from the actual lecture. The difference between this type of distance learning and computer-based education of a few years ago is that the students can interact with the instructor in a synchronous manner. This means that the lecture can be delivered in audio and video format from one location, perhaps from a university, and the students can be dispersed literally all over the world.

This model of distance learning is growing in popularity, in part because of the changing nature of our society. There are several examples of institutions that are flourishing using this type of distance learning. For instance, the University of Phoenix has been offering accredited courses to students all over the world for several years now. It is one of the fastest growing educational institutions in the country. Its primary targets are working professionals looking to upgrade their education, often as part-time students. They can take all their courses online, using a Web browser in their homes.

Other universities have adopted a more middle ground. For example, Nova Southeastern University provides much of its curriculum online. Students still meet two or three times a semester for extended weekends in Ft. Lauderdale. Such "face time" is still an important part of the educational process.

Now that such respected universities such as Carnegie Mellon and Stanford University have begun offering courses online as well, many other institutions are looking at their own online curricula. In many respects, they have to; if they don't begin to offer prospective students distance learning alternatives, other institutions will. The notion of "making yourself obsolete, before your competitor does" has been successfully adopted in other disciplines (for example, with Intel in microprocessors); there is no reason to believe it isn't equally applicable to mainstream education.

Effects on mainstream education

As distance learning changes, so must mainstream educational institutions. Several recent articles (for example, "A Different Course", by Robert Cwiklik, *Wall Street Journal*, November 16, 1998), have stated that the current university structure is perhaps one of the institutions likely to be most affected by this new mode of distance learning. Online delivery of course material promises to fundamentally alter the face of the education community and training industry.

For example, universities that lack the faculty to teach a particular course might choose to license course content from another university. As the professor lectures in one

location, students in another location can also participate. This could lead to the development of "star" professors who become internationally known for their course material and delivery. This type of instructional outsourcing could have profound effects on the nature of departments in universities. It may create tiers of institutions wherein one university develops course content but another delivers it, perhaps even to a third university.

Distance learning may also have a profound effect on students. No longer will they have to move to a new city to attend the university of their choice. Lectures may be experienced synchronously, via streaming multimedia, or asynchronously at students' leisure. Having students located off campus, even for courses, would greatly change the nature of a university. Resource planning, tuition structure, curriculum content, faculty hiring, and research programs would all be affected. This in turn would affect the learning experience for students, for instance in interaction with their teachers.

The distinction between education and training is always a contentious issue in universities. With distance learning, the problem is exacerbated. Private industry can (and already does) offer courses that are particularly appealing to high-tech employers and employees alike. The courses can be taken at nearly any time online, and the content is often more closely aligned with the real-world needs of working professionals. For universities, there is a need to examine distance learning for mainstream education. Both types of learning, education, and training, are made possible by the collaborative infrastructure provided by net-centric computing.

The role of NCC

Without some of the technologies provided by NCC, distance learning as described here would not be possible. Net access, geographically distributed collaboration, and executable content are important aspects of distance learning.

High-speed Net access enables the delivery of bandwidth-intensive multimedia content to the home or small office. As technologies such as DSL and cable modems become more widely available, their adoption rate will drive the market for distance learning. For example, there is no reason why telecommuting could not be combined with professional development to leverage the investment employers have made in setting up home offices for their workers.

Eliminating geographical distances between points on the network, and hence fostering collaboration among people, is just one way NCC can be used to improve the distance learning experience. Research results from computer-supported collaborative work can be used in conjunction with traditional educational techniques to make the remoteness of distance learning less of an issue. In fact, NCC offers new ways of interacting. For example, chat rooms are commonly set up between lecturers and students to act as proxy person-to-person meetings. Dynamic chat rooms can be created for a few students to discuss privately specific issues related to the material. This type of public/private switching is not possible in a traditional learning environment without interrupting the flow of the class.

Executable content across the network means that exercises and examples can be implemented within the ubiquitous user interface of today, the Web browser. This can be accomplished through portable applications written in Java. It can also be accomplished by using thin clients, with the application running on offsite servers. The content delivery can go in both directions: students can submit their assignments to their instructors or demonstrate their projects to their classmates across the network.

The role of NCC in the spread of distance learning is significant. There is little doubt that both traditional educational institutions and the private sector will adopt some aspects of distance learning as the infrastructure for (a)synchronous course delivery improves. For both instructors and students, distance learning will continue to grow by degrees.

About this column

The focus of the *Net Effects* column is the impact of net-centric computing on a wide variety of issues, including computer science, information technology (IT), and software engineering.

About the author

Scott Tilley is a visiting scientist with the Software Engineering Institute at Carnegie Mellon University, an assistant professor in the Department of Computer Science at the University of California, Riverside, and principal with *S.R. Tilley & Associates*, a strategic and tactical information technology consulting boutique. He can be reached at stilley@sei.cmu.edu.

The views expressed in this article are the author's only and do not represent directly or imply any official position or view of the Software Engineering Institute or Carnegie Mellon University. This article is intended to stimulate further discussion about this topic.

Making Team Plans

NEWS AT SEI

Author

Watts S. Humphrey

This library item is related to the following area(s) of work:

System of Systems

This article was originally published in News at SEI on: December 1, 1998

At the team kick-off meeting, management told the engineers that the company critically needed their new product in nine months. This group was introducing the Team Software Process^{SM} (TSP^{SM}), and I had convinced management that the team should make a development plan before they decided on the schedule. Management had agreed, and we scheduled a meeting for two days later to review the engineers' plan. Now, the 12-engineer team was assembled and ready to make their plan. They had a lot of questions.

- How do they make a plan when they don't know the requirements?
- How detailed should they make the plan, and how much of the project should they cover?
- Suppose the plan doesn't finish in nine months; what do they do then?

And finally,

- Since they knew so little about the product, could any plan they made now be useful?

These questions are the subject of this column.

How Do They Make a Plan When They Don't Know the Requirements?

The team was very concerned about the vague state of the requirements. While a couple of the engineers had a general idea of what the product was to do, they did not

Related Links

Training

Migrating Legacy Systems to SOA Environments - eLearning

Service-Oriented Architecture: Best Practices for Successful Adoption

See more related courses »

see how they could make a realistic plan without much more detail. I was coaching this team and pointed out that they could make an accurate plan right after final product delivery. Their plan would be most accurate then, but it would be least useful. On the other hand, they most needed a plan at the beginning of the project, but it would necessarily be least accurate. So, while they did not yet know very much about the product, they agreed to make the best plan they could produce.

Plan Accuracy

Clearly, the more you know about a product's requirements and design, the more likely you are to make a good plan. Thus, plans will always be least accurate at the beginning of a project, and they should get progressively more accurate as the work progresses. This suggests three things. First, you must plan, even if you don't know very much about the job. Second, you should recognize that the initial plans will be the least accurate. And third, you need to continually remake the plans as you learn more about the work.

How Detailed Should They Make the Plan, and How Much of the Project Should They Cover?

Planning is much like designing a product. It is a good idea to start with an overall architecture or process and then to lay out the entire structure. What development tasks are required, in what order, and how long will they take? Until you have an overall framework, a detailed plan could address the wrong tasks or focus too much effort in the wrong places.

For example, I was assigned to a project some years ago. There had been many small schedule changes, but everybody thought the project was on schedule. Nobody had ever produced an overall plan. However, when we did, we found that testing time had been dangerously reduced. The project was in serious trouble.

Without an overall plan, it is hard to see the cumulative impact of many small schedule slips. They all add up, however, and without an overall perspective, the latter phases will invariably be squeezed. So, while detailed plans are essential, they must be made in the context of an overall plan that runs from the start date all the way to the final product delivery. Therefore, the first step must be to make an overall plan.

Start With the Process, Then List the Products and Make the Estimate

Once the engineers agreed to make an overall plan, they had to decide on what development process to use. By starting with the organization's overall process framework, they defined their specific project process in less than an hour.

Next, they defined the products to be produced by each process phase. They estimated the sizes of the requirements and design documents and postulated an overall product structure. They judged what components would be required and how big each component was likely to be. Each engineer contributed to these discussions, and they compared this job with others they had worked on. It was surprising how much relevant information the 12 of them had.

Next, the team had to figure out the effort required to develop each of these products. Again, every engineer contributed his or her views. In some cases, they had real data for similar jobs. In other cases, they made overall judgments based on general recollections. In the end, they came up with estimates for every product. While some of these estimates were guesses, they were informed guesses made by experienced engineers who had previously done similar work.

Make the Schedule

The last overall planning step was to produce the schedule. Here, the engineers estimated how many hours they each had available for the project each week. Since many had prior obligations that would continue, they allowed time for this other work as well. When they were done, they had an estimate of the total hours the entire team would have available for each project week. Then they spread the work over these hours to produce the schedule.

By this time, the engineers had a pretty good idea of how big the job was. Thus, they were not surprised that the project would take much longer than the 9 months that

management wanted. The full schedule actually turned out to be 18 months. At this point, the team had defined the complete process that they would use, produced a product list, made product-size estimates, and generated an overall plan—all in one afternoon. While they were still concerned about the plan's accuracy, they knew this was a big job, and there was no chance they could do the work in 9 months. They also had a lot of data to back up their 18-month schedule.

Next Came the Detailed Plan

The next step was to look at the work that lay immediately ahead. On the morning of the second day, the team made a detailed plan for the requirements phase. First, they examined the requirements process and broke it into the steps needed to produce, review, and correct each requirements product. To make sure their detailed plans fit into the overall plan, they started with the overall estimates and then estimated the engineering hours for each step. They then named an engineer for each task, and each engineer then used the same overall planning data as the starting point for a personal plan for the immediate next phase.

When the team put these plans together, the result was a shock. The combined detailed plans took much longer than the top-down plan for the same work. How could this be? The same engineers had made the plan and they had used the same product list, size estimates, and development rates.

The problem was *unbalanced workload*. The lead engineers were involved in every step of the work, and the less experienced engineers often had little to do. While the lead engineers could likely produce the best products and everyone felt that they should participate in every product review, this made them a serious bottleneck.

After some discussion, the team agreed to unload much of the lead engineers' work. By balancing the workload, the less experienced engineers got much more to do and the lead engineers concentrated on the most critical parts of the job. The final balanced plan produced the same schedule as the overall plan, and the team now felt they had a sound basis for doing the work. At this point, it was noon of the second day, and the team had all afternoon to assess project risks and to prepare a presentation for the management meeting.

What Happened

When the team presented their plan the next morning, management was impressed with the plan, but unhappy with the schedule. They really did need the product in 9 months, but, after considerable discussion, they were convinced that the 18-month schedule was the best that the team could do.

The team followed this plan in doing the job. The requirements phase took several weeks longer than planned and the design phase also took a little longer. But, the team stuck to their guns and did a quality job. As a consequence, there were fewer late requirements and design changes, implementation took less time than planned, and testing took practically no time at all.

In the end, the team finished the job 6 weeks ahead of the original 18-month schedule. Because of the well-thought-out design and the high product quality, marketing was able to contain the customer problem, and the product was a success.

Teamwork

Teams have a great deal of knowledge and experience, and when they are all involved in producing their own plans, they will invariably do a first-class job. After all, they will do the work, they have the most at stake, and they will derive the most benefit from having a realistic plan. With a detailed plan, teams know precisely how to do the work, and they feel obligated to finish on the dates to which they committed.

Closing Comments

First, early plans are invariably less accurate than those made later. The reason is that engineers often overlook tasks, they don't allow enough time to clear up requirements problems, and they assume that they will work full time on the job. Also, in many organizations, management fails to protect their teams from the normal turmoil and disruption of a running business. Thus, when you consider all the pressures in working

software organizations, the early team plans are almost always aggressive. Thus, even if an earlier date is critically important, it is invariably a mistake to cut these initial plans. If the problem is severe, the team should make a new plan with different resource assumptions or work content. The best approach, however, is to wait until the end of the requirements phase to replan. Then everyone will better understand the work, and they can make a more accurate plan.

Second, there are lots of estimating tools and methods. While I am partial to the PROBE method described in one of my books, estimating is a largely intuitive process (1). So, use whatever methods help your intuition. However, do not rely on some magic tool to produce the plan. While the detailed printout may look impressive, plans are only as good as the thought that went into them. Remember that the principal benefits of planning are the engineers' shared knowledge of how to do the work and the team's commitment to the plan. Use whatever tools and methods you have available to help make the plan and to check your results, but use these tools only to support your planning, not to replace it.

Third, to help you work efficiently and to coordinate your work with your teammates, you need a detailed plan for the work immediately ahead. While you can rarely produce a detailed plan for an entire development job, you should start with an overall plan and then produce a detailed plan for the phase you are about to start.

Fourth, when management is unhappy with your team's plan, don't change it without making a new plan. When you do, however, make sure you get different resource and work-content assumptions. Without changes in their planning assumptions, teams invariably think of previously overlooked tasks and end up with a longer schedule.

Finally, remember: if you cannot plan accurately, plan often. Plans are only as good as the knowledge on which they are based. As you gain new knowledge, produce new plans. As long as the previous plan is useful, however, don't bother making a new plan. But, the moment the plan ceases to provide helpful guidance, make a new plan.

The Commercial

While the methods I have described are not complex, they are not obvious. That is the purpose of the Team Software Process that we have developed at the SEI. It provides the guidance that teams need to follow these methods on the job. The catch, however, is that to use the TSP, engineers need to be trained in the Personal Software Process℠ (PSP℠), and their management needs overall training and guidance on how to lead and guide TSP teams.

Acknowledgements

First, I would like to thank Walden Mathews for asking some perceptive questions about planning. The answers to his questions provided the basis for this column. Also, in writing papers and columns, I make a practice of asking associates to review early drafts. For this column, I particularly appreciate the helpful suggestions of John Goodenough, Mark Paulk, Bill Peterson, Marsha Pomeroy-Huff, and Dave Zubrow.

In Closing, An Invitation to Readers

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

Thanks for your attention and please stay tuned in.

Watts S. Humphrey
watts@sei.cmu.edu

1. Humphrey, Watts S. *A Discipline for Software Engineering*. New York: Addison-Wesley, 1995.

About the Author

Watts S. Humphrey founded the Software Process Program at the SEI. He is a fellow of

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

The views expressed in this article are the author's only and do not represent directly or imply any official position or view of the Software Engineering Institute or Carnegie Mellon University. This article is intended to stimulate further discussion about this topic.

What Messages Are You Sending to Vendors?

NEWS AT SEI

Author

Shawn Hernan

This article was originally published in News at SEI on: December 1, 1998

The previous issue of this column covered today's strategic reliance on technology and the fundamental business need of all organizations for security and privacy. We contrasted this reliance with the lack of security offered in the infrastructure and supporting services upon which the technology is based. The message I hope you were left with was your need to use security as a discriminating factor in system and software procurement. But just purchasing a product with security features is not the end of the story; just as important is how you deploy that product for use. I'm joined in this release by Shawn Hernan of the CERTÒ Operations group. We'll write about how your interactions with a vendor can affect a vendor's decisions on the security of configurations for better or for worse!

Why on earth would a vendor do that?

One aspect of the work that we carry out at the CERTÒ Coordination Center is vulnerability handling: reviewing reports of security flaws, working with vendors and domain experts to identify patches and workarounds, and issuing advisories to the community at large when appropriate. As a result, we constantly receive reports of apparent security weaknesses in protocols, in software, and in product configurations. The cause of the flaws identified can vary widely. Some are the result of common or naive programming errors, while others point to far more subtle issues. On some occasions, the cause of a security weakness is immediately obvious. On other occasions, we're left scratching our heads and saying, "Why on earth did the vendor do that?" (On closer inspection, the answer is often surprising, but not always immediately obvious.) On more than one occasion, the answer has been "That's what the customer asked for." It is hard to imagine that a customer would request a security weakness; but as we'll show, it happens more often than you might think.

Increasing awareness and customer demand

As a result of increased awareness either from security organizations or their own adverse experiences, organizations have begun to seek improved security. But simply requesting security features does not result in the delivery of a product that will be secure out of the box. Systems are shipped with insecure defaults and with network services enabled by default (regardless of what percentage of the services a site wants or plans to use). This contrasts with the model espoused by many security experts¾deploy systems with secure defaults and disabled network services, then only enable the functionality that is needed.

Too many sites don't take the time to address security issues once products are deployed, even if security was requested in the procurement stage. After delivery, the pressure is on to get the product installed and interoperating with the existing technologies. Remembering to turn off unnecessary services that pose a security risk or enabling security features are often items that don't make it to the top of a system administrator's to-do list. As a result, deployed products interoperate but pose a security risk (even though they may have security features, these features are not enabled).

Primarily, organizations are using technology as a result of a business need. They want security but they don't want it to affect their ability to conduct business effectively and efficiently. So in addition to requesting security, they continue to demand high-performance products that will operate in their heterogeneous environments. The resulting message sent to the vendors is

> *Provide up-to-date technology with increasedfunctionality and good performance, security features, and default interoperability.*

On the surface that sounds like a reasonable enough message to send.

The vendor dilemma

Consider the software marketplace from a vendor's perspective. Vendors have a whole range of factors that they must consider when developing a software product.

The biggest driver for most organizations is new software sales, but other major factors include

- keeping pace with technology changes
- keeping pace with competitors' functionality
- providing interoperability among a range of products
- broadening the range of supported platforms
- providing new functionality to meet changing business needs

Clearly vendors have quite a dilemma when choosing where to apply their available resources.

Security in its own right is notably absent from the list of major factors for the majority of vendors. Rather, it is considered as a facet in one of the factors listed. Only customer pressure for increased security features can make this change. Unfortunately, as far as security is concerned, the vendors continue to receive what they would perceive as contradictory messages from customers: Provide products with default security features. Provide products with default interoperability or make this product secure—turn off the default security to make this product easier to use. As a result, the vendors provide products with security features, but to address the interoperability demand, the features are rarely enabled by default. Just recently one vendor's experiences of trying to implement improved security was brought to our attention.

This vendor's story reinforces the general vendors' dilemma.

One vendor's experience

One email product vendor has been among the market leaders in implementing security features in its products. The vendor ships both email servers and email clients, and was among the first to add a particular kind of secure authentication to either client or server. Because it was among the first vendors to add the secure authentication scheme to its products, there was a concern about interoperability: Would its email client be able to work with other vendors' email servers, and viceversa? Would the secure authentication scheme prevent interoperation with other vendors' products?

Complicating matters was the fact that the email protocol did not provide for explicit failure messages when an authentication attempt failed. That is, the client was unable to tell if the authentication attempt failed because the password was incorrect or because the server did not support the same authentication scheme. Possible options if the client received a failure message:

- Ask the user for the password again, assuming it was incorrect the first time.
- Try a less secure but more widely implemented authentication scheme, namely plaintext passwords.

In other words, the vendor was faced with a tradeoff between interoperability and security by default.

The vendor chose security by default and started to ship the client so that the default behavior was to stick with the secure authentication scheme but give the end user a way to configure the client to use a less secure authentication scheme. The effect of this security-conscious choice was that the client would work only with a server from the same vendor, until other vendors implemented the same authentication scheme.

The vendor provided documentation with the product to allow an end user to configure the product to work with other vendors' servers. So the issues of security and interoperability were addressed, but security was primary.

Although the end user could configure the product to work with other vendors' servers, the vendor received more than 280 trouble reports from sites that thought the client was broken or that simply didn't want to reconfigure the client. The customers wanted interoperability by default. This market pressure forced the vendor to choose a different set of defaults—the product will now try less secure authentication schemes if the more secure scheme fails. Thus, if a user makes an error in typing a password, the client will try the same incorrect password using all of the authentication schemes, including plain text. This means that if the user makes a typo in entering a password, the slightly incorrect password is sent on the network in plain text. More importantly, if an intruder is able to convince a user to establish a connection to a mail server of the intruder's choice, the intruder can recover the user's password. Thus a consequence of the customers' demands for default interoperability was that they obtained a less secure product.

Having changed the default configuration of the product, we would expect that the vendor would have received trouble reports from other customers complaining about the less secure configuration. But they received only one such report. The message sent to this vendor was loud and clear: Default interoperability is more important than default security.

A failed standard?

The need for more secure out-of-the-box defaults is well recognized, so much so that in 1996 The Open Group (recognized in standards setting for open systems) produced the X/OpenÒ Baseline Security Services (XBSS) standard (http://www.opengroup.org/prods/xs.htm). The standard includes a base set of security-related functionality and a standard "safe" configuration on delivery. The standard looks promising because it is consistent with other security standards, yet its

focus is more specific and business oriented than other standards such as the Orange Book, which characterizes criteria for Department of Defense systems for secure computing architectures across a broad scope from security policy, accountability, and assurance to documentation. It includes sensible secure defaults such as

- passwords on all accounts by default and changed during installation
- auditing on by default
- access control lists for users and groups
- owner-only access to objects by default

However, we do not know of any out-of-the-box products that conform to XBSS. If the need for such a standard is recognized and vendors have the capability to implement the standard in their products, then why isn't it being adopted? We suspect that the answer is a lack of market demand for security in relation to the demand for default interoperability.

Message received and understood?

When asked, most people will claim that they want security features in their products. But when confronted with the choice between security and other features, security often comes up short. How can vendors provide security when customers complain about secure defaults at the same time that they are demanding security? This is a bit like specifying that your new house must have a security system with motion detectors inside and detectors on all windows and doors; but you never turn it on, or you never change the code on the keypad from the vendor-supplied default of 1234!

When commenting on his company's experiences, one person from the vendor organization in our story said, "The bottom line with security is that people want it, they just don't want to have to know about it." Probably most people would agree—that is exactly what they want, although invisible security may bring its own drawbacks. But in today's market, with so many vendors competing on so many fronts, it is not possible to provide invisible but effective security without a considerable investment. As the old adage goes, you get what you pay for. Unless the community is willing to pay for security, vendors won't invest in providing it.

In the meantime, be careful what message you send to your vendors, as you may get what you ask for, and it may not be what you expected. We only hope that the next time we're analyzing a security vulnerability and are left scratching our heads and saying, "Why on earth did the vendor do that?" it won't be because the customer asked for it.

About the authors

Moira J. West-Brown is a senior member of the technical staff within the CERTÒ Coordination Center, based at the SEI, where she leads a group responsible for facilitating and assisting the formation of new computer security incident response teams (CSIRTs) around the globe.

Before coming to the CERTÒ/CC in 1991, West-Brown had extensive experience in system administration, software development and user support/liaison, gained at a variety of companies ranging from academic institutions and industrial software consultancies to government-funded research programs. She is an active figure in the international CSIRT community and has developed a variety of tutorial and workshop materials focusing mainly on operational and collaborative CSIRT issues. She was elected to the Forum of Incident Response and Security Teams Steering Committee in 1995 and is currently the Steering Committee Chair. She holds a first-class bachelor's degree in computational science from the University of Hull, UK.

Shawn Hernan is a member of the technical staff at the CERT Coordination Center where he leads the vulnerability handling group. Prior to joining CERT, Shawn worked

for the Systems and Networks division of the University of Pittsburgh for seven years where he developed databases and network applications, and shared in the system administration of the centralized computing facilities and the large campus network.