Carnegie Mellon
Software Engineering Institute

news @ sei interactive

Volume 5 | Number 2 | Second Quarter 2002

http://www.interactive.sei.cmu.edu

news @ sei interactive

From the Director

Software is the engine of innovation in our Internet-connected world.1 Research yields new ideas that software transforms into new products. Unlike traditional industries such as the automotive industry, software requires no factories for manufacturing, no costly distribution system, and hence no large infrastructure investment. But it does require skilled software engineers.

I am concerned that, because we are not continuously improving our software engineering skills, our economy and our Internet infrastructure are at great risk. We do not use proven, disciplined engineering methods for software development. For example, we still usually build programs line by line rather than assembling them from reusable parts. We are not creating sufficient incentives for the software industry to use proven best design practices that minimize the introduction of software defects. Because few universities recognize software engineering as an important field of study and research, insufficient numbers of qualified software engineers are entering the work force. The marketplace does not demand well-engineered software. Simply put, we—all of us—are too complacent. We do not demand better software because we do not really expect software to work.

Defects in products that are accessible to the Internet render them vulnerable to cyber attacks. The Carnegie Mellon University CERT Coordination Center® (CERT/CC) documented more than 2,500 commercial-product vulnerabilities last year and determined that more than 95% of the 53,000 unique cyber incidents it investigated were a direct result of intruders exploiting such vulnerabilities.2 Because only a small number of root causes underlies the enormous number of vulnerabilities seen in commercial software, these defects, and hence most cyber attacks, could be prevented if vendors used the proven best design techniques of software engineering. The best way to ensure the security of our software is to design software in a way that does not allow defects into software in the first place.

Commonly used software-development practices also result in lost productivity, as time and money are wasted on rework. The state of software development today is similar to the state of traditional manufacturing in the 1960s and 1970s, when Japanese manufacturers adopted Deming's Total Quality Management principles, significantly reduced defects introduced during manufacturing, and thus drastically reduced recalls and associated costly rework. The Standish Group in 1999 reported that 23% of software and information-technology projects were cancelled before completion, while only 28% finished on time and budget with expected functionality.3 Standish Group data also indicates that 60-80% of the cost of software development is rework—that is, fixing defects that are found during testing. This cost would be avoided if better design practices were used.

It is high time that we in the software industry adopt what I call a Quality Doctrine for Software:

- quality systems vs. defective products
- secure designs vs. unlimited vulnerabilities
- measured quality vs. undisciplined coding
- shorter cycle times vs. endless testing
- doing it right the first time vs. fixing it later
- self-directed teams vs. unpredictable schedules

trained and disciplined software practitioners vs. more of the same

By doing so, we can significantly advance the state of practice in software engineering and have a profound impact on both the security of our information systems and on our economy.

Stephen E. Cross
SEI Director and CEO

The COTS Spot

Is There an Intruder in My Computer?

Lawrence R. Rogers

Understanding the ways you can secure your home has much in common with understanding how to protect the security of your computer network. Let's look at the parallels. Imagine that it's summertime, and you are getting ready to go on vacation. This year is a little different, though, because over the last few months, you've bought a new DVD player, a big-screen TV, and a computer. You decided to beef up your home security by contracting the services of a security company. They've installed a system intended to guard the perimeter of your house. With everything now secure, you head for the beach and a week away from cell phones, beepers, email, and your boss!

When you come home, all looks well; the TV, DVD player, and computer are all where you left them. There were no calls from the security company, so it's safe to assume that your house wasn't broken into and robbed, right? That seems like a reasonable conclusion.

Let's add to that scenario fresh tire marks on the lawn, a broken pane of glass from a door, and a report from the security company that an alarm went off. Now you are sure that somebody tried to break in. Did they get in, and, if so, what did they do?

You check the house and everything looks normal, that is, everything is as you remember it. Nothing was moved or disturbed, as nearly as you can tell. You conclude that even though there was a break-in, nothing was taken. You fix the window, reseed the lawn, thank the security company for their information, and move on.

You might have drawn the wrong conclusion, though. The thieves didn't steal the big items, preferring instead to take smaller ones, like the ring that was in your bedroom jewelry box. Since you only wear that ring on special occasions, you probably won't notice it's gone until you want to wear it again. Even then, you might not connect its loss with the break-in.

So, how would you know if anything had been stolen or tampered with by someone breaking into your house? That's a tough question. You could take pictures all around your house to help jog your memory should there be a break-in. Would that have helped you to determine that your ring had been stolen?

In order to remember all of the items around your house and know where they were you'd need to photograph literally every inch. That's an almost impossible task. Moreover, every time you left for a few days (or even a few hours), you'd have to retake

every picture so that you would catch recent changes on film. It might be fun to do once or twice, but the process would get old quickly.

Video surveillance is another way to record the events around your house. Assuming everything is installed properly, works as designed, and the videotape does not run out, a video log file can help you understand what happened while you were away. Again, constantly videotaping every inch of your house is a daunting task, and it can be fairly expensive.

Now let's switch from the scenario of a home break-in to a computer break-in. How would you know if an intruder tried to break in to your computer, if the intruder was successful, and ultimately what the intruder did once he or she broke in?

Just as you needed to take photos of all items in your house to be able to detect unwanted changes, you need to have a record of every file, directory, device, and setting on your computer. Similarly, as the videotapes show the changes that took place while you were gone, you need a log of the changes that happened while your computer was running. Additionally, just as you are aware from the videotapes that it's normal for the mail carrier to come to your door at about noon every day, you need to be aware what events are normal for your computer.

Do you know these things about your computer system? Do you have a list of all of the files, directories, devices, and settings? Do you know how they change as the system and applications run? Do you know what is normal and, conversely, what is unexpected and therefore potentially a sign of an intrusion? If you don't, how would you know if someone has broken into your system? And if you have had a break-in, would you be able to figure out what the intruder did?

Characterizing Your System

This process of identifying all files, directories, devices, and settings, as well as a having a log of their changes and some understanding of normalcy, is called characterizing a system. While it is pretty easy to identify these items, it's not as simple to know how they change as your system runs. It should be easy, though, because somebody does know, and you ought to be able to use their knowledge to help characterize your system. Do you have any idea who keeps this information? How about the operating systems and applications vendors? After all, they either wrote the programs you're running or they have access to the source code used to create them. They can identify which files, directories, and devices change so you can decide which changes are normal. Now, if only vendors would tell you what to expect when your system runs!

Is this reasonable information to expect from a software vendor? Let's look at vendors from other industries. Take the automobile industry for example. My owner's manual—the one that's supposed to be in the glove compartment—says "Under certain driving conditions such as heavy stop and go traffic, or driving up hills in hot weather, the [engine coolant temperature gauge] pointer may indicate at the top of the NORMAL band. This is also acceptable." What's more, the owner's manual goes on to tell me what to do when the pointer is out of the NORMAL band. They are telling owners what constitutes normal and how to react to an abnormal condition.

Now, go look at the documentation for one of your applications. Does it tell you what happens to your system when you run that application? Does it tell you, for example, whether it creates files somewhere in the file system or that running 1,000 instances at the same time may cause your system to slow to a crawl? Probably not.

What are you—the vigilant systems administrator—to do? In this period of time before the vendors provide you with the information you need to understand and secure your systems, you'll need to figure all of this out yourself. First, you'll need to characterize the operating system and its applications in a pseudo-production test environment. That usually means acquiring systems that you'll use to understand what happens once they're released for production use. Set them up and run them as you would in production.

Next, you'll need some characterization software. There are both commercial and freeware products. One popular tool for characterizing a system is TripWire from tripwiresecurity.com. It is multi-platform, and some versions even come with source code. There are many other tools with similar functionality. These all fall under the general category of host-based intrusion detection tools. Try an Internet search using that phrase to see what other tools are available.

Finally, you put it all together and learn what files, directories, devices, and settings are on your systems and how they change over time. In your controlled test environment, you'll learn what is normal. Be aware, though, that once your system goes into production, you'll probably learn more about what constitutes normal, because no matter how good your test systems are, they only approximate your production environment. That's all right; just seek to understand this new set of changes and incorporate them into your characterization.

Now files, directories, devices, and settings are really only a part of the complete characterization of a computer system. Other attributes to look at are

- running programs. What resources do they consume and at what times do they run? For example, if your file system backup programs were running at 11 a.m., would that be considered normal? How about a word processor that has already used ten hours of CPU time?
- network traffic. If your email server suddenly starts making HTTP connections to another computer system, is that normal? What if the flow of Web traffic suddenly increases by an order of magnitude, is that normal?
- performance. Would you know if your Web server was "slow" today relative to other days? How many transactions can your transaction server handle?
- the operating system itself. Intruders are actively changing how the operating system works, so applications work differently even though they remain unchanged. Imagine what would happen if the operating system call that executes a program was changed to execute a different program instead.

Unfortunately, the tools available to check these attributes are not as mature as those that check the files, directories, devices, and settings. Nevertheless, as a vigilant systems administrator, you need to account for these other attributes in your full system characterization.

At last, it's summertime again and you're looking forward to some relaxing time away from your normal routine. You've noted where things are in your house, and you've employed that security company to keep an eye on your perimeter. At the office, you've also taken the time to learn more about your systems, and you feel confident that you know how they work and what constitutes normal. Your coworkers will use your new characterization to watch these systems during your absence. Next stop: the beach!

About the Author

Lawrence R. Rogers is a senior member of the technical staff in the Networked Systems Survivability Program at the Software Engineering Institute (SEI). The CERT Coordination Center® is a part of this program. Rogers's primary focus is analyzing system and network vulnerabilities and helping to transition security technology into production use. His professional interests are in the areas of the administering systems in a secure fashion and software tools and techniques for creating new systems being deployed on the Internet. Rogers also works as a trainer of system administrators, authoring and delivering courseware. Before joining the SEI, Rogers worked for 10 years at Princeton University. Rogers co-authored the Advanced Programmer's Guide to UNIX Systems V with Rebecca Thomas and Jean Yates. He received a BS in systems analysis from Miami University in 1976 and an MA in computer engineering in 1978 from Case Western Reserve University.

This and other columns by Larry Rogers, along with extensive information about computer and network security, can be found at <http://www.cert.org>.

The Architect

Aligning Business Models, Business Architectures, and IT Architectures

Rick Kazman, Hong-Mei Chen

The alignment of business models and IT (information technology) architectures has been a critical issue for IT organizations for as long as IT has been an important factor in the success of organizations. Despite the near-universal acceptance of this principle, how to actually realize alignment is no clearer today than it was ten years ago. Specifically, it is not always clear *when* to align, *what* to align, and *how* to align. For example, when businesses encounter changes (either internal or external changes, such as business strategies, new IT products, or new legal requirements), they have to "know" whether there is any misalignment caused by the changes, have a way to gauge the misalignment, and then correct the misalignment. They have to decide in what ways, how much, at what cost, and according to what timetable the misalignment should be corrected. Since rapid change in the business environment is the only constant, the research question is: Can misalignment be prevented and, if so, how? Current SEI research has resulted in development of a method for detecting and correcting misalignment called the Business IT Alignment Method (BITAM).

Earlier research on alignment focused on defining models of alignment information and the relationships among the parts of the models. This early research was important and influential—it introduced the ideas of alignment to a wide audience and structured the space of inquiry. But it said little about how to *gauge* and actually correct misalignment. That was always the "magic."

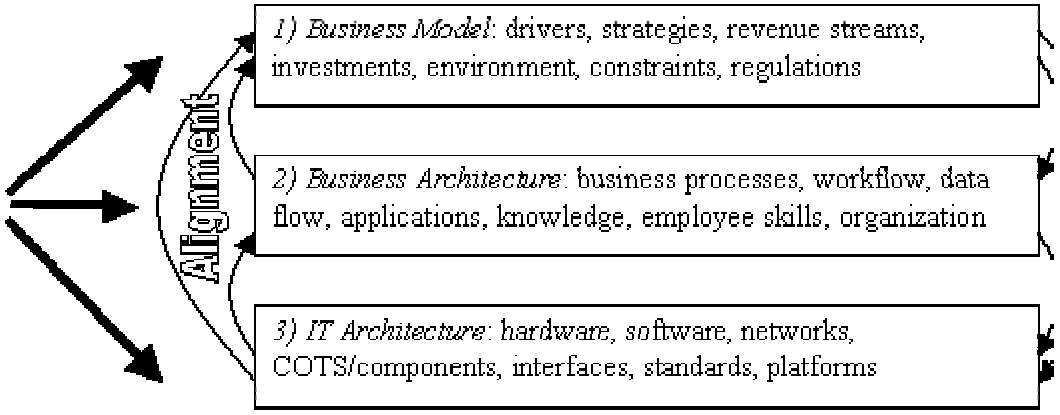
Furthermore, since 1993, when these models were first proposed, the environment with respect to IT has changed dramatically. Back then, IT functions were often a supporting role in an organization. Since the Internet revolution, IT has become critical to business success, for strategic advantage as well as for strategic and operational necessity. And IT planning has become an integrated part of business planning. With rapid technological advancements and business-environment changes, organizations are required to adapt and seek ways to continuously innovate. The older models of IT alignment were not created in this environment and are not suited for it. The strong interdependency between business models and the IT systems that realized those models was not explored in depth. In fact, the discussion of business models and processes was often completely removed from the design of the IT architecture. Earlier models address alignment issues only at the "strategic" or conceptual level of a business system. They were not able to address alignment issues down to the level of IT architecture, which is critical for alignment. Based on this background, the goals of this research are

- detection of misalignment
    - to empirically categorize the types of misalignment and the types of changes that may cause each of them
    - to develop a model of information required for realignment that has been tested with large IT organizations, beginning with business goals and requirements and ending with IT architectures

1. correction of misalignment
    - to precisely and *operationally* define what alignment is, and how it should be measured and managed
    - to create an information model with standard means of eliciting, collecting, and organizing the information needed by the alignment/realignment process
    - to consider a range of realignment strategies and have a decision procedure for choosing among the alternatives

- *prevention of misalignment*
    - to develop processes that organizations can put in place that allow them to continually measure alignment and gauge when they are becoming misaligned. They can then take corrective measures as a continual part of their processes, rather than as a post-facto cleanup step.

This research is situated in a model of IT architectures as shown in Figure 1.



1.1.1.1.1.1

1.1.1.1.1.2 Figure 1: Business IT Alignment Model

This three-level model delineates the layers of business models, business architectures, and IT architectures. Changes may affect any of the layers, and these changes will ripple to the other layers. For example, a change in the business environment (a new competitor) may dictate that some existing business processes be changed (e.g., changes to customer relationship processes), which will in turn change the IT architecture that supports these processes (perhaps a new Web-based customer relationship management system will be integrated).

Each of these layers needs to be studied and modeled. However, simply understanding the layers is not sufficient. Misalignments are improper *mappings* between the layers. Realignment activities are those activities that restore coherence to the mappings. Thus, we need to characterize each of the layers as well as the mappings among them.

Foundational Ideas

There are three stages of maturity in an organization's ability to deal with misalignment. Each stage builds on the previous one. The three stages are detection, correction, and prevention of misalignments. To be able to correct a misalignment you must be able to detect it, and you must be able to gauge the extent of the misalignment and determine an effective realignment strategy. To be able to prevent misalignment, you must be able to do continual detection and correction as part of your development process.

Detection

For an organization to detect alignment problems, they need to be able to precisely characterize business goals and the relationship between these goals and IT requirements. Typically IT organizations have no formal means of characterizing business goals. Business goals are developed and communicated in an ad hoc manner, and most IT organizations' requirements processes are imprecise. In addition, an IT organization needs to have an established process for tracing requirements to their realization in business architectures and from there to IT architectures. These steps are necessary to be able to detect and characterize misalignment.

Correction

Once a misalignment has been detected, the organization needs techniques for characterizing the nature of the misalignment and the degree to which the three levels are misaligned. Once this characterization has been made, it is possible to consider actions that may be undertaken to correct the misalignment. But any corrective action may have consequences for any of the three levels. So any theory of misalignment or method for realignment must include techniques for

- considering a palette of corrective strategies
- comparing the various strategies, along with their tradeoffs and economic implications
- choosing the optimal strategies based on their consequences on all three levels

When selecting realignment strategies, a number of difficulties become apparent. For example, as already mentioned, it is unclear how business goals should be characterized. But correction involves many stakeholders and considerable risk. To mitigate this risk, we need to include all relevant stakeholders in the correction process. And we need to use these stakeholders to negotiate requirements and evaluate proposed solutions. Finally, we need a way to ensure that there is traceability—from business goals through to requirements through to architectures.

Prevention

From Figure 1 we see that there are three alignments that we need to manage: the business model to the business architecture; the business architecture to the IT architecture; and the business model to the IT architecture. The only way to prevent misalignments is to manage these three alignments continuously, as befits an organization of high process maturity. We can do this as follows:

- We align the business model to the business architecture by creating and exercising operational scenarios that satisfy the business requirements.
- We align the business architecture to the IT architecture by exercising the same set of operational scenarios.
- We align the business model to the IT architecture by creating and exercising scenarios that satisfy the business drivers (quality attribute requirements).

We can then check the alignments in the reverse direction. Checking the alignments from IT architecture to business architecture and from business architecture to business goals involves checking that the results of exercising the operational scenarios still meet the goals of the level above. However, checking the alignment from the resulting IT architecture to the business goals is more complex. We believe that this is a process of ensuring that the IT *investment strategy* is fulfilled by the alignment activities. The Cost Benefit Analysis Method (CBAM) can be used to validate that the investments fulfill the organization's business goals. [1].

Note that this approach is in stark contract to existing work on IT alignment, in that it proposes a concrete set of steps, each of which contributes to the creation of some *measurable* outcomes. While other approaches, such as basing alignment on a balanced scorecard [2], are useful for raising awareness of misalignment issues, they do not provide explicit guidance regarding how to measure failure or success and, more crucially, they do not aid in determining realignment strategies. The objective of our research is to do both.

Overview of the BITAM

Managing IT alignment boils down to four distinct questions: (1) How can we capture business goals (and the requirements derived from them) precisely, unambiguously, and repeatably? Additionally, how can these business goals and requirements be negotiated in an informed manner among a diverse group of stakeholders? (2) How can we document IT architectures in a manner that can be analyzed with respect to the process of alignment? (3) How can we gauge misalignment, either qualitatively or quantitatively, and how can we validate these measures? (4) How can we choose realignment strategies and prove that these strategies are optimal with respect to the organization's business goals, including return on investment?

To realize these alignments, we have developed a 12-step method in the BITAM. The steps are as follows:

1. Elicit business drivers from key management stakeholders.

2. Elicit a set of operational scenarios from the entire group of stakeholders.

3. Elicit a set of change scenarios from the entire group of stakeholders.

4. Prioritize the operational scenarios and change scenarios.

5. Elicit the business architecture from the key information architects.

6. Elicit the IT architecture from the key technical architects.

7. Map the operational scenarios to the business architecture.

8. Map the change scenarios to the IT architecture.

9. Assess the misalignments.

10. Propose realignment business strategies.

11. Propose realignment architectural strategies.

12. Evaluate the new business and architectural strategies as investments.

These steps are evolving, but they provide a starting place from which to iteratively refine. For example, a requirements negotiation strategy has been incorporated in the method. The SEI has created a second version of the CBAM that addresses many of the problems with the initial version, primarily in quantifying uncertainty and managing the variability in stakeholder judgments. There is also a substantial body of results and experience from previous case studies. In the course of developing these methods, SEI staff have designed documentation templates for eliciting architectural information, standard sets of analysis questions, standardized approaches for dealing with the analysis of quality attributes, techniques for quantifying uncertainty, and techniques for prioritizing and validating stakeholder judgments.

Conclusions and Future Work

The BITAM holds tremendous promise for aiding organizations in managing their IT alignment processes. The BITAM is built on a foundation of road-tested analysis

methods, such as the Architecture Tradeoff Analysis Method [3] and CBAM, and is being validated in the same fashion—through real-world case studies.

References

[1] Software Engineering Institute. "Cost Benefit Analysis Method (CBAM)." <http://www.sei.cmu.edu/ata/products_services/cbam.html?si> (2002).

[2] Kaplan, R. and Norton, D. *The Balanced Scorecard: Translating Strategy into Action.* Boston: Harvard Business School Press, 1996.

[3] Software Engineering Institute. "The Architecture Tradeoff Analysis Method." <http://www.sei.cmu.edu/ata/ata_method.html?si> (2002).

About the Author

Rick Kazman is a Senior Member of the Technical Staff at the SEI. His primary research interests are software architecture, design and analysis tools, software visualization, and software engineering economics. He is the author of over 50 papers and co-author of several books, including Software Architecture in Practice and Evaluating Software Architectures: Methods and Case Studies.

The COTS Spot

Risk/Misfit Redux

Robert C. Seacord

Introduction

In our book, *Building Systems from Commercial Components* [1], we introduced Risk/Misfit as a decision aid for exposing and quantifying the cost and risks of using and repairing components. Risk/Misfit requires that the use of a particular component feature be justified according to the design risk that arises from not using the feature.

There are a number of uses for Risk/Misfit. In the book, we provide an extended illustration for selecting among strategies for repairing a design risk caused by the lack of a component feature. Another use is a best-fit evaluation of a component ensemble.

Component ensembles are a central theme of our approach to building systems from commercial components (they were covered in both my second and third quarter columns last year). Component ensembles are collections of compatible components that satisfy a design problem. Model problems (the subject of my second quarter 2001 column) can be used to establish feasibility, that is, that a component ensemble can satisfy a design problem within a given set of constraints. However, when multiple design ensembles are feasible, we must use Risk/Misfit or another comparable technique to select an optimal ensemble.

Ensemble Evaluation

To select an optimal component ensemble with Risk/Misfit, it is first necessary to establish common criteria against which the ensembles can be evaluated. It is insufficient to simply list the features provided by each ensemble and quantify the costs and risks of not having these features, as the feature sets are driven by the ensembles and cannot provide a common basis for evaluation. Instead, we invert the Risk/Misfit technique we applied to the selection of repair strategies, and start by listing the major risks inherent in the design problem that the ensemble is meant to solve. As in the model problem process, these risks can be iteratively extended during the evaluation process, as long as common criteria are reapplied to each ensemble under consideration.

In this modified process, we then calculate the maximum risk for each ensemble. In Table 1, we list three project risks and assign values to each of these by defining a scale (in this case, 1–100) and selecting a value for each risk. In the third column, we estimate the

value of eliminating each risk. This value is obtained by asking yourself (or your manager), "How much would we pay to completely eliminate risk X?" If this is not a positive number, the risk is probably not significant enough to be considered in the evaluation.

Table 1: Calculating Ensemble Risk

Project Risks	MR	Worth ($) to Eliminate	V ($/R)	Feature Set	RR	ρ(r)
Inadequate performance	92	$956,000	$10,391	E1: F2, F4	5	$51,955
				E2: F1	12	$124,692
				E3: F3, F7	6	$62,346
Lack of software engineers qualified to work with proposed technologies	87	$270,000	$3,103	E1: F3	10	$31,030
				E2: F4, F5	0	$0
				E3: F1, F2	3	$9,309
Inadequate data confidentiality	79	$456,000	$5,772	E1: F6	10	$57,720
				E2: F8	22	$126,984
				E3: —	79	$456,000

The fourth column in our table simply involves a small amount of math. In it, we divide the total worth to eliminate the risk by the maximum risk (MR), resulting in a value per risk unit.

For example, Table 1 lists "inadequate performance" as a project risk. This risk is assigned a maximum risk value of 92, and it would be worth a whopping $956,000 for this problem to simply go away. These two figures are used to calculate the worth of eliminating each risk unit at $10,391.

Now that the preliminaries are out of the way, we can get to the fun part. We now identify those features of each ensemble that address each project risk. These features must be apparent in the application of the component ensemble to the design problem. Many component features are apparent in the overall ensemble, while others are not. Only those component features that are apparent in the solution should be used in the Risk/Misfit evaluation. For example, if your product supports a PKI feature, but you do not intend to use it in the ensemble, do not evaluate that feature.

Some ensembles might demonstrate emergent properties that result from combining products. If an emergent property produces a positive effect, it should be considered as a feature of the ensemble.

In the next column, we calculate the residual risk (RR) remaining once the availability of these ensemble features is considered. For example, in our first ensemble (E1), features F2 and F3 both address the risk of inadequate performance. However, these solutions are not perfect (solutions seldom are) and some residual risks remain. These residual risks are captured in the RR column. The final column, labeled "ρ(r)," converts residual risk to dollars. In this case, the function ρ(r) multiplies the residual risk number by the cost per unit risk value we calculated for column 4.

Concluding the Evaluation

Once we have completed this analysis for each project risk, we can calculate the total residual risk for each ensemble by simply summing the ρ(r) values for each ensemble for each risk. For example, the total residual risk for ensemble E1 would be equal to Σ ρ(r): $51,955 + $31,030 + $57,720, or $140,705. Total residual risk costs for each ensemble are shown in the third column of Table 2 below.

At this point, we could compare each ensemble and select the one with the lowest residual risk. However, we would also like to include another factor, which is the total cost of ownership (TCO) for the ensemble. The TCO can include the cost of licensing all the products, integration costs, administration costs, training costs, and other costs related to buying, learning, and using the product. How you calculate the TCO is largely up to you and your organization, since organization and other factors can affect which costs may or may not be of concern to you in your decision-making process.

Once we have the TCO for each ensemble, we can add these costs to the overall residual-risks costs for each ensemble. Again, the ensemble with the lowest overall cost and risk would normally be selected. In Table 2, Ensemble E2 would be selected, as it has the lowest combined TCO and residual risk.

Table 2: Ensemble Total Cost of Ownership

Ensemble	TCO	Σ ρ(r)	TCO + Σ ρ(r)
E1	$350,000	$140,705	$490,705
E2	$247,000	$151,676	$370,000
E3	$423,000	$527,655	$950,655

Summary

While the above example is somewhat simplistic, it should provide the general sense of how Risk/Misfit can be applied to the best-fit evaluation of component ensembles. The area in which this example could be further expanded is in the evaluation and selection of repair strategies. For example, it may be that the residual risk for many of the ensembles described above are still quite high. It is quite possible that repairs can be made to further reduce these residual risks. Also, repairs could be made to reduce risk resulting from missing features (that is, the residual risk is the same as the maximum risk). The cost of making these repairs and the residual risk remaining after their completion must also be accounted for in the evaluation.

References

[1] Wallnau, Kurt C.; Hissam, Scott A.; and Seacord, Robert C. *Building Systems from Commercial Components*. Boston: Addison-Wesley, 2002, ISBN: 0201700646.

About the Author

Robert C. Seacord is a senior member of the technical staff at the SEI and an eclectic technologist. He is coauthor of the book *Building Systems from Commercial Components* as well as more than 40 papers on component-based software engineering, Web-based system design, legacy system modernization, component repositories and search engines, security, and user interface design and development.

Watts New

Surviving Failure

Watts S. Humphrey

You're on a project and it's headed south. While everybody is trying their hardest, and you are doing your level best to help, you can feel it in your bones: the project is doomed to fail. What can you do? You have three choices.

13. Keep plugging away and hope things will improve.

14. Look for another job.

15. Try to fix the problems.

Keep plugging away

While continuing to plug away is essential, it will not actually improve things, and it is not very professional. Often the best way to guarantee project failure is to keep working in the same way. Inertia is a form of surrender. You are acting helpless and hoping somebody will save the day, or at least hoping that the crash will not be fatal. So, plug away by all means, but do something else as well.

Look for Another Job

Choice two is to look for another job, either in your current organization or elsewhere. This is always an option, and you should consider it if things get bad enough, but job-hopping has serious drawbacks. First, the situation in the new organization may not be much better—and it could be worse. Second, since projects often fail, you cannot continually run from failure or your resume will look like an employment catalogue. While this is not as serious a concern as it once was, it costs money to hire, orient, and train people. Unless management believes you will stay long enough to recoup their investment, they will not hire you. Third, changing jobs is disruptive and could involve a move and a new home. Once you have done that a few times, it loses its charm. Finally, in any organization, it takes time to become established and accepted. Until you are known and respected by management, you will not be considered for the best jobs. By moving, you start all over again at the bottom of the seniority list.

Fix the Problems

Assuming that you don't want to give up, disrupt your life, or become unemployable, your best choice is to fix the problems before it is too late. Doing this, however, is tricky and it could actually damage your career if not done properly. Remember, the bearer of bad news often gets the blame. So if you are outspoken about the project's problems, expect to be made the scapegoat. This does not mean that you shouldn't act like a professional and try to fix the problems, just that you must do it very carefully.

Think Like a Manager

Since you must deal with management to solve most project problems, try to put yourself in their shoes. Consider the problems they face and decide what you could do that would help. In doing this, you can safely make three assumptions.

1. Management already suspects that the project is in trouble.

2. They want solutions, not problems.

3. Managers do not want competition.

Management Already Senses the Problem

Managers have lots of ways to get information, and the higher they are in the organization, the more sources they have. Managers also often develop a good intuitive sense and they can smell trouble even before anyone tells them. Once managers have worked with a few projects, the troubled ones take on a distinct character. The people begin to look worried and uneasy, the laughter and fun disappear, and status reports get vague and imprecise.

There are also various test, support, financial, and administrative groups that deal with most projects, and their people will hear of, or at least sense, the first signs of trouble. These people will almost certainly have passed on what they have learned to management and, if it is bad news, you can be sure that it will travel fast. So, management either knows about the problems already or has a strong suspicion.

Management Wants Solutions, Not Problems

Busy managers have lots of problems. In fact, a manager's time is largely devoted to solving problems, whether generated by the projects, passed down from higher

management, or imposed by the customer. If you go to your manager with another problem, expect to be greeted like the plague. However, if you show up with an offer of help instead, you will likely be received with open arms.

Managers Do Not Want Competition

You have a manager who is responsible for your assignments, evaluations, pay, and promotion. If your manager sees you as supportive, you can likely get help in fixing the project. However, if your manager suspects you of competing for his or her job or thinks that you are out to get exposure to senior management, expect to get cut off at the knees. If your manager is experienced, you will not even know that you have been skewered until much later, if ever.

So, watch the chain of command and start with your immediate manager. Don't do anything your manager doesn't know about and agree with. While that doesn't mean your manager must know every step before you take it, be completely open and honest. Explain your approach, make sure you both understand the plan, and that you both agree on what you can do without prior approval. However, if the manager does not agree and you go over his or her head to a more senior manager, expect you or your manager to ultimately be fired.

If your manager agrees, he or she may let you carry the story upstairs, but most will do it themselves and you will not be involved or even get any credit. The key is to not worry about credit and visibility, but to concentrate on solving the problems. If you do that, sooner or later you will get plenty of visibility. There is a wonderful line by Dick Garwin, the designer of the first hydrogen bomb: "You can get credit for something or get it done, but not both" [1].

A Strategy for Survival

When you are on a troubled project, the basic survival strategy is to act professionally. It has six steps.

1. Understand the source of the problem.

2. Decide how to fix it.

3. Fix what you can fix by yourself.

4. Review what you have done with your manager.

5. Decide on a strategy for the next steps.

6. Agree on what you can do to help.

Understanding the Problem

While problems come in many flavors, the most common involve unreachable goals. So stop and really think about the current situation and how it happened. Until you understand the problem, it will be very hard to fix.

Generally, the software problems I have seen are caused by either unrealistic schedules or inadequate resources. In either case, management has imposed, demanded, or agreed to a schedule that the current team is unable to meet. Under these conditions, just continuing to plug away and hoping for some kind of miracle merely postpones the day of reckoning and makes it harder to address the problems.

Schedule problems are particularly troublesome because they invariably lead to a host of other problems. When engineers strive to meet an impossible schedule, they are invariably working without a plan, or they have a plan that is unrealistic and useless in guiding the work. When you're in trouble, panic is your worst enemy and that is exactly how teams behave without plans. Everybody rushes the requirements and design work so they can start coding and testing. Pretty soon, nobody knows where the project stands, modules are overlooked, fixes get lost, work is duplicated, and records are misplaced. The project is out of control.

Deciding How to Fix the Problem

Tiny projects can occasionally survive panics, but the larger they are, the harder they crash. The best rule to remember is: when you are in a hole, stop digging. Put down the shovel and make a plan. With few exceptions, when projects are in trouble, the most critical need is to make a plan. Involve the whole team and make as detailed and realistic a plan as you can.

While making a plan will sound plausible to anyone who believes in planning, most programmers can't see how this could help them to get all their code written and tested. Unfortunately, there is no simple answer that will satisfy everybody. However, there are

several reasons that, when taken together, should convince even a skeptical software professional.

1. Every troubled software-intensive project I have seen that was in serious trouble did not have a realistic plan.

2. Every large software project that I have worked with that did not have a plan was in trouble.

3. When a team makes a plan at the beginning of the job, it can generally negotiate a realistic schedule with management and the customer.

4. Even in the middle of a crisis project, stopping to develop a plan will calm the panic, produce a clear understanding of the situation, and provide guidance on getting out of the hole.

5. With the plan in hand, the team can negotiate a recovery plan with management and the customer.

6. When the team follows its plan in doing the work, the team members will know what to do, the team will be able to track and report status, and management will know what to expect and when.

So planning is not magic, but it sure helps.

Fix What You Can Yourself

Before running to your manager with a recommendation to make a plan, look at your own work. Make a list of what you must do and then make a plan for doing it. If this plan shows that you can finish on the desired schedule, maybe you overreacted and the team can finish on time. But if not, you will have a convincing story to show your manager about your problems and how you plan to address them.

In making this plan, observe a few cautions. To the extent that you can, base the plan on historical data [2,3,4]. Also, be discreet in making the plan. After all, if your manager thought planning was a good idea, the team would probably have a plan and you would not be in this mess. So get your story straight before talking about it. Next, if your friends

or teammates have some planning experience, get them to review your plan and identify any holes or errors. Once you have a plan you believe in, talk with your manager.

Talking With Your Manager

In talking with your manager, concentrate on your own work and the plan you have made for doing it. If your plan shows that you can't meet the committed dates, ask for guidance. Maybe your plan includes too many tasks or the manager might see some way to simplify the work. If this leads the manager to asking the other team members to make similar plans, you are on the road to success. But if not, and if the plan helped you to have a realistic discussion with your manager, discuss what you have done with your teammates and suggest that they do the same thing. Then, when more team members have plans, you can all go to the manager and show that the problem is bigger than he or she thought.

Agree on the Next Steps

If, as is likely, the composite of all the team members' plans shows that the project is in serious trouble, suggest that the team make a complete plan. If the manager agrees, he or she will then likely go to higher management to review the project and get agreement to a replanning effort. Then, with that plan, you, your teammates, and your manager will have a sound basis for renegotiating the team's schedule. If you get this far, you will be able to turn the failed project into at least a partial success. If not, consider your other alternatives: either keep your head down and continue plugging away, or find another job.

Conclusions

As long as you continue to work quietly on a failed project, you are part of the problem. By taking a more active role and addressing your part of the job first, you can become part of the solution. This will help the project and your career, and it will help you to behave like a true professional. It will also lead to a satisfying and rewarding way to work.

Acknowledgements

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

In closing, an invitation to readers

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

Thanks for your attention and please stay tuned in.

Watts S. Humphrey
watts@sei.cmu.edu

References

[1] Broad, William J. “Who Built the H-Bomb? Debate Revives.” *The New York Times*, April 24, 2001, pg. D1.

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

[3] Humphrey, Watts S. *Introduction to the Team Software Process*. Reading, MA: Addison Wesley, 1999.

[4] Humphrey, Watts S. *Winning with Software: an Executive Strategy*. MA: Addison Wesley, 2002.

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.

Preventing Security-Related Defects

Lauren Heinz

Shawn Hernan, a team leader with the CERT Coordination Center® at the SEI, spends a good part of his day tracking and analyzing the vulnerability reports he receives from software vendors who have discovered security-related problems in their products. Since January, Hernan and his team have logged more than 1,600 new software vulnerabilities—a staggering number for any system or network administrator to evaluate and possibly act on. The current system of issuing copious warnings and expecting organizations to keep up, Hernan says, is quickly becoming unrealistic.

"It is nearly impossible, and extremely time consuming, for anyone to stay current on these reports and patches. Just reading through them and seeing which ones affect you can take weeks," Hernan says. "We need to look more at preventing them in the first place."

In an effort to help organizations reduce vulnerabilities in software, Hernan has teamed up with members of the Team Software Process (TSP) Initiative at the SEI to consider how

improved software quality methods such as the TSP might help developers detect and prevent security defects earlier in the software-development life cycle. Hernan has also developed a list of the five common causes of software defects, which he hopes will help organizations begin to improve their security practices.

TSP and Quality Software

According to CERT Coordination Center statistics, more than 90% of software security incidents are caused by attackers exploiting known software defect types.

The TSP provides a framework, set of processes, and collection of disciplined methods for producing quality software. With TSP, software teams can produce near defect-free software with typically fewer than 0.1 defects per thousand source lines of code (KSLOC). Given that a vulnerability is a flaw or defect in a piece of software that allows an intruder to read, modify, or disable the way that software functions in a system, using TSP could yield significant results. "We know that TSP reduces overall software defects. We want to see if it can reduce security defects in software," says Noopur Davis, a member of the TSP team who is working with Hernan. "The question is, how do you design, implement, review, and test software for security?"

The answer lies, in part, in extending TSP for secure systems. Davis said this extension of TSP would support secure software-development practices, help predict the likelihood of security defects by creating specific, security-related measures, and be dynamically tailored to respond to new threats. In the coming months, Davis and her team will explore how applying the TSP can help developers identify and remove vulnerabilities during the early stages of software design and development.

Lack of Understanding?

Why are so many software vulnerabilities being reported? The problem, Hernan and Davis agree, is not a lack of available data, knowledge, training, or support regarding security practices. Incident response centers (such as the SEI's CERT Coordination Center), guidelines, checklists, best practices, and other useful materials are widely available. "I have come to believe that a vulnerability is not so much a property of software as it is a property of an organization," Hernan says.

Meanwhile, Davis is curious to see how simply applying TSP might help organizations eliminate more defects, including potential vulnerabilities. For example, most operating systems and commercial software applications are known to have more than 2 defects per KSLOC, or 2,000+ defects per million SLOC. Even if only 5% of these defects are potential security concerns, there are 100 vulnerabilities per million SLOC. "That is still a huge number of defects," she says. "There is significant potential here for TSP to reduce those numbers."

Five Common Causes

Although there are thousands of known software vulnerabilities, Hernan has been able to categorize them into five main causes (listed below). In addition to realizing potential

improvements through applying disciplined processes, organizations can have a better chance of creating secure applications and systems by programming software with these causes in mind:

1. Buffer overflows. These occur when a programmer does not properly check whether there is adequate space to hold data entered by a user. An intruder can exploit this vulnerability by intentionally entering excess data, which interrupts the program's normal operations and allows the intruder to take control.
2. Timing windows. Problems with timing windows occur when a temporary file is exploited by an intruder to gain access to the file, overwrite important data, and use the file as a gateway for advancing further into the system.

3. Insecure default configurations. These occur when vendors use known default passwords to make it as easy as possible for consumers to set up new systems. Unfortunately, most intruders know these passwords and can access systems effortlessly.
4. Bad protocols. Some protocols, or the standards by which information is exchanged over the Internet, lack any security at all. For example, unsolicited commercial email, commonly referred to as "spam," is the irritating result of poor protocol programming.
5. Trusting untrustworthy information. This is usually a problem that affects routers, or those computers that connect one network to another. When routers are not programmed to verify that they are receiving information from a unique host, bogus routers can gain access to systems and do damage.

"If you can develop software that prevents these top five categories, you are going to prevent thousands of potential vulnerabilities," Hernan says.

Next Steps

Davis and Hernan are currently seeking organizations to take part in pilot projects to help develop specific technical solutions—new design and implementation practices—for security, review methods and checklists, and tools. If your organization is interested in participating, please contact the SEI.

For more, contact—

Customer Relations

Phone
412-268-5800

Email
customer-relations@sei.cmu.edu

World Wide Web
http://www.sei.cmu.edu/tsp?si
http://www.cert.org

TIDE: Promoting Technology Adoption Through Technology Collaboration

Len Estrin

Today, the manufacturing paradigm is evolving from a large number of discrete, monolithic organizations to decentralized suppliers linked in supply chains. To optimize performance, supply chains operate in a highly coordinated manner through virtual manufacturing networks.

Unfortunately, smaller manufacturing enterprises (SMEs) are hard pressed to keep pace with this emerging environment. While there are many reasons for this, one of the most fundamental is that traditional ways of conducting business are not fast, accurate, or reliable enough to keep pace.

In a technology summit held in spring 1998, Congressman Mike Doyle initiated a plan to address this national issue. This plan became the nucleus of the Technology Insertion Demonstration and Evaluation (TIDE) Program. Funded through the Defense Department's Manufacturing Technology Program and managed by the SEI, TIDE focuses on helping SMEs overcome the barriers to technology adoption.

Under TIDE, a number of organizations are collaborating with smaller manufacturers in Pennsylvania on projects that demonstrate the benefits of technology adoption. In addition to the SEI, these organizations include several colleges within Carnegie Mellon University, the National Institute for Standards and Technology, Catalyst Connection (formerly the Southwestern PA Industrial Resource Council), and Duquesne University's Institute for Economic Transformation.

SEI and Carnegie Mellon—Partnering for TIDE Productivity

In essence, TIDE is bringing world-class technical and managerial leadership to SMEs. Much of that technical and managerial leadership results from the joint efforts of the software experts of the SEI and the faculty and facilities of Carnegie Mellon University.

For example, the Software Industry Center (SWIC) is an interdisciplinary industry research center at Carnegie Mellon. As part of the TIDE Program, SWIC staff members and researchers are developing business cases to document the return on investment of advanced technologies. SWIC staff members are also delivering strategic guidance to help TIDE demonstration companies become more effective members of the supply chain for original equipment manufacturers and defense contractors. Finally, SWIC researchers are working closely with these companies and other members of the TIDE program to

develop reports and papers. The papers will be published in widely read journals so that other SMEs can benefit from the lessons learned in the demonstration projects.

In another collaboration effort, researchers at Carnegie Mellon University's Robotics Institute and members of the SEI's technical staff are modifying sophisticated scheduling tools, originally developed for the Air Force, to give SMEs a dynamic manufacturing scheduling capability. These new tools will enable SMEs to effectively respond to customer requests and to provide the sudden load capacity their customers frequently demand.

The TIDE program has also established a Software Integration Laboratory at the National Robotics Engineering Consortium facility in Lawrenceville, PA. TIDE personnel are using the lab to help SMEs identify software implementation issues. At the lab, SMEs duplicate their legacy systems and data to examine the interaction with software programs they are considering.

The Results Speak for Themselves

The TIDE program verifies the benefits of advanced technology and paves the way for SMEs to adopt it. In the first demonstration project, TIDE personnel helped Carco Electronics to implement 3D solid modeling and finite element analysis software. Staff members from the SEI and Duquesne University's Institute for Economic Transformation examined Carco's needs and current processes. Based on the findings, they conducted three in-house training programs on solid modeling and functional economic analysis (FEA). TIDE personnel also recommended that Carco increase bandwidth on its computer network to improve modeling speed, and that Carco double its number of software licenses so that more drafters could access them. They also worked with Carco leadership to incorporate solid modeling and in-house FEA as part of the company's strategic vision.

Previously, Carco used computer-aided design (CAD) modeling and FEA to validate components and systems before assembly. Now it is using CAD and FEA early in the design process. As a result, drafting and design time have decreased by 25%. Turnaround time for solid model drawings has dropped from weeks to days. On a recent project, the number of engineering change notices decreased from an average 15 per effort to 1.

When Kurt J. Lesker Company adopted advanced CAD capability through TIDE, it achieved similar results. To date, the company has decreased engineering cycle times by approximately 20% on new systems and 30% on redesigns. The company has reduced

engineering hours for designs of individual components by 25%. And engineering time for products built from stock mechanical components has decreased by 50%.

Evaluating Lessons Learned

While each project features specific results, several lessons apply to nearly all the technology-adoption efforts. The first is that advanced technology is more than a tool. Used effectively, it can be one strategy to help SMEs achieve business goals. By viewing technology as a strategy, managers can look for ways to apply the same technology throughout their organizations. For example, Carco Electronics is now using its 3D CAD modeling software to represent it's ideas in proposals, and has won new business as a result.

Another lesson is that implementing technology often requires non-technological changes. Depending on the organization, there may be organizational issues: who uses the software, designers or engineers? There may be operational issues: how many copies of the software must be purchased? There may be cultural issues: how should sales/marketing and engineering work together when preparing bids, proposals, and sales presentations? The point is, the technology does not exist in isolation. To implement it effectively, SMEs have to consider its effect on all of their business processes.

A third lesson is that, properly implemented, the technology can yield results that are greater than imagined. Certainly, advanced technology can reduce errors and rework, increase productivity, and improve design and engineering quality. But it can also strengthen customer relationships, improve employee morale, and lead to a culture of continuous improvement.

The TIDE program is disseminating the lessons learned and information on the demonstration projects through articles in trade and industry publications, papers in business journals, technical notes, workshops, courses, and the TIDE Web site. The TIDE program is also sponsoring a regional conference September 24, 2002, in Warrendale, PA. The conference will cover overcoming barriers to adopting advanced technology, and will include presentations from the SEI, participating SMEs, and other organizations.

The goal of these activities is two-fold: first, to improve the capabilities and competitiveness of small manufacturers throughout the country, and second, to encourage large OEMs and defense manufacturers to partner with SMEs in their supply chain organizations.

By documenting the benefits of advanced technology, helping SMEs overcome barriers to adopting it, and publicizing the results, the TIDE program is enabling small

manufacturers to succeed in a supply chain-centric world while improving the defense industry in the united states.

For more information, contact—

Customer Relations

Phone
412-268-5800

Email
customer-relations@sei.cmu.edu

World Wide Web
http://www.sei.cmu.edu/tide?si

First International Conference on COTS-Based Software Systems a Success

Lisa Brownsword

It is often thought that using pre-packaged commercial off-the-shelf (COTS) software components is a shortcut for building the same old software systems in the same old way. The SEI has been at the forefront of disproving this notion. Successful use of COTS components is at the heart of an entirely new way of building and sustaining today's industry and government systems. The International Conference on COTS-Based Software Systems (ICCBSS) was the first annual international conference to focus on the challenges of creating and maintaining COTS-based software systems that depend on the successful integration of COTS software and hardware products and, potentially, of custom or legacy (pre-existing) components.

Practitioners and researchers gathered for three intensive days, February 4-6, 2002, in Orlando, FL. Turnout for ICCBSS exceeded all expectations. One hundred seventy-five attendees—almost double the number that the sponsors anticipated—came to the conference from North America, Europe, Africa, and Asia.

Attendees' backgrounds reflected aspects and areas of involvement with COTS products: they included integrators, project managers, acquirers, and researchers from industry, government, and academia.

Attendees' responses were positive, and their comments evidenced their enthusiasm: "Very knowledgeable and high quality speakers." "Invaluable information, whether you are experienced using COTS software components or just starting out." "Timely (and much needed) information for any manager or engineer." "A much-needed forum to exchange ideas on this pressing topic."

Noteworthy keynote speakers served to stimulate each day's discussions. Dr. Barry Boehm, of the University of Southern California Center for Software Engineering and the Center for Empirically Based Software

Engineering (CeBASE), discussed 10 empirical hypotheses about (CBS) that CeBASE is collating and, over time, verifying with additional project data For example, over 99% of all executing computer software instructions come from COTS.

Mike Moore, of NASA, focused on his CBS program management experiences for a large, complex system. Some key highlights of actions he found vital to his projects' success included maintaining relationships with alternative vendors and standardizing interfaces to allow greater "plug and play" across alternative vendors.

Dr. Ivar Jacobson, of Rational Software Corporation, outlined four macro trends that are moving the focus of development to the early stages of a project: components that follow a reuse standard and are used to build systems of interconnecting reusable components; a verification step being included with every activity; using intelligent agents to make software platforms transparent; and generating systems directly from business and system models.

Ralph Hempel, an independent consultant, described a new generation of engineers who are learning to build unconventional solutions for embedded systems, such as alarm systems and automotive controls, through the use of robotic building blocks.

To accommodate the diverse needs of the attendees, three presentation tracks—process, technical, and experience—featured a broad range of current practices and promising research directions. The process track offered a range of proven processes to help acquirers, builders, and maintainers of CBS. Designers and integrators benefited from the technology track presentations, which highlighted state-of-the-art and state-of-the-practice techniques, such as using security wrappers to ensure that critical systems have the necessary levels of security. Practical hints, observations, and insights for achieving success were offered in the experience track. Because so much attention is paid to the failures of CBS, it was noteworthy that so many projects of varying size and across a large spectrum of commercial and government application domains reported on their successes.

To investigate in-depth topics crucial to COTS practice, the conference offered several tutorials. These explored the areas of CBS cost estimation, product evaluation, and system design and assembly. An introductory tutorial provided attendees with an understanding of the differences in the development and maintenance of CBS and the major actions necessary for their successful implementation.

The keynotes and most of the presentations are available for download on the conference Web site (see below). Copies of the proceedings are available for purchase through the publisher, Springer-Verlag. Ordering information is also on the conference Web site.

Preparations for ICCBSS 2003 are underway. It will be held February 10-12, 2003, in Ottawa, Canada. The European Software Institute will join ICCBSS 2003 as a sponsor, along with the SEI, the National Research Council Canada, and the University of Southern California Center for Software Engineering. Details about ICCBSS 2003 are available on the conference Web site.

For more information, contact—

Customer Relations

Phone
412-268-5800

Email
customer-relations@sei.cmu.edu

World Wide Web
http://www.iccbss.org/?si

CERT/CC and Secret Service Collaborate on Security

Erin Harper

Many people imagine that CERT Coordination Center® (CERT/CC) staff members spend their days sitting at computers researching cyber security incidents, and think of Secret Service agents as the people in the dark glasses standing behind the president. So why were they working together in the Information Technology Center at the 2002 Winter Olympic Games? Because security is not what it used to be.

Agents must constantly be aware of potential threats posed by the use of emerging technologies—and these days they have to be able to fight criminals by using computers as competently as they use guns. The CERT/CC began an 18-month collaboration with the United States Secret Service in October 2001 to help the service determine how the security of information networks can affect physical security.

"We are a completely different service now than we were even five or six years ago," says Cornelius Tate, the SEI's resident liaison from the Secret Service. "We still use traditional protection methods, but now there is a whole new layer."

Besides providing for the physical protection of designated government officials and visiting heads of state, the Secret Service also provides security at events of national significance, shares federal jurisdiction with the FBI for investigating and prosecuting cyber crime across state lines, and safeguards the financial system of the United States.

Special Security for Special Events

Soon after their collaboration began, CERT/CC staff members accompanied Secret Service agents on advance trips to the 2002 Olympic Games and to Super Bowl XXXVI, which had been designated as national special security events by the U.S. Congress. When an event receives this designation, the Secret Service assumes lead responsibility for the design and implementation of the operational security plan.

CERT/CC staff members were also present in the Olympic Information Technology Center with the Secret Service from the time the athletes arrived until the closing ceremonies, gathering information for the development of a methodology that will help agents identify and plan a response to cyber threats that could be used to cause physical harm. Taking what they learned from their advance trips to the Olympics, they also traveled to New Orleans to assess potential cyber/physical security risks before the Super Bowl.

Training agents to recognize and mitigate these risks is important because the link between cyber and physical security is increasing as organizations become more dependent on networked systems. Many things that used to be controlled mechanically—such as elevators, lights, and door locks—are now controlled electronically and are affected by networked systems. At the Olympics, software applications were used not only to relay up-to-the-second event results to the media, but also to provide security clearance and access to the venues. In such situations, a cyber security breach could lead to vulnerabilities in physical security.

Focusing on Prevention

Many of our nation's critical infrastructure industries (such as power grids, banking and finance systems, and the water supply) now depend on networked systems as well. Some Australian residents were made acutely aware of this when a disgruntled worker hacked into a computerized waste-management system and caused millions of gallons of raw sewage to spill out into local rivers, parks, and the grounds of an upscale hotel. The worker had made more than 45 attempts to take control of the waste system without being detected.

While most organizations realize that they need to protect their systems from outside intruders, many do not realize that the greatest threat could come from an employee who has access to the organization's computer network every day. The Secret Service National Threat Assessment Center (NTAC) previously conducted studies of assassins and school shooters and has now expanded its scope to include this insider threat. NTAC is hoping to become aware of potential problems before it is too late to take action. The CERT/CC is using its technical expertise to help NTAC identify changes in employees' online behavior that might signal a cause for concern when considered in combination with changes in their behavior offline.

Protecting the Nation's Financial Infrastructure

The Secret Service has traditionally been responsible for safeguarding the financial system of the United States, and protects against a host of illegal electronic activities, such as credit card fraud, identity theft, and the production of U.S. currency through electronic means. In the past, an intruder might have used a gun to commit a robbery, but now cyber criminals possess skills that could enable them to use a personal computer to gain access to thousands of credit card numbers through an insecure e-commerce site.

The Secret Service realized that it could not protect consumers from these kinds of fraud unless it knew as much about computer networks as the criminals did, so it formed the

Electronic Crimes Special Agent Program (ECSAP). As part of its collaboration with the Secret Service, the CERT/CC is currently performing a training and needs analysis for ECSAP. Although ECSAP agents will be the first to receive new training materials developed during this collaboration, every agent coming into the Secret Service will eventually benefit from this research.

Working Toward Common Goals

"We view the collaboration with the CERT/CC as a valued partnership and a tremendous opportunity," says Special Agent Tate. "We bring hands-on experience, and the CERT/CC provides a research and development capability that extends beyond the scope of our traditional protective mission." Tom Longstaff, manager of research and development for the CERT/CC, adds, "The goals of the Secret Service align very closely with the goals of the CERT/CC. We both need to be proactive, to be able to recognize potential threats early and then head off problems before they happen."

Working with the Secret Service to develop specialized training materials is one of the many ways that the CERT/CC is using its extensive research to protect the systems through which we are all becoming increasingly connected.

For more, contact—

Jan Philpot

Phone
703-908-8208

Email
philpot@sei.cmu.edu

World Wide Web
http://www.cert.org