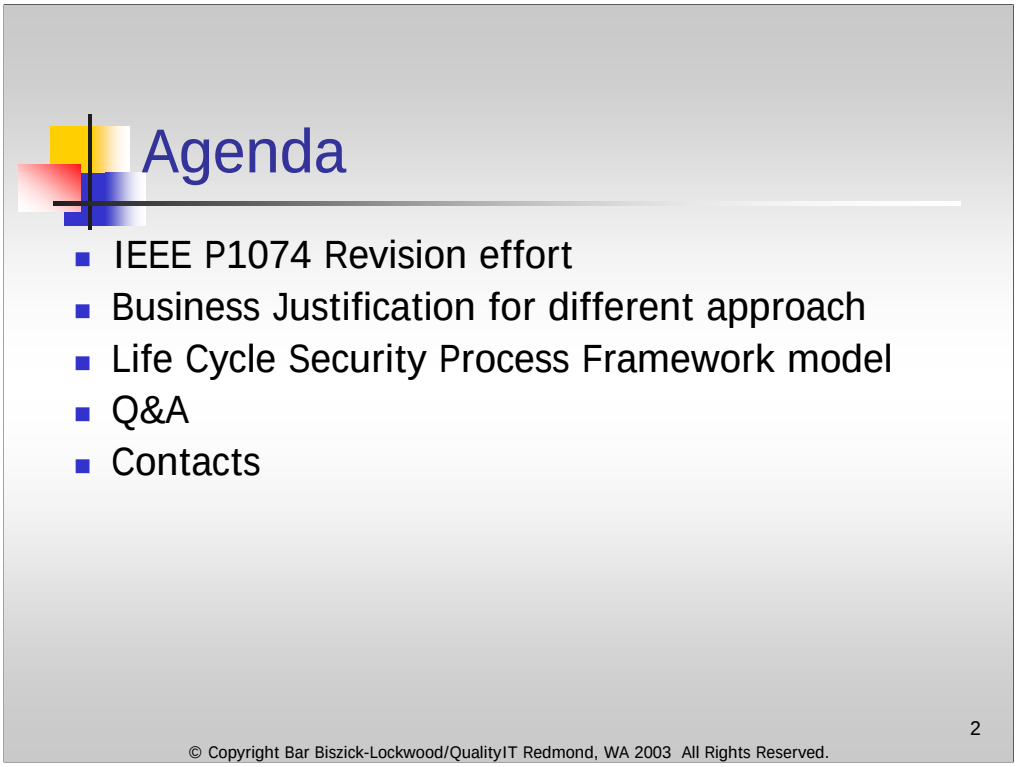
Framework Solution for Life Cycle Security

Bar Biszick-Lockwood, cisa, cissp, csqa
IT Quality and Security Assurance
www.qualityit.net
mailto:barbis@qualityit.net

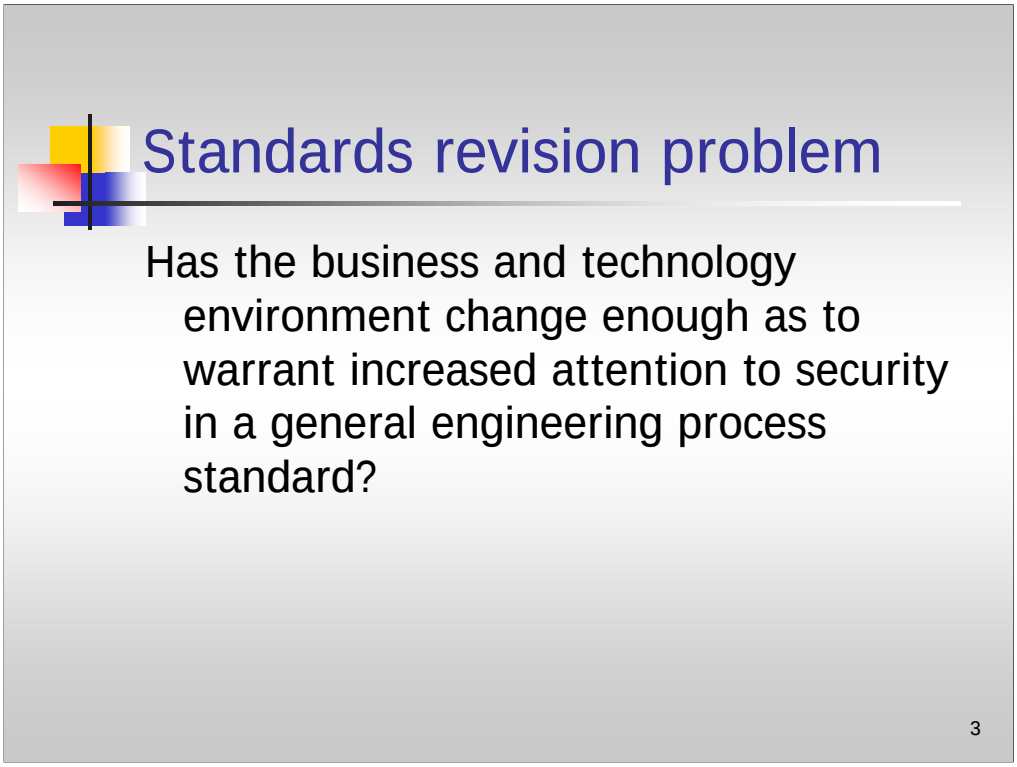
This talk is the result of IEEE standards revision work done between July 2002 and January 2004.



IEEE P1074 *Standard for Developing Software Life Cycle Processes* is used by process analysts and Project Managers to select and sequence the activities that will make up an organization's software development life cycle standard.

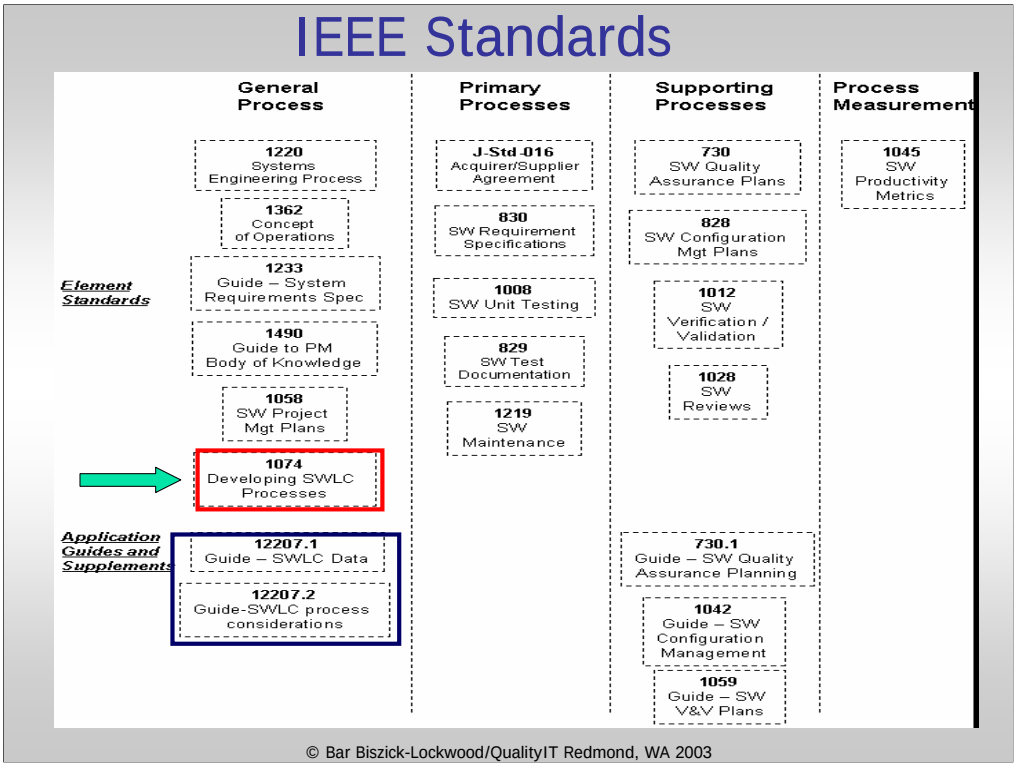
For those of you unfamiliar with the IEEE standards revision process, IEEE engineering standards are generally updated every 3-5 years by groups of volunteer professionals who assess the standard to make sure it continue to meet changing technology and business needs. In July 2002, I was invited to participate in the workgroup revising IEEE P1074.

When asked what I wanted to accomplish as part of the team, I reflected on the fact that the last five years have seen unprecedented growth in information security concerns. Therefore, I volunteered to head up the effort that would evaluate whether information assurance was being handled appropriately in the standard, in light of current business needs.



Our hypothesis was to either prove or disprove that the business and technology environment had changed significantly enough to warrant increased attention to security in this general engineering process standard.

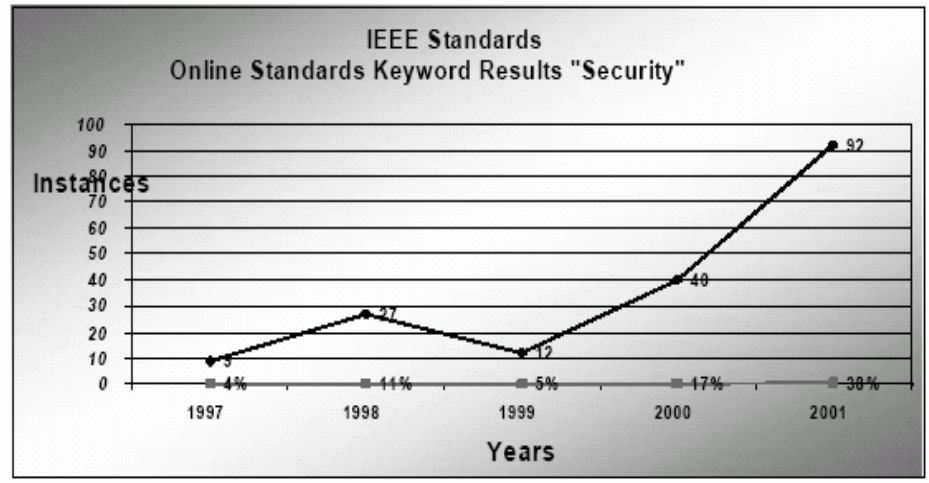
Now, if you've had any experience in standards writing or revisions, then you know that introducing radical change to an IEEE standard is not unlike trying to move a mountain with a pitchfork. Despite the obvious evidence, we knew that we would need to build a compelling business case to support any recommendations that would increase the visibility and priority of security activities in this standard..



First, we looked at how security is dealt with compared to other comparable standards. Direct comparison with ISO/IEC 12207.1 Standard for Information Technology—Software life cycle processes and 12207.2 Software life cycle processes—Life cycle data and with the Canadian ISO/IEC JTC1/ SC7 F.3 Organizational Life Cycle Processes revealed a general lack of attention to security activities in all of these standards. Security was simply mentioned, along with other critical requirements such as reliability and safety, as important factors to consider on projects. Little attention, and virtually no guidance was offered, and there were no references to security specific standards either, since none related to life cycle process exist at this time.

Then we looked at the available security guidance, identifying some 98 ISO standards. Some address the lower layers of the OSI related to things like integrated circuits, data interchange, and connection modes, but the bulk were protocol related standards addressing the Session and Application layers. These dealt primarily with entity authentication, key management and digital signature solutions. We did however, identify two ISO standards with clear relevance to our task: ISO 17799 IEC 17799:2000 *Code of Practice For Information Security Management* and ISO/IEC 15408 *Common Criteria for Information Technology Security Evaluation.*

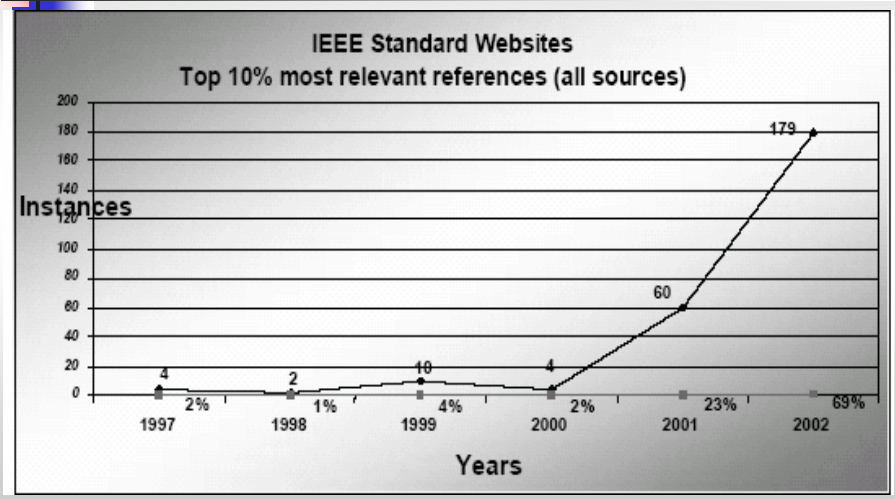
Increase in security standards



5

The IEEE-USA has taken a policy position that supports government and industry initiatives, and reflects changing practices and perceptions of its Constituency. That focus has resulted in a 34% increase of standards with security as a major topical area since 1997.

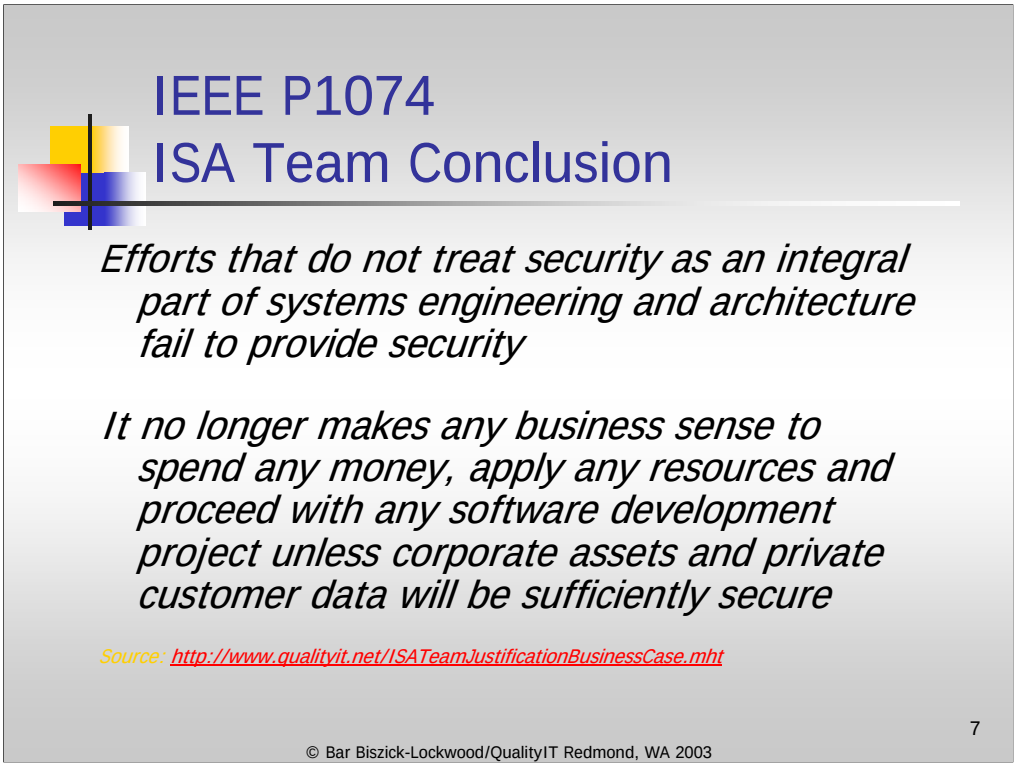
Overall IEEE constituent activities



6

More dramatic is the increase in IEEE general constituent activities related to security over the past few years. To provide cultural perspective, we analyzed general IEEE constituent activities through a keyword search related to security, and found a tremendous increase in the number of instances security appears as a reference or topic of interest among the IEEE constituency.

A small survey of key professionals operating in a variety of information technology professions also helped validate our hypothesis that that there has been a significant enough increase in interest and concern among IT professionals regarding security as to warrant rethinking how we handle security in all engineering standards.

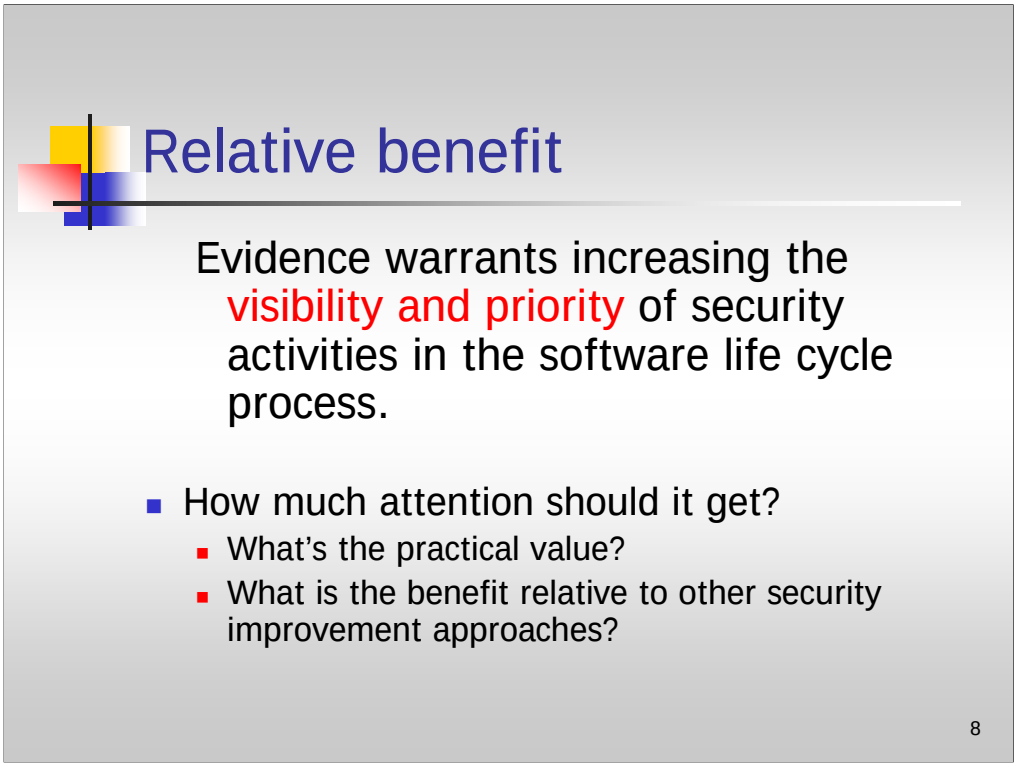


Ultimately, we concluded that:

Efforts that do not treat security as an integral part of systems engineering and architecture fail to provide security

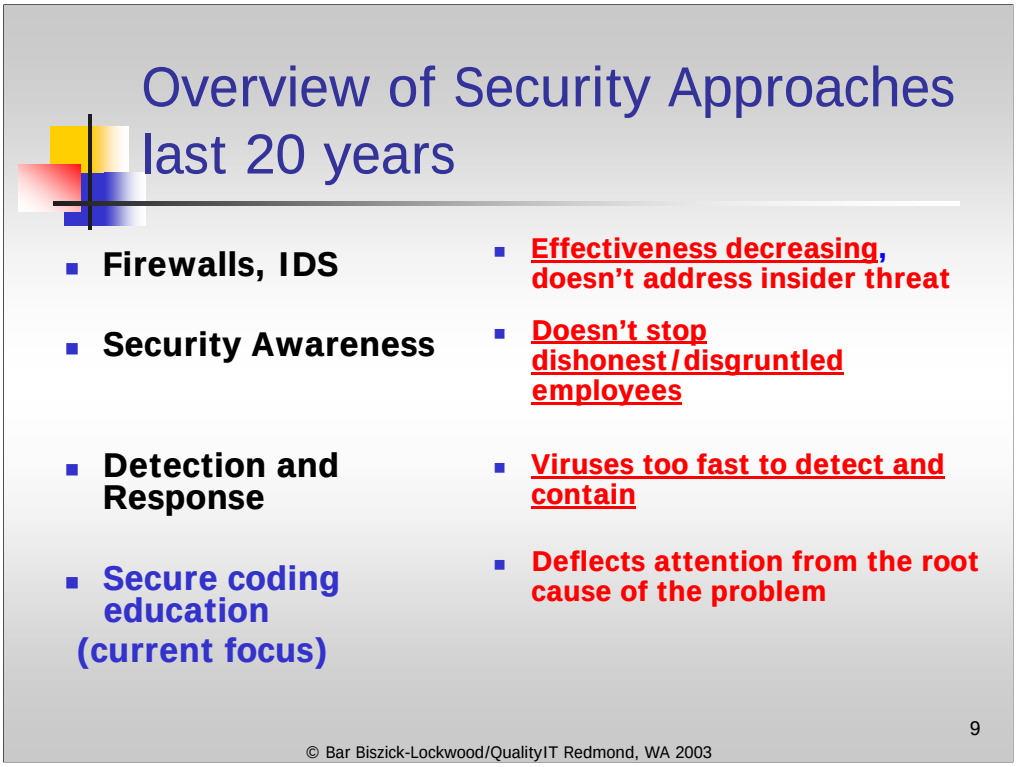
It no longer makes any business sense to spend any money, apply any resources and proceed with any software development project unless corporate assets and private customer data will be sufficiently secure.

This finding has a significant effect on the criteria used to judge the viability of software projects, as well as the resources, budget, tools and skills sets that will be used. It means that where management's chief concern has been historically centered around ROI connected to rich features, speed, and ease of use in software systems, their concern must now shift to balancing these benefits against potential security risk which can have a far greater effect on consumer approval, business stability and profitability in the long run. It means that project teams--traditionally focused entirely on responding to customer requests--must now learn to communicate security risk up front to customers, so the appropriate budget and project needs will be justified. They must have the courage to revise proposed approaches—sometimes radically—in light of real time threat information. Such changes may be costly and unpopular but business in the new millennium requires engineers to demonstrate an unprecedented level of risk awareness and ethics in engineering to meet this contemporary challenge.



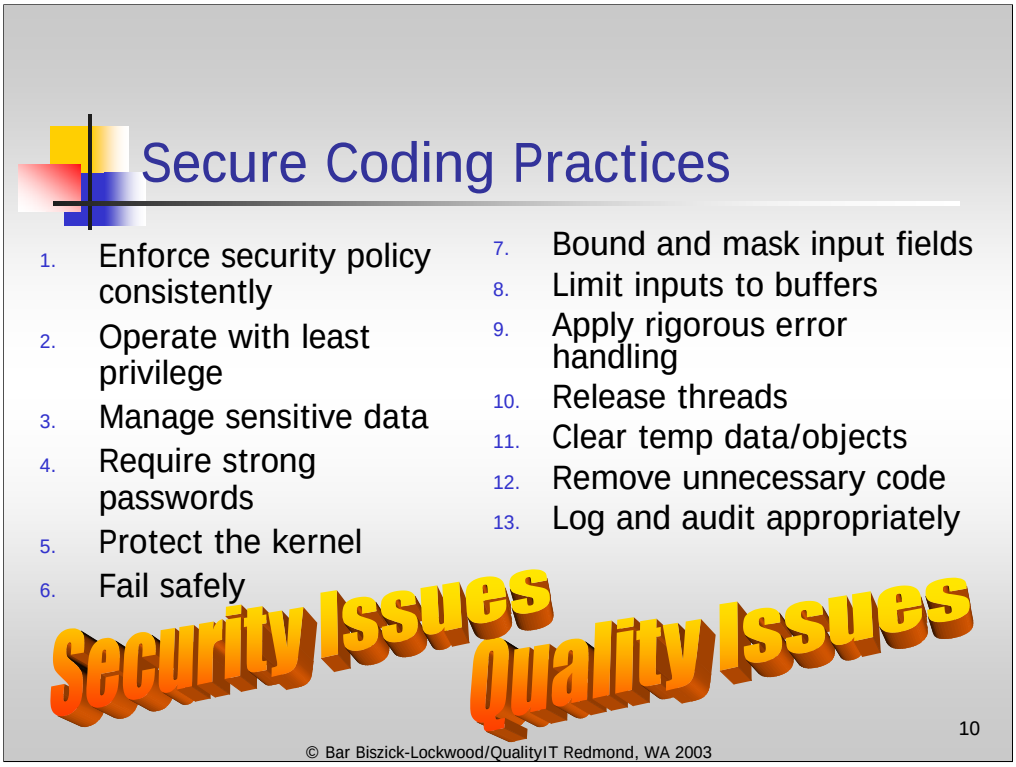
OK, so we've proven security warrants more attention in general process standards, but the question then became "how much more?" What practical value can be had from changing this standard, and how does it compare to the value provided by other improvement approaches.

So, we thought the best way to get a sense of how important life cycle changes might be the overall security problem would be to compare them to the current area of industry focus, secure coding.



It has taken us 20 years of fancy dancing around the problem of security to finally admit that—to hijack a colloquial phases—"It's the Technology, stupid!" Firewalls, meant to keep the outsider's out, fail when an insider is the problem. No amount of security awareness will deter a dishonest or disgruntled employee, who account for almost as much financial damage to organizations as do external hackers. And most people agree that if we are not already there, zero day attacks are inevitable, rendering our emergency response teams almost completely ineffective. We now know that to increase security we have to attack the difficult job of actually building security controls into our technology, not trying to impose it from the outside.

Indeed there is good evidence that suggests that as much as 80% of the security issues can be traced to coding problems. We naturally assume it's because our developers don't know how to code for security, so our current focus is on educating them. But on closer inspection, we determined that secure coding education really deflects us from the root cause of security problems. It turns out that secure coding education was less important as a comparative for 1074, than it was as more compelling evidence to support our recommendations.

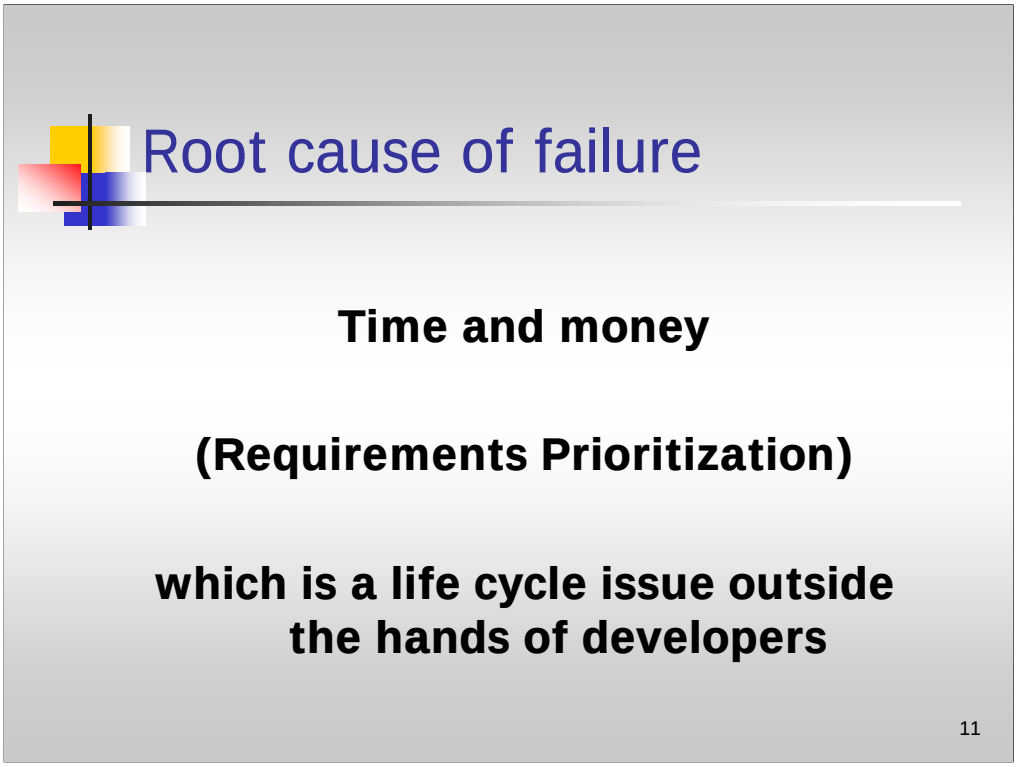


If you troll the web and read the books that promote secure coding, you can boil down the recommendations into these handful of principles.

The group on the left are clearly security problems. But if you ask developers if they know how to do these things invariably they respond that it depends on the technology, the business objectives and the specific implementation. That raises some questions about the validity of these principles as classroom subject matter.

But more surprising is the group on the right. Note they represent the majority of the list. And also note, they none of them have anything specifically to do with security. These are quality coding issues, and in fact, they are basic, fundamental things our developers learned how to do in coding 101 class. Most developers know how to do these things, but they are just not doing them. And if you ask them why, their response is usually that they lack time and money.

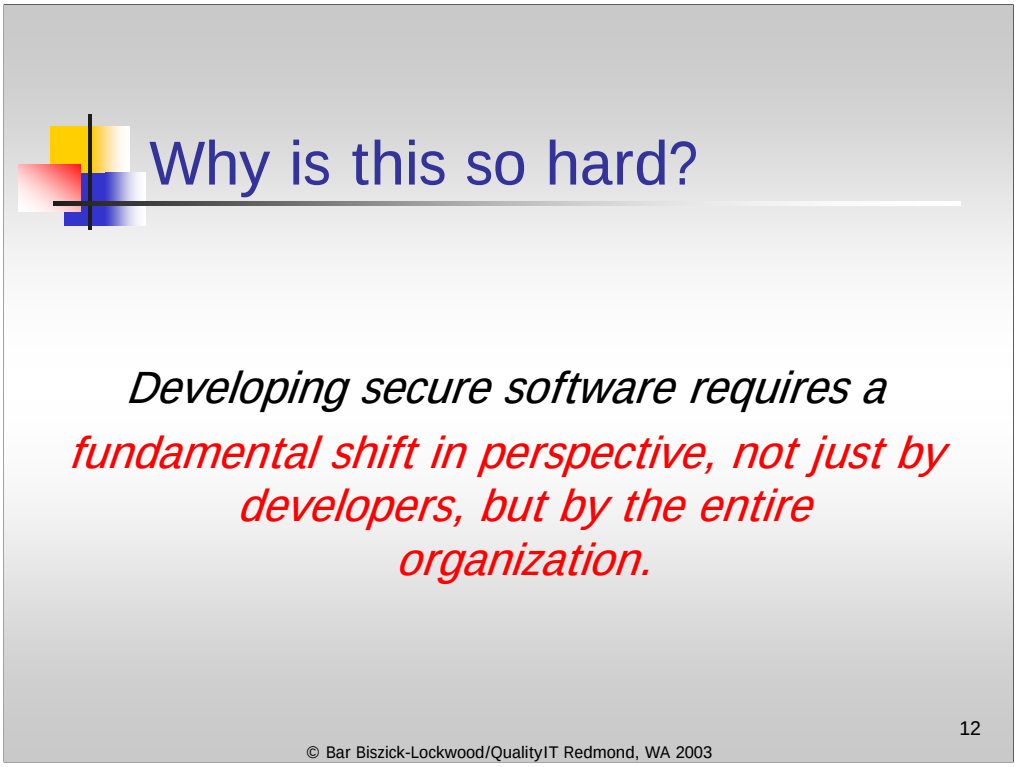
Furthermore, regarding those on the left, most



Time and money is not a secure coding problem, it's a requirements prioritization problem. It's a life cycle issue outside the hands of developers and often entirely outside the hands of most project groups.

Requirements are ultimately controlled by the project sponsor—the entity paying for the creation of the software. Invariably the sponsor's focus is on profitability, not security. And since customers are attracted to what business value a product can provide, sponsors will always emphasize functionality at the expense of secure, quality programming, unless the consequences of poor software can be shown to have clear business impact.

Few project teams are skilled in representing the security risks and their potential consequences to business management in a way they can comprehend and fully appreciate.



The reason why this is so hard, is that building secure software requires a shift in perspective. And it's a shift that isn't confined to developers or even to the project team. It's a shift that must be embraced by the entire organization.

Take a moment and think about what computer automation is. For the last 50 years, the programming discipline has been driven by the challenge of making products optimally efficient—easy and flexible to use, fast at recovering in the event of a problem, so business can move faster and avoid interruptions. Our attention—and all of our coder's learning and experience—has been around providing robust functionality to their customers.

Here are some of the catch phrases used by management:

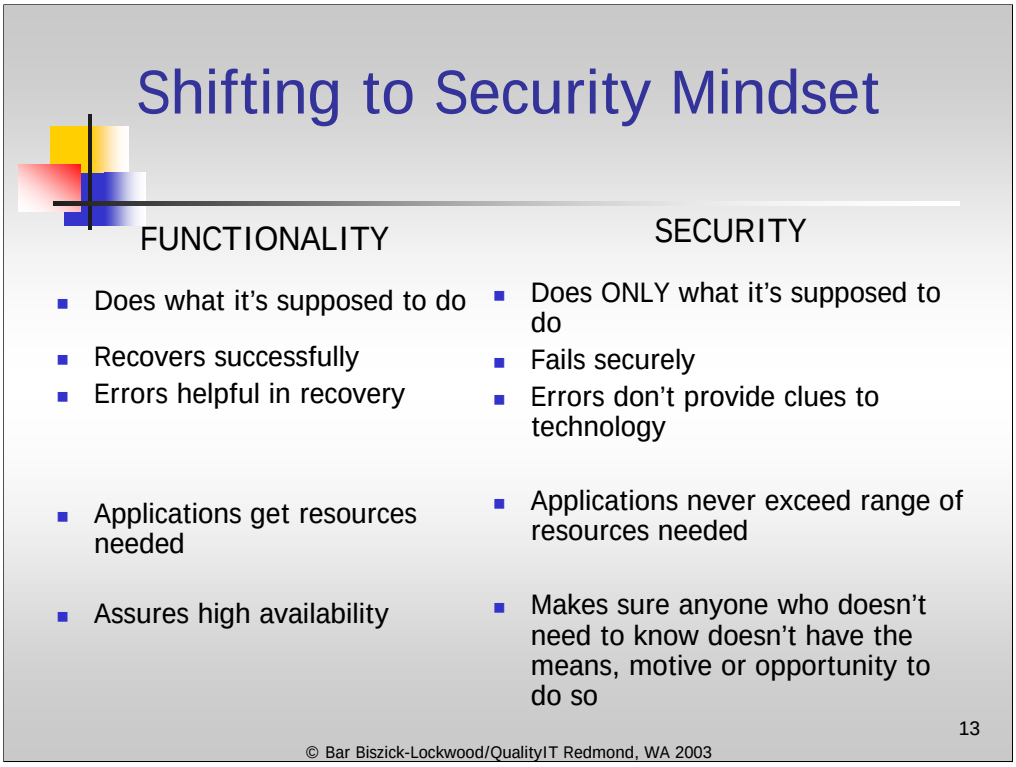
"Your job is to satisfy your customer"

"We need applications that insure high availability"

"Your job is to minimize impact to productivity by making sure a failed application can be brought quickly back on line."

These are some of the things our management has been drilling into our developers over the last 50 years. And our programmers have done a superb job at meeting these challenges.

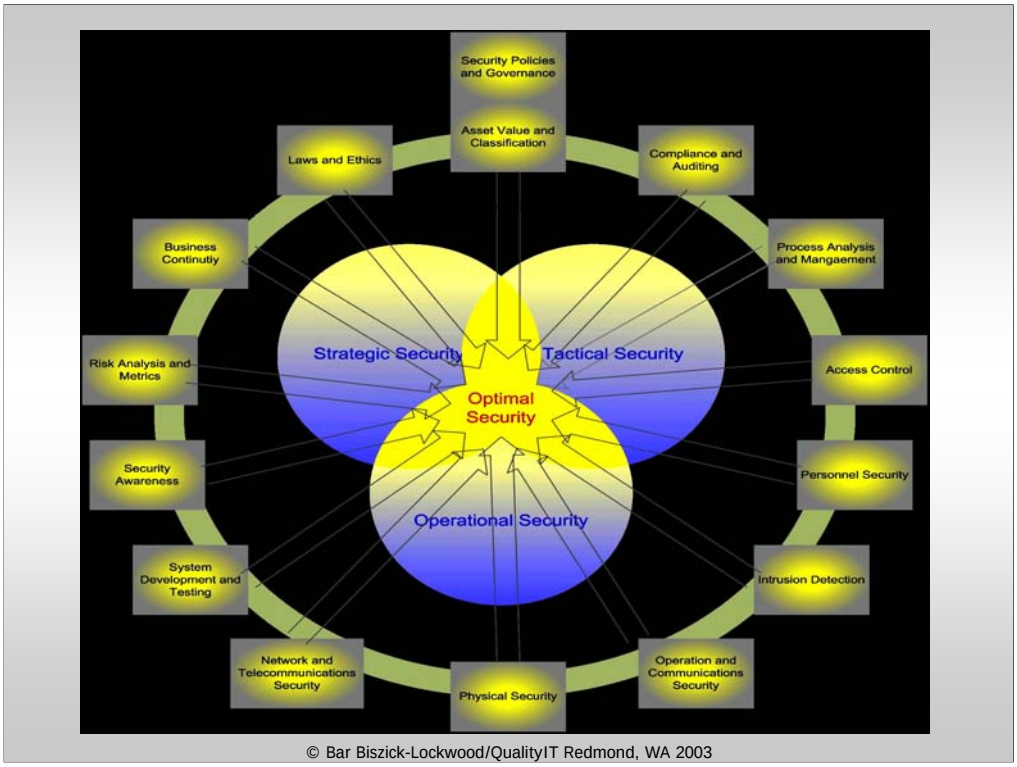
Now, we're asking them to relearn their job focus and to take on new responsibilities that fly in the face of what they've learned and experienced. Instead of satisfying the customer we're asking them to second guess management, point up potentialities and constrain customer desires. We're asking them to become the voice of ethical conscience for our organizations customer needs and management decisions-.Few programmers and technical leads are equipped with the necessary risk analysis skills and communications skills to meet this challenge. Let's see just how hard this shift really is.



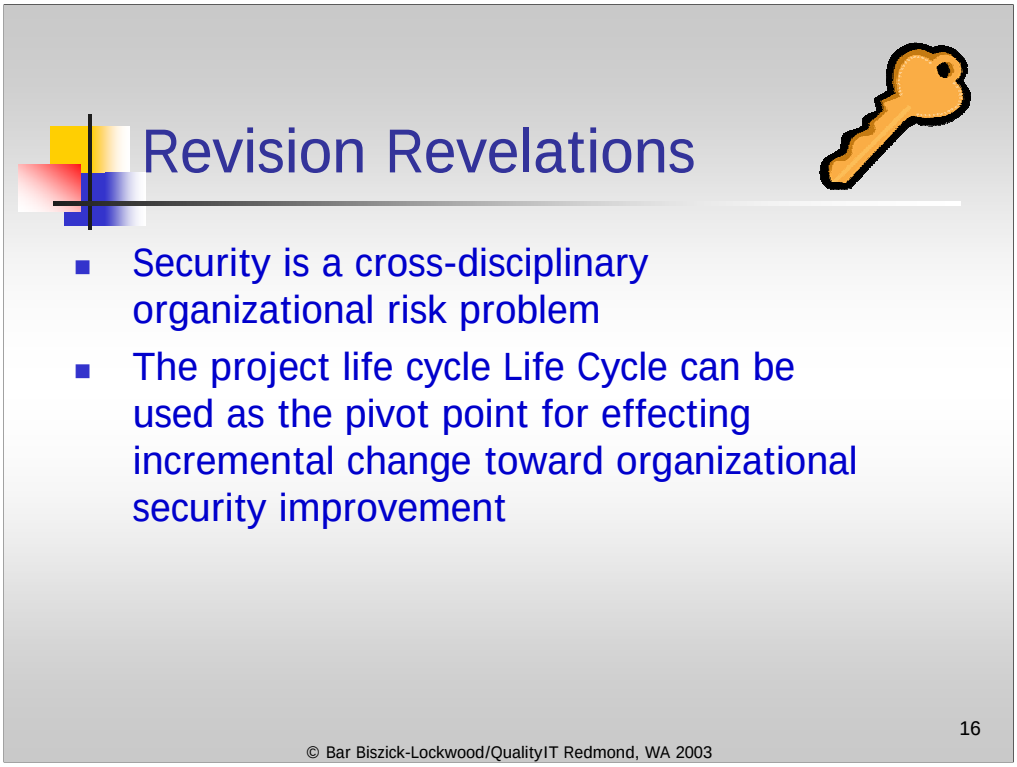
Over the past 5 years, developers have had to shift their thinking from builidng applications that do what they're supposed to do, to doing only what they're supposed to do; from recovering successfully to also failing securely; from making sure applications get the resources they need to making sure applications never exceed the range of resources needed.

These additional requirements come at a price. And it's a price our developers are not well equipped to express to management in way that appropriately informs them of the risks and consequences.

What we lack is a wide angle view of organizational risk and responsibility as it relates to technology. What would such a wide angle view look like?

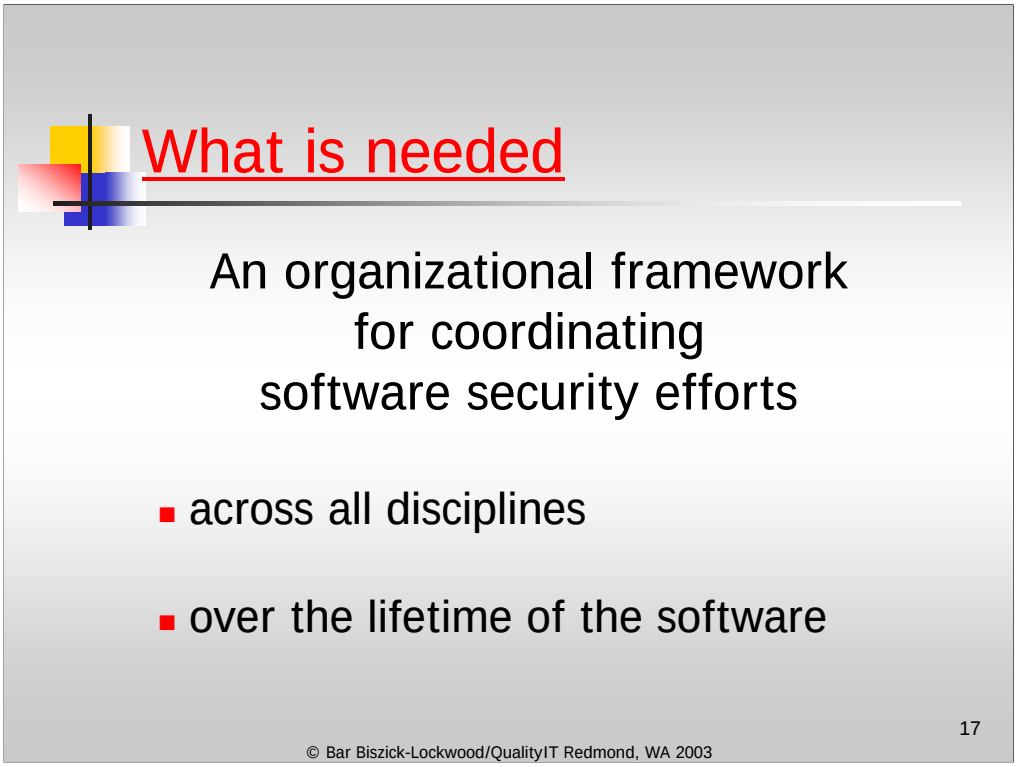


Perhaps something like this. Management needs to see optimal security as the effective coordination of security activities Almost none of these important security activities is the sole responsibility of a single group. Merely passing the hot potato from group to group is getting us nowhere. We need to put our attention on improving coordination between these groups.

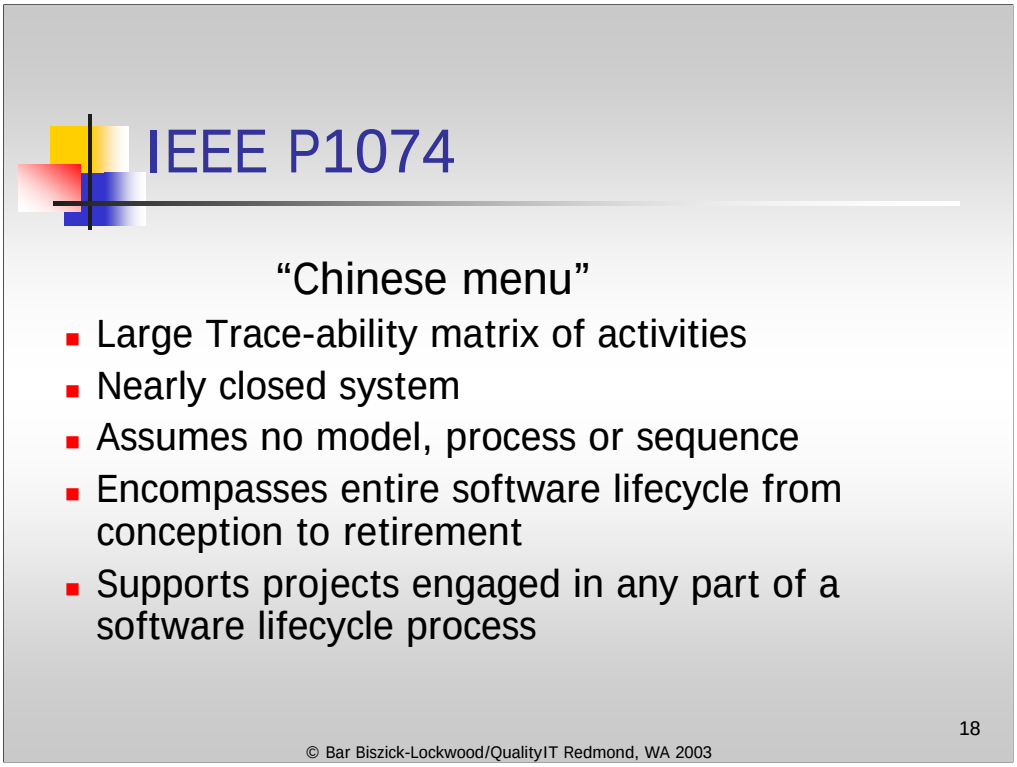


Recognizing that security is a cross-disciplinary organizational problem, we asked our selves when, during the normal course of business do these different groups come together to effect work?

The answer was, “on software projects.” Whether these project be when a company is creating a wholly new software product, upgrading an existing product, correcting (patching) a product, creating custom code to better utilize vendor products, or simply integrating a COTS product into an IT infrastructure, the Life Cycle of the projects can be used a the pivot point to effect incremental change toward improving information security.



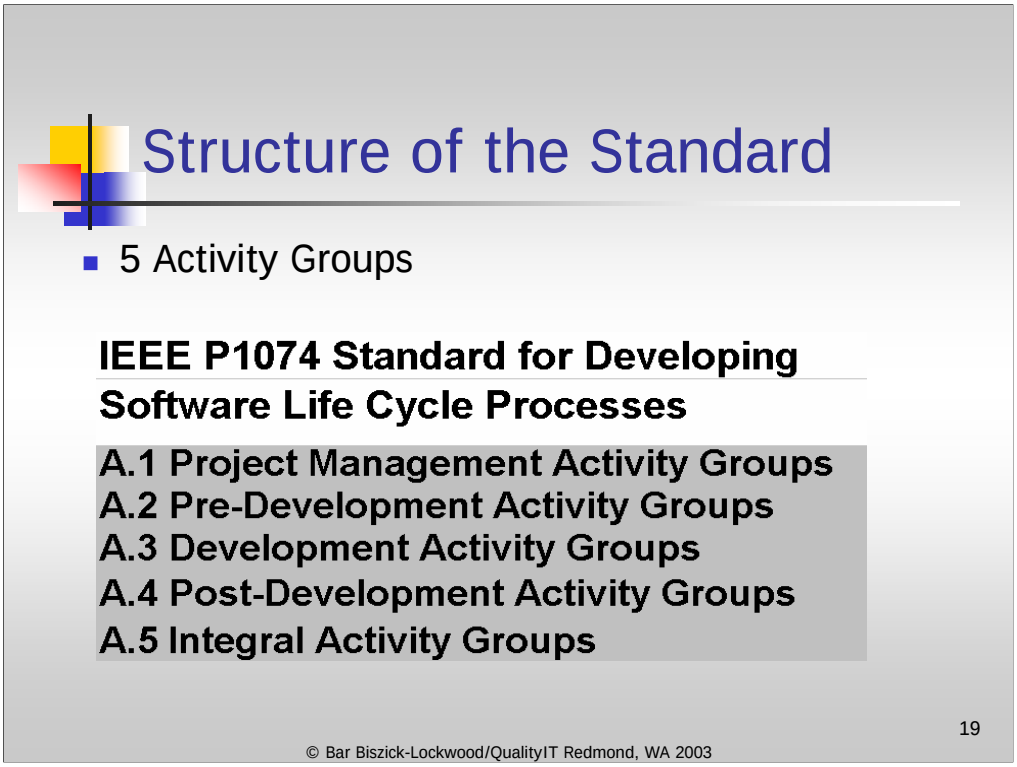
Now that model depicts how futile it is to try to compartmentalize security responsibility in an organization. Most enterprise security decisions depend on real time input from all three groups. So what we really need is an organizational framework for coordinating software security efforts across all disciplines and over the lifetime of software.



That's where IEEE P1074 comes in.

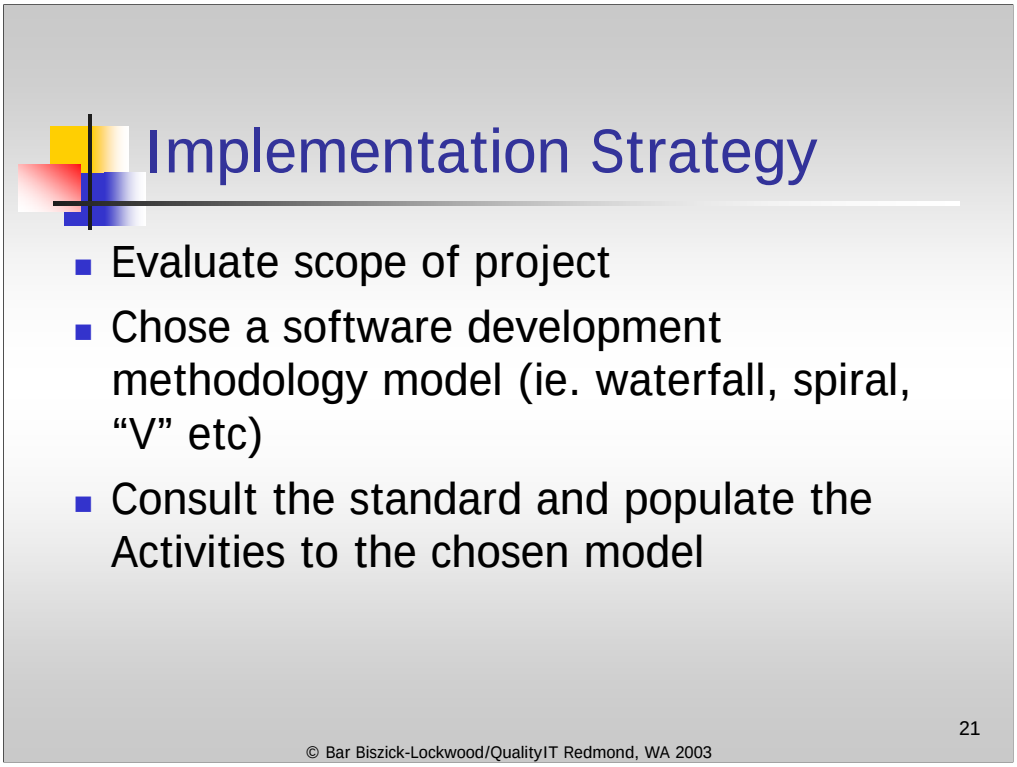
Think of it as a Chinese menu of all the recommended activities that should be executed during the lifetime of a software application.

As a general engineering process standard, it provides detailed activity descriptions describing the minimum effort required to produce a piece of quality software. It can accommodate any SDLC model, and any software programming method or style. The standard is organized into 17 interrelated activity groups, composed of more than 70 activities, including their specified inputs and outputs. There are a few mandatory activities, but for the most part an architect can choose a subset of activities appropriate for his project, and sequence the chosen Activities to create the appropriate quality process.

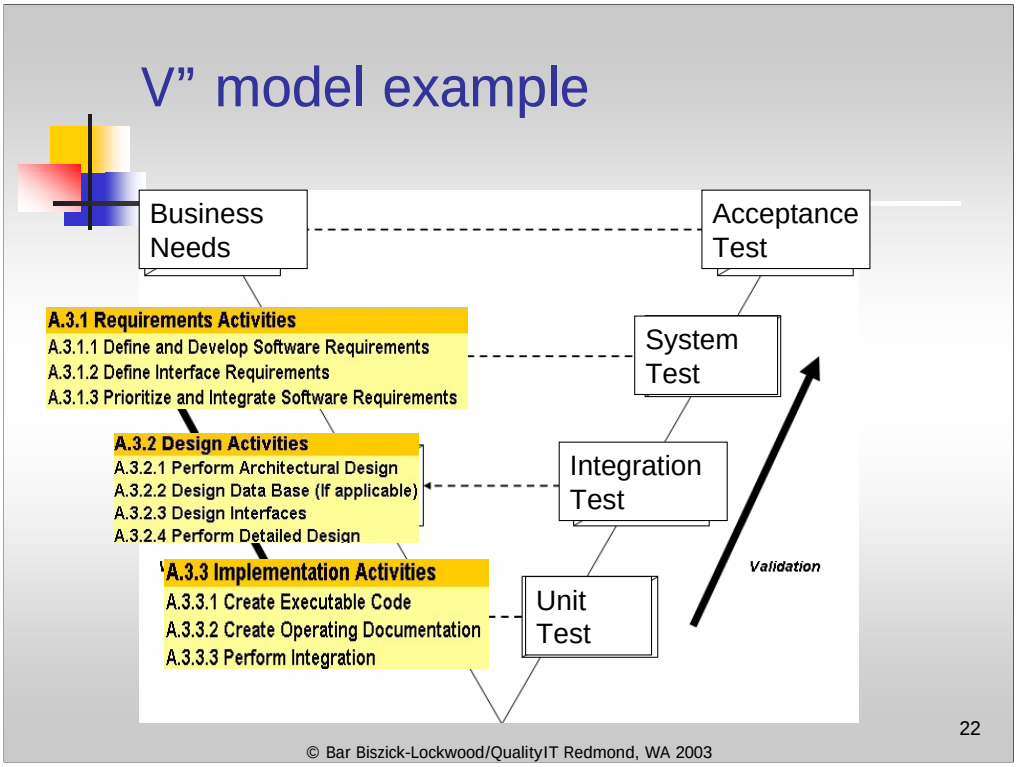


The standard presents 5 activity groups. Common activities in the Project Management group include Planning Project Management, Estimation and Resource allocations, where the Integral Activities group includes activities that tend to be called up many times during a life cycle, such as the Create Documentation and Conducts Reviews activities, for example.

Each Activity group is broken down into Activity Categories, which include detailed descriptions of specific activities.



You use the standard by evaluating the scope of the project, choosing a life cycle model, such as “spiral,” “v,” or “w” for instance, and populating the selected activities to the different phases of the model.



In this case, we've chosen a "V" model, and populated the Requirements, Design and Implementation phases with selected activities, for a project that begins after the Initiation Phase.

Originating Activity

A.3.3.3 Perform Integration

A.3.3.3.1 Input Information

Input Information	Source Activity
System Components	
SPMPI	Plan Project Management (A.1.2.7)
Integration Planned Information	Plan Integration (A.1.2.8)
Imported Software	Import Software (A.2.3.4)
Source Code (When Required)	Create Executable Code (A.3.3.1)
Executable Code	Create Executable Code (A.3.3.1)
Tested Software	Execute Tests (A.5.1.6)
Data Base (If Available)	Create Executable Code (A.3.3.1)
Stubs and Drivers (If Available)	Create Test Data (A.5.1.5)

A.3.3.3.2 Description

This Activity shall integrate the Data Base, Source Code (if required), Executable Code, and Stubs and Drivers, as specified in the Integration Planned Information, into the Integrated Software. Other necessary Executable Code, from the SLCP as defined in the SPMPI, shall also be integrated. If a system includes both hardware and software components, the system integration could be included as part of this Activity.

Prior to the distribution of the Integrated Software, the following Activities shall be invoked:

a) A.5.1.1, Conduct Reviews

b) A.5.1.6, Execute Tests

c) A.5.2.2, Perform Configuration Control

A.3.3.3.3 Output Information

Output Information	Destination Activity
Integrated Software	Execute Tests (A.5.1.6)

Each activity is structured like a quality gate. Inputs are gathered and transformed during the activity to yield an output. In this case, the “Perform Integration” activity combines the listed inputs to yield a piece of Integrated Software. This output is sent to a destination activity, “Execute Tests.”

Receiving Activity

A.5.1.6 Execute Tests

A.5.1.6.1 Input Information

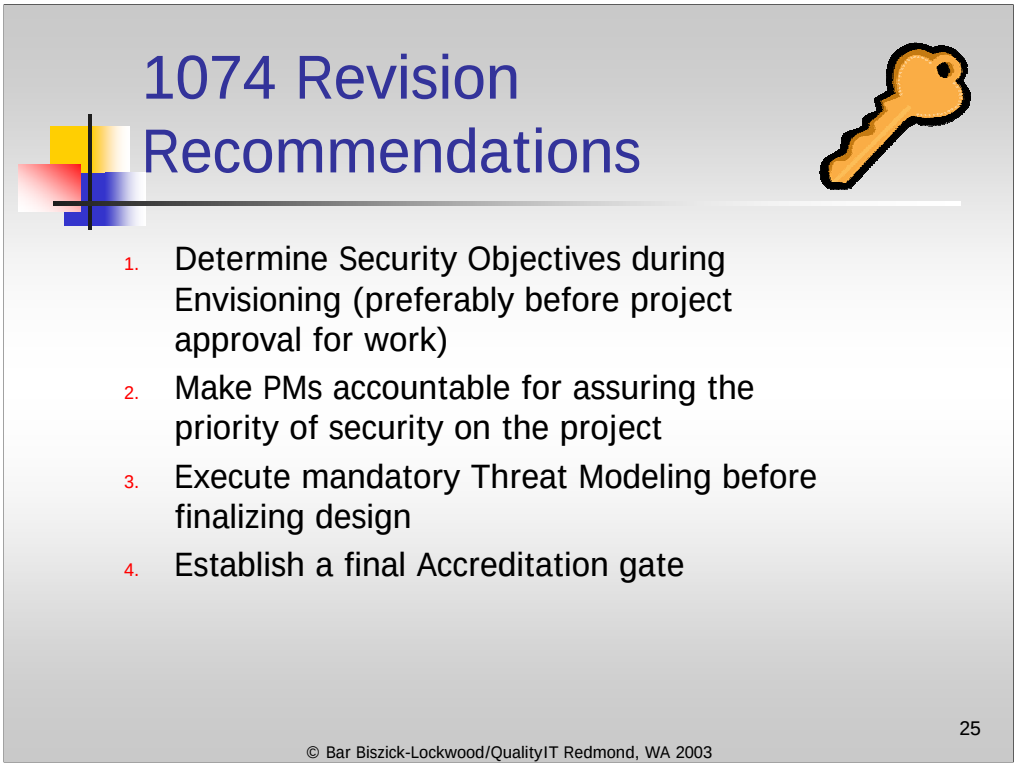
Input Information	Source Activity
Test Environment Components	
Evaluation Planned Information	Plan Evaluations (A.1.2.1)
Imported Software	Import Software (A.2.3.4)
Executable Code	Create Executable Code (A.3.3.1)
Integrated Software	Perform Integration (A.3.3.3)
Test Procedures	Develop Test Procedures (A.5.1.4)
Test Data	Create Test Data (A.5.1.5)
Stubs and Drivers (If Available)	Create Test Data (A.5.1.5)

A.5.1.6.2 Description

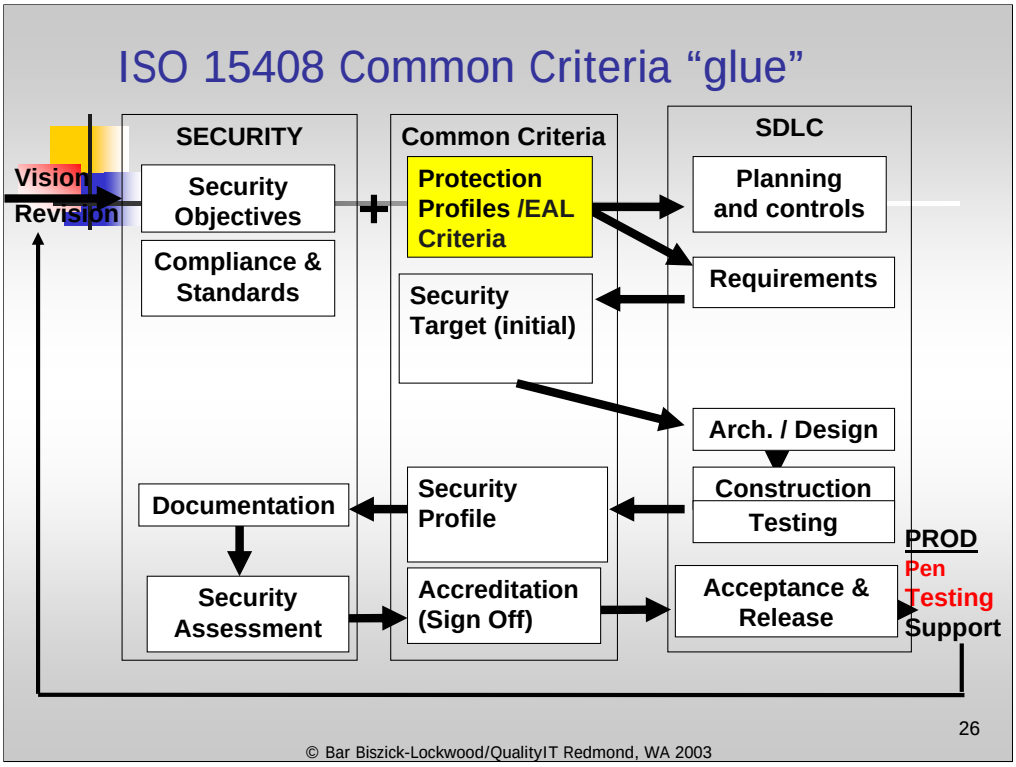
This Activity shall configure the Test Environment Components as required by the Test Procedures. Tests shall be conducted on Executable Code units/modules/components, Integrated Software, and the full system using Test Data and the associated Test Procedures, in accordance with the Evaluation Planned Information. This Activity could be iterative, with several Instances performed during the life of the software. Not all

The receiving activity, in this case the Execute Tests activity, combines the Integrated Software with other inputs to yield a pieces of Integrated, Tested software, and so on.

So in essence, IEEE P1074 is a huge traceability matrix, representing an almost closed system of evolving project artifacts. Some artifacts will be processed only one time on a project. Others will invoke activities again and again during their evolution. For instance, almost every activity will send outputs to activities related to review, documentation and configuration management.

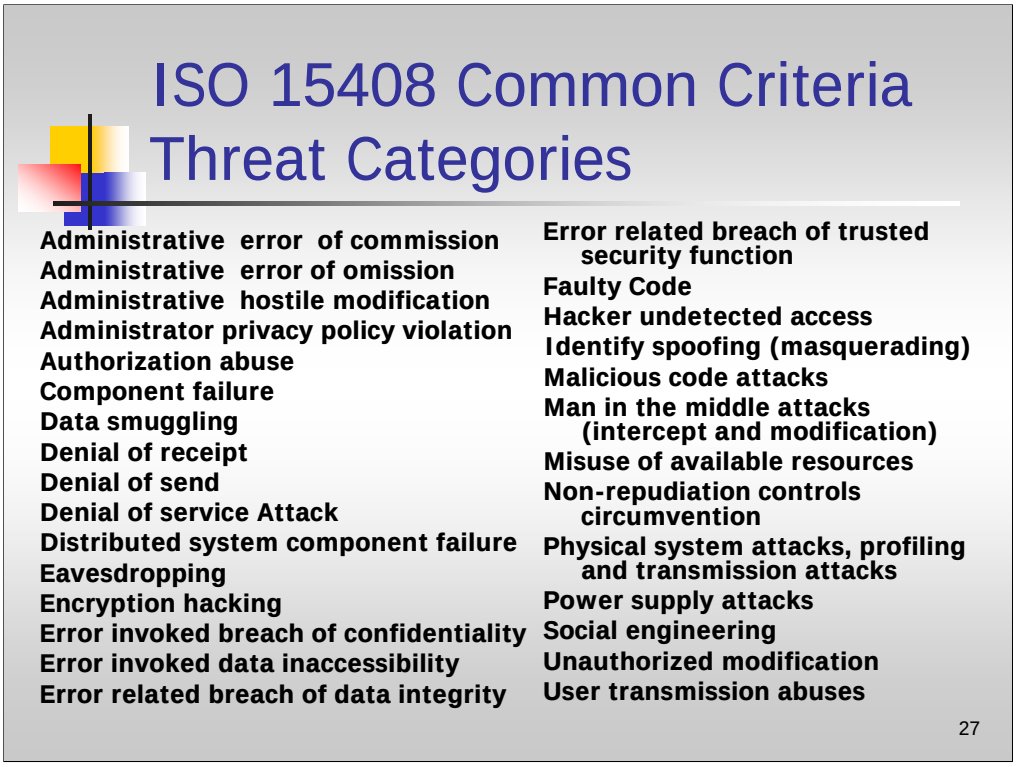


Since our goal was to elevate the visibility and priority of security activities in the life cycle to better reflect current engineering needs, we chose to enhance the standard with the addition of two new life cycle activities, “Determine Security Objectives,” and “Confirm Security Accreditation,” and to enhance “Develop Architectural Design” with specific inputs and guidance that makes Threat Modeling a mandatory activity. We also recommended that the activity called “Plan Project” include guidance that makes clear a PMs responsibility for determining the level of security to be applied both the product, as well as to the project, be determined early on the project, and that they are responsible for assuring the appropriate priority of security based on this knowledge.

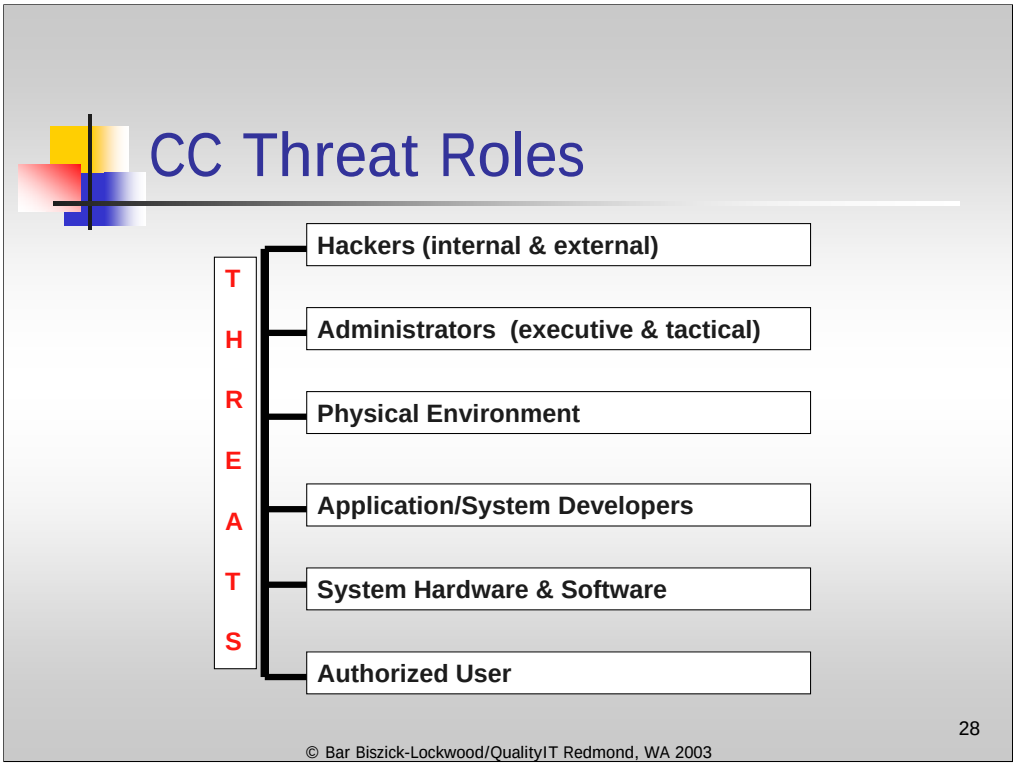


We mined the assets of ISO/IEC 15408 which can help project teams to determine minimum security requirements for their projects. Typically regarded as separate enterprise silos, Common Criteria assets and principles can act as the “glue” that binds the Enterprise Security Life Cycle process to the Software Development Life Cycle process.

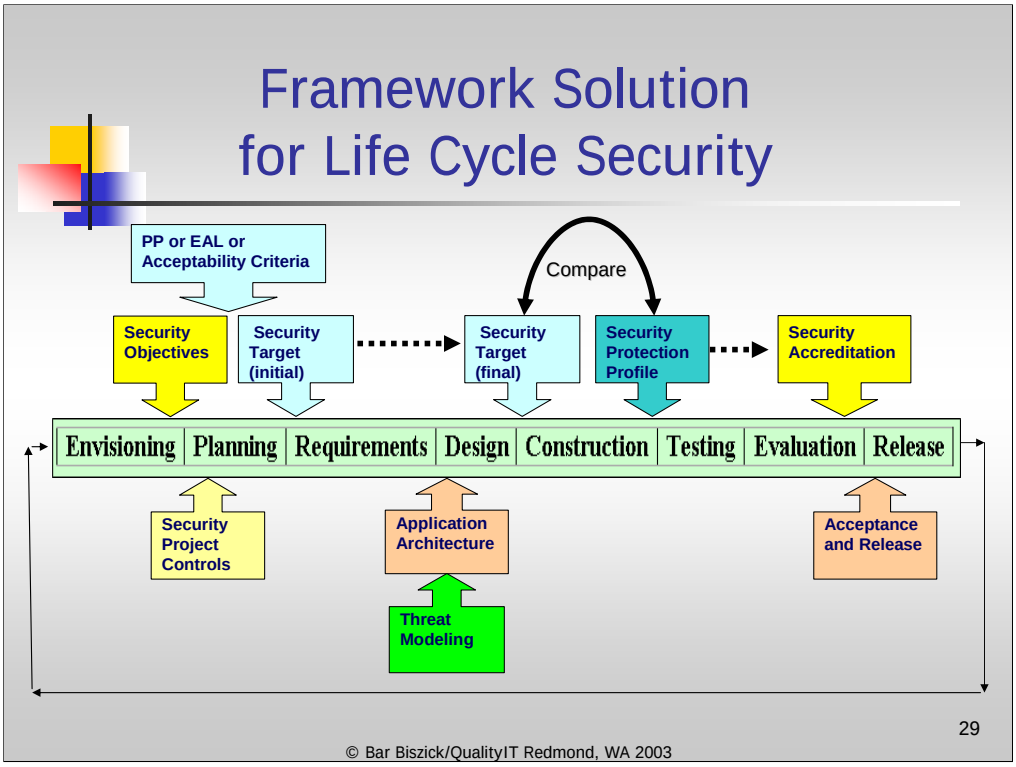
Common Criteria Protection Profiles or EAL criteria can be applied as reference assets on projects whose product falls into the same class. For instance, there is a Protection Profile enumerating the requirements used to build routers. So if you’re building a router, you can reference the Protection Profile for routers, to make sure you don’t overlook any requirements deemed important to building a router with good security. Similarly, if you are building any kind of system that requires role based access control, there is a Role Based Access Control Protection Profile rated at EAL 2, which could act as the baseline security requirements for your product. These can be identified early in the project, helping to determine the programming and testing resources and skills that will be needed for the product, even before functional requirements gathering begins. This can reveal security constraints that can affect whether or not the product investment is viable, whether more highly skilled coders and testers will be needed, or whether special security testing products might need to be purchased.



Common Criteria guidance can also guide Threat Modeling. The Common Criteria lists these categories of common threats, and provides detailed descriptions of the conditions under which they might manifest.



All threats must have a threat agent. Common Criteria also defines a number of common threat roles as well that are useful to consider when doing Threat Modeling.



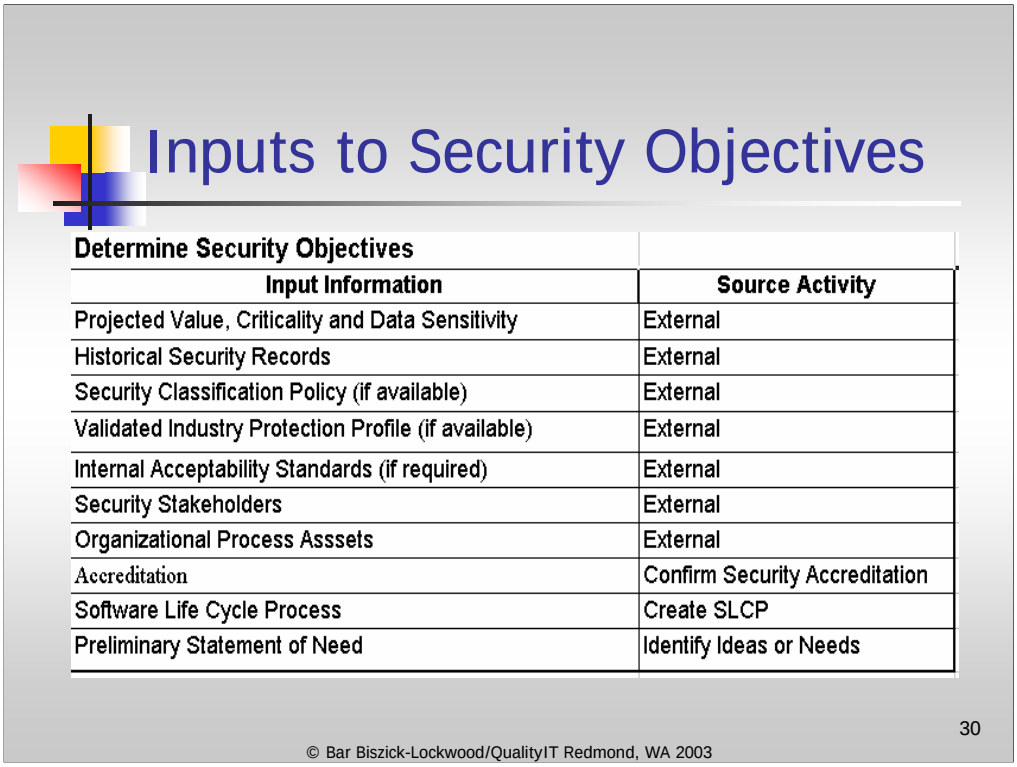
The recommendations we are making to update this general Life Cycle Process standard are as follows:

Security requirements are usually gathered late in the requirements phase. We are suggesting instead that enterprise security objectives and guidance from Common Criteria be identified prior to authorizing the project, so the project manager has as much information as possible to properly plan the project and determine the relative priority of security on the project.

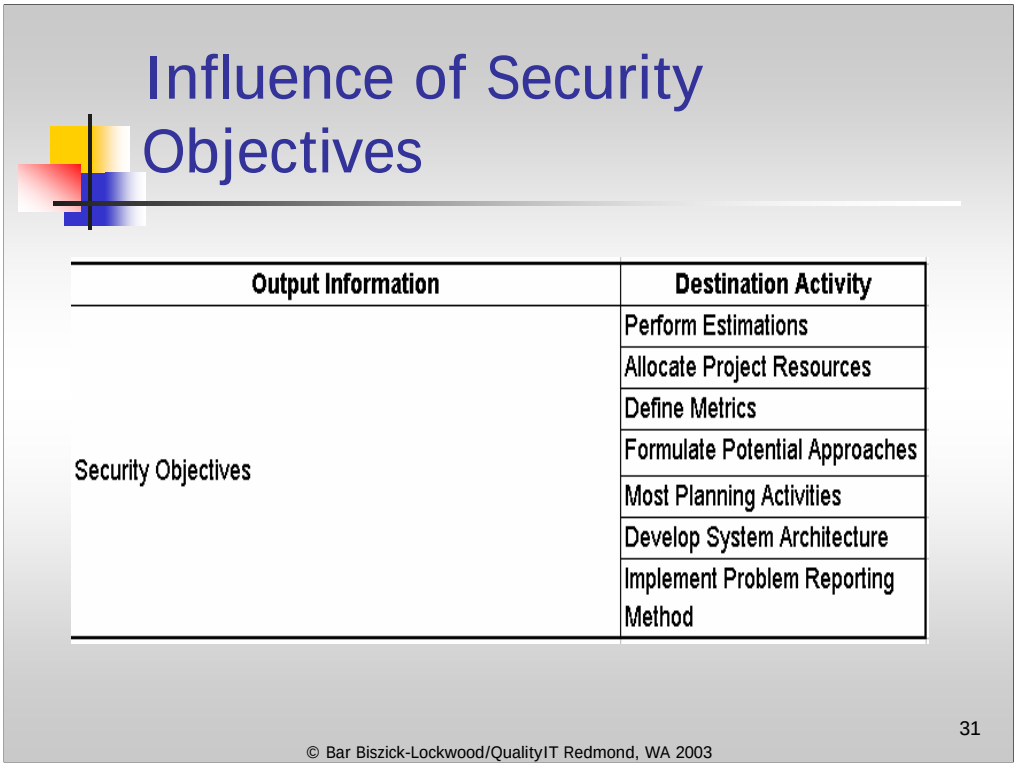
This set of security requirements makes up the initial security target, which is enhanced through Threat Modeling during the design phase to yield the final set of security requirements, including those for built in security functionality.

These security requirements can be managed as part of the total requirements set, but should probably be easily separated from them since the testing results that come from comparing to what is actually built will be provided to the security officer at the end of the evaluation period. This officer should be outside the project team, and independent of the sponsor. Base on the testing results the officer will either authorize the product for use, or decline such accreditation. Their opinion will be provided to the sponsor, who will act on the judgment, which may include returning a substandard product to the engineering team for security remediation or accepting the risk and releasing the product despite security flaws.

Now the “Confirm Security Accreditation” activity which anchors the other end of the SDLC, does not suggest the need for formal product accreditation. Accreditation simply means “authority to operate.” If external criteria, such as Protection Profiles, cannot be found to provide the initial product security requirements, then an organization would define acceptability criteria during the Security Objectives activity, so that the organization’s expectations as regards product security are made clear to the project team. The security officer’s primary goal would be to report to management, whether or not this criteria has been met.



The most important change to the standard is the addition of the “Determine Security Objectives” activity which is placed at the top of the project. Many factors bear on making good judgments about the amount and kind of security controls that will be applied to a product. You have to consider the value of the data, the criticality of the system, your breach history, the system and people security classification policies, as well as past accreditation history and any Industry Protection profiles or EAL levels that might be available.



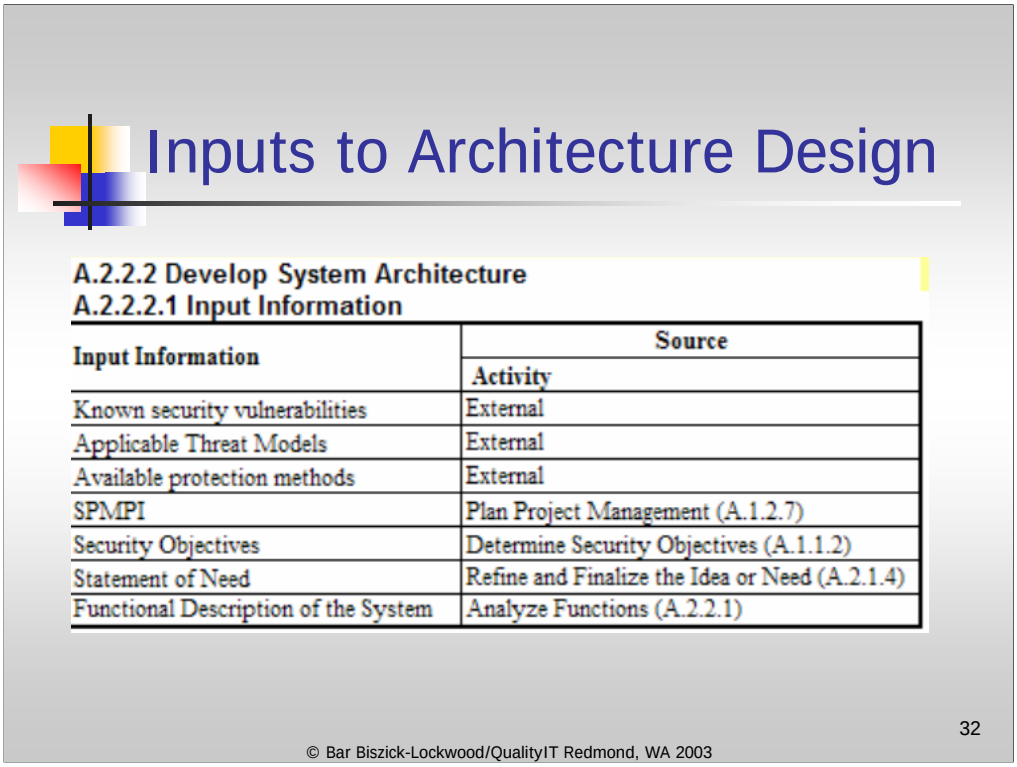
Combining these inputs will yield your initial Security Objectives, and this output will be provided to the project manager as inputs to activities such as “Perform Estimations,” and “Allocate Project Resources.” With the knowledge of enterprise security objectives, project managers can make good judgments about what level of security an organization expects from its software, and therefore how best to staff the project.

So determining Security Objectives early in the project is essential to ensuring no project will go forward that could undermine critical security infrastructure, or risk proprietary assets or private customer data. Security Objectives will

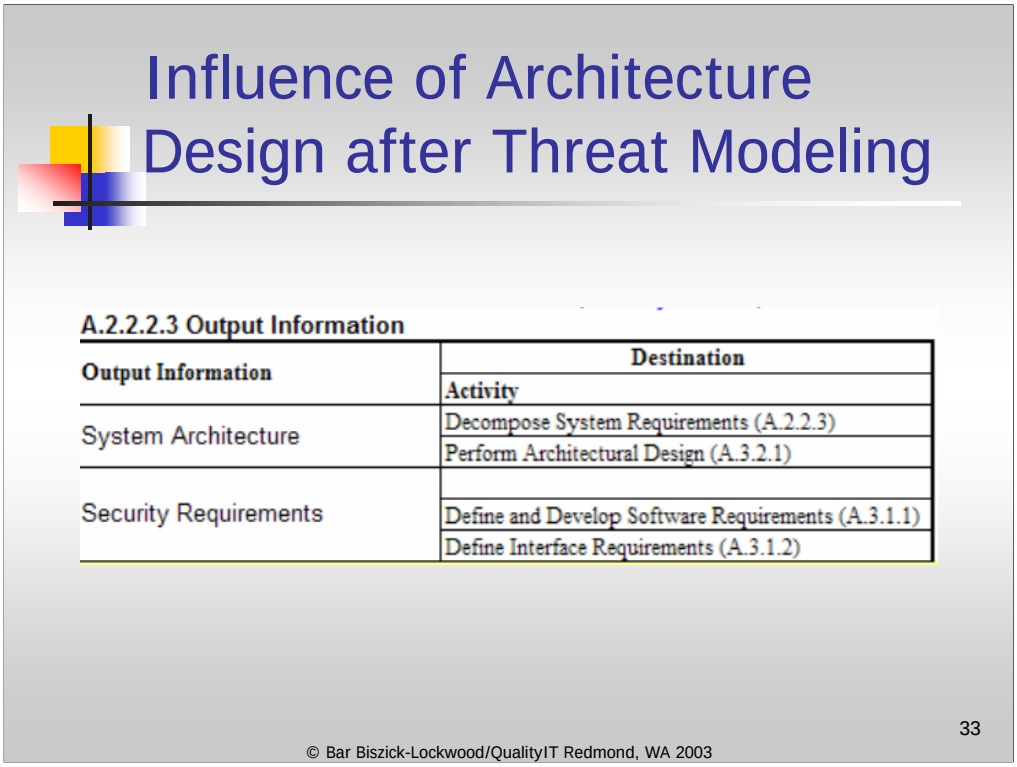
-determine the security feasibility of the project

-determine how much security should be applied to the project

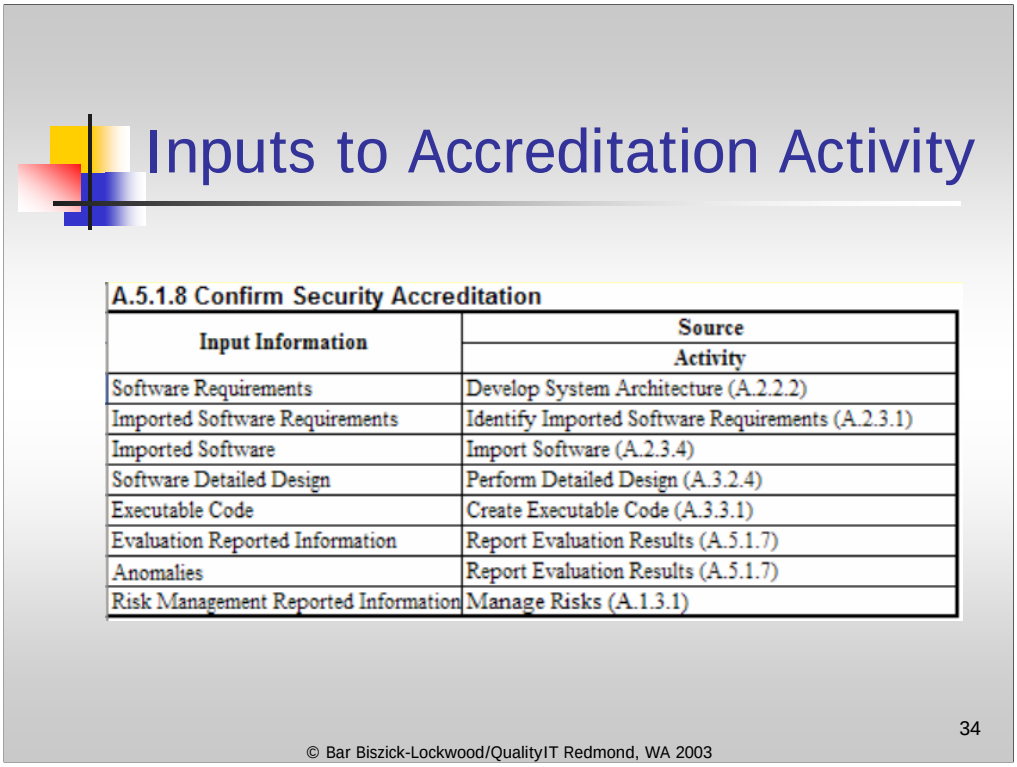
-determine how much security should be built into a product



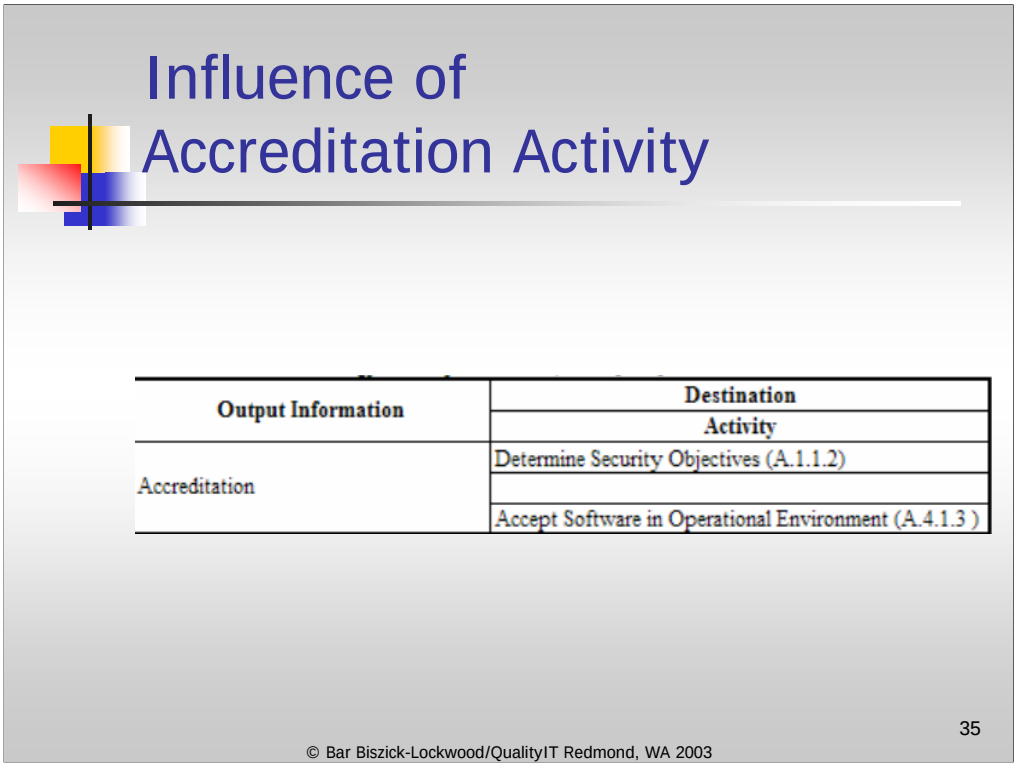
The Architecture Design must now take into account not only the requirements defined in the security objectives, but Known Security Vulnerabilities inherent in the infrastructure or technology being used, the results of applicable threat models and the available protection methods. Descriptive guidance that accompanies this activity clearly states that Threat Modeling should be a mandatory activity, and should be completed before the design is finalized.



The output from the “Develop Architectural Design” yields two artifacts—the design itself, and the new security requirements discovered during Threat Modeling. These requirements are added to the growing set of product requirements, completing the subset that addresses security.

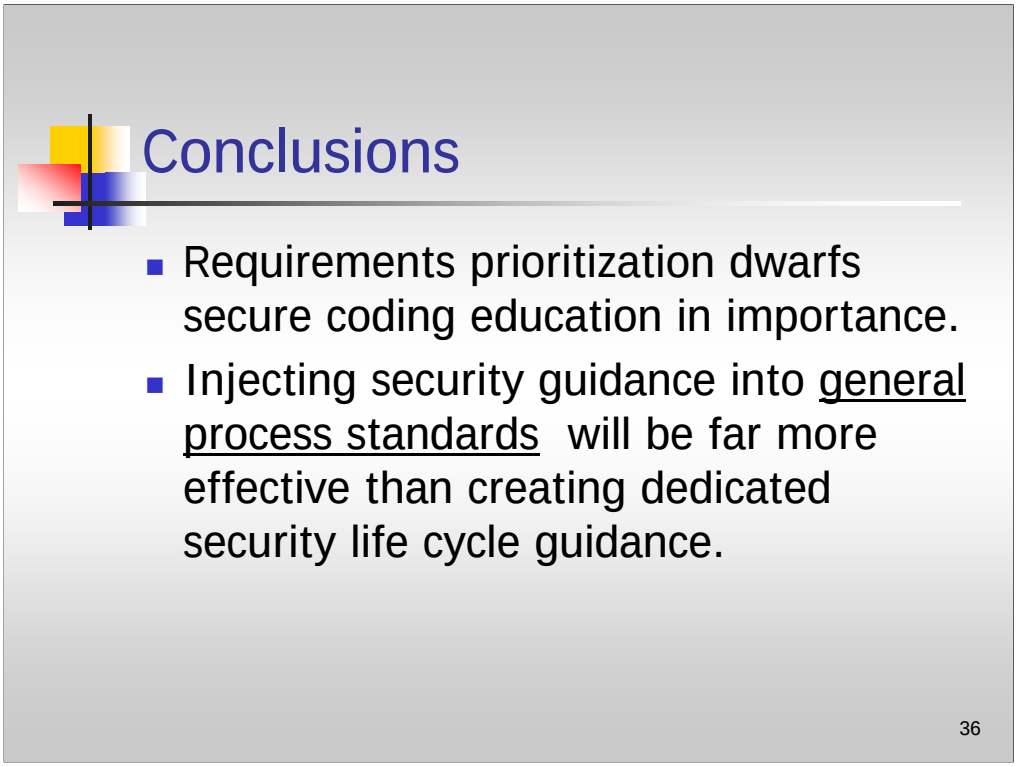


The “Confirm Security Accreditation” will take into account all the artifacts required in testing, and their results. The results of security testing, along with unexplained anomalies and any identified security risks will be provided to the security officer for consideration.



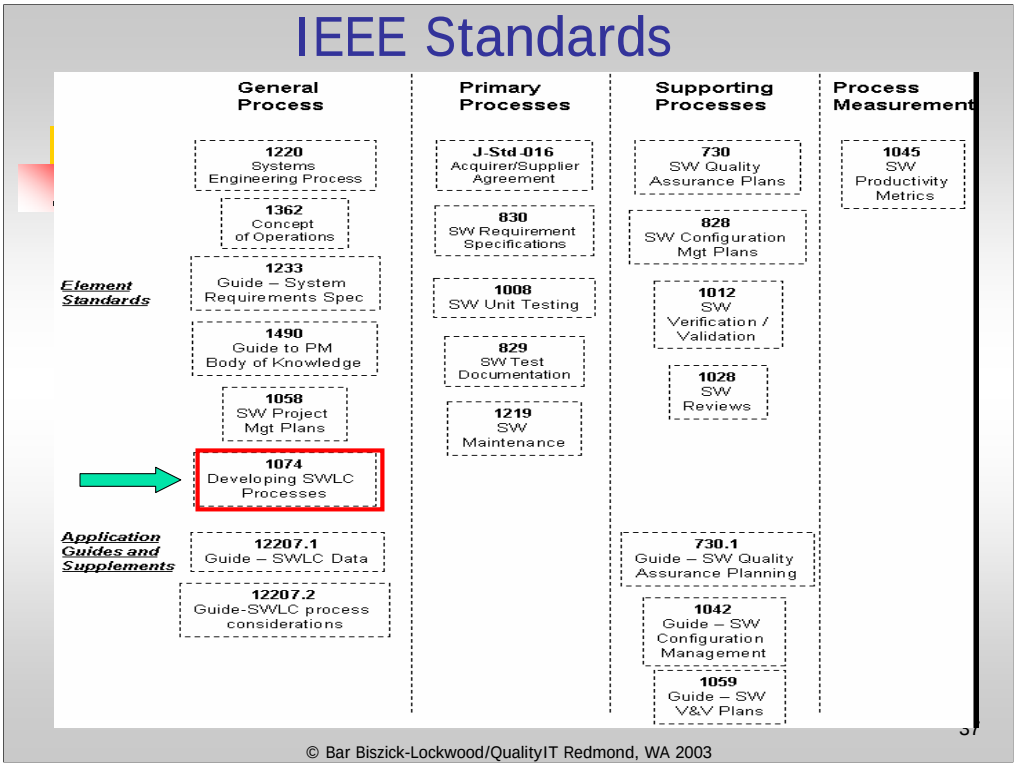
The output from the “Confirm Security Accreditation” activity is an accreditation recommendation which is sent to the activity used to accept the software in the operational environment. Practically speaking, this would be a judgment—either positive or negative—provided by a security officer, as to whether the product meets pre-specified security acceptability levels.

In the “Accept Software in Operational Environment” activity, management would consider the security officer’s judgment to determine if they accept any risk still remaining in the product, or would refer it back to the project team for further work.

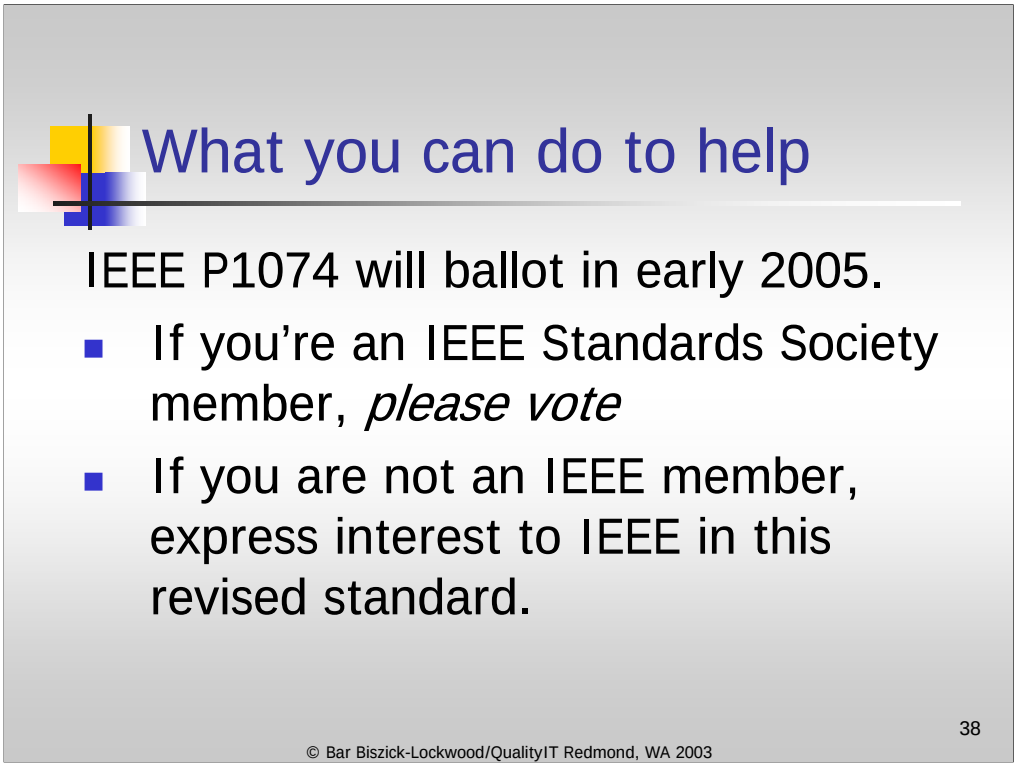


Compared to the current security improvement trend, we believe requirements prioritization dwarfs secure coding education in importance when it comes to building secure software. That is not to say that coding education is not needed, but that by using this framework educational needs and deficiencies specific to the project will be identified early enough in the project to address.

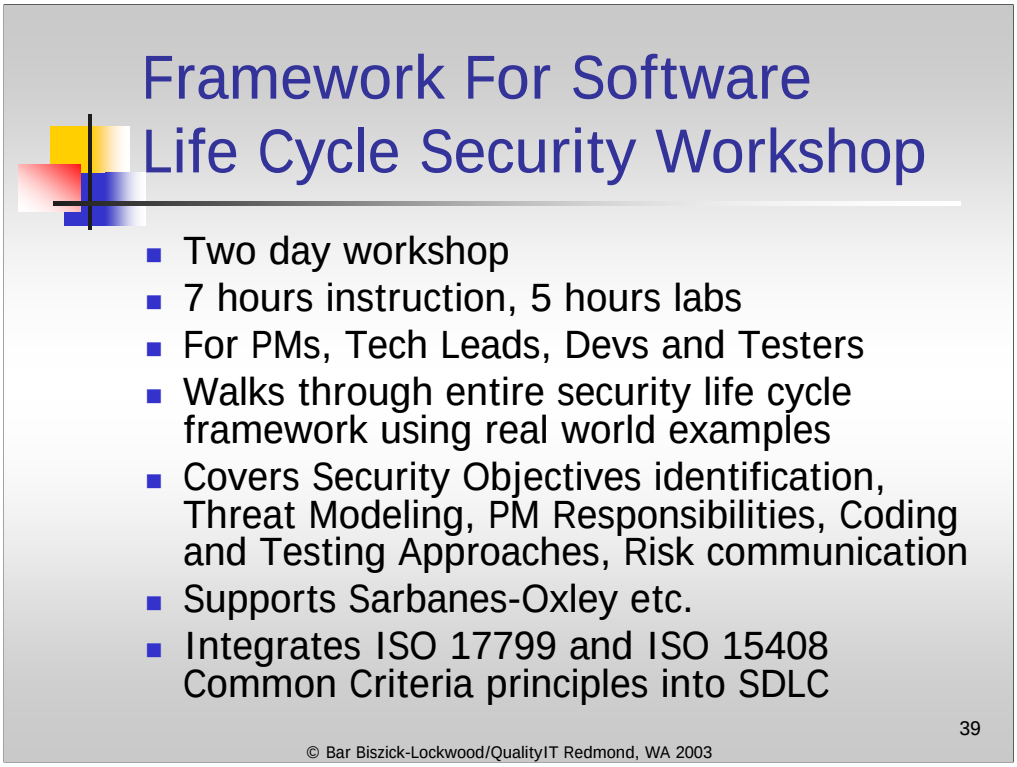
We also believe that the lack of general guidance in addressing security on projects is severe, but that trying to build separate set of security guidance in this area is counterproductive. Instead, we should be looking for ways to elevate the visibility and priority of security in general engineering standards like IEEE P1074.



The value to doing so is obvious. If we are successful in leveraging security guidance into this elemental IEEE standard, it will have a trickle down effect on all other general engineering standards that will have to come into alignment with it. If instead we continue to regard security process as something separate and extra in the engineering process, we will create bodies of guidance that will be overwhelming to the standards user, and will likely conflict. We can ensure the general engineering practitioner understands how security fits into their world by simply updating the accepted engineering standards.



Changes to IEEE P1074 are coming and if you're a member of IEEE I hope you'll weigh in with your vote and support it. If you are not an IEEE member I hope you'll express interest in the revision of this standard.



For those that are interested, I've created a two day workshop that guides project teams through the framework.

Contact Info

- Bar Biszick, cisa, cissp, csqa
  - IT Quality and Security Assurance
  - 206-388-3333
  - mailto:barbis@qualityit.net
  - http://www.qualityit.net/mambo
    - Supporting Rescources\White Papers:
      - Justification for Elevating the Visibility and Priority of Security Activities in in IEEE P1074
      - Recommended Information Security Assurance Additions for the Revised P1074 Standard

Note: Bar Biszick-Lockwood is not a representative of IEEE

40

You can contact me at the address listed, and find a copy of this presentation, the business case and the specific revision recommendations on my website.

Thank you.