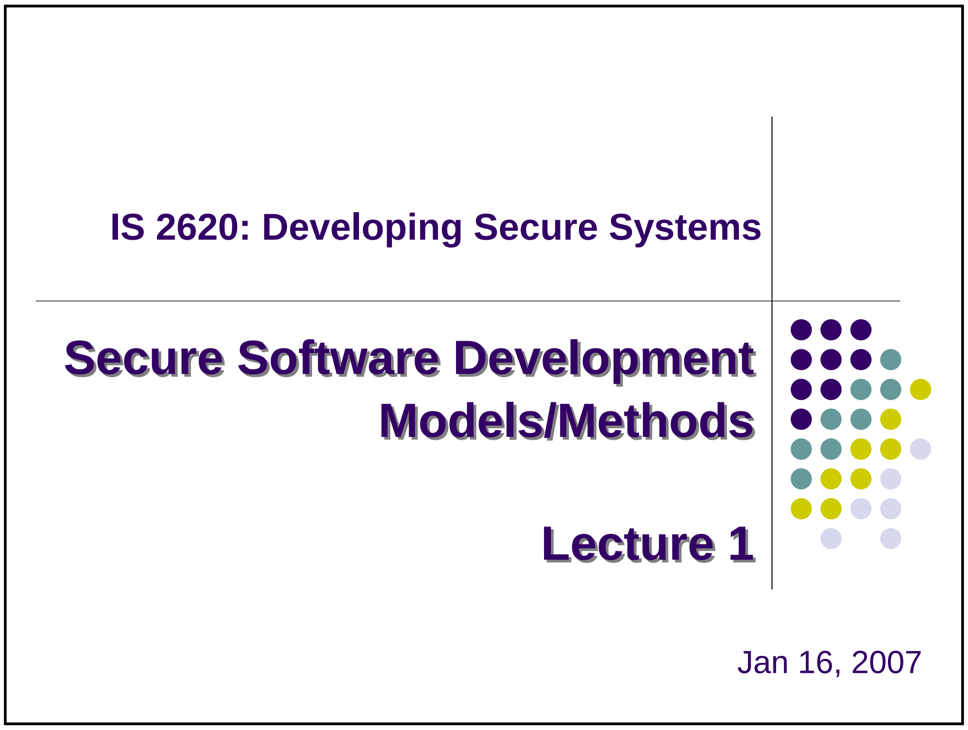
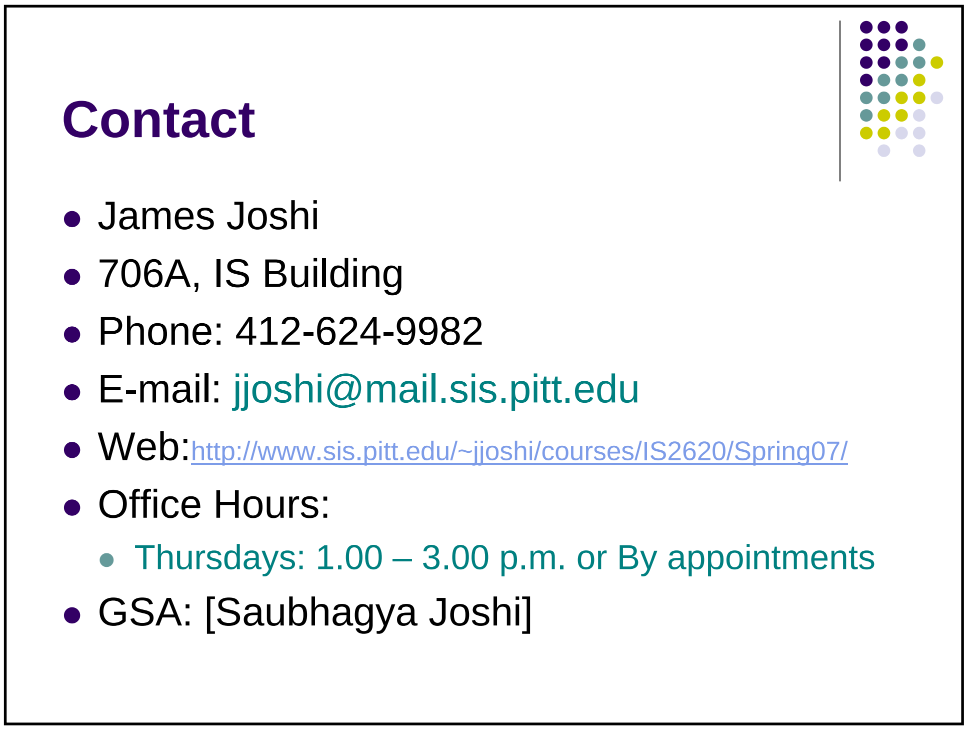


IS 2620: Developing Secure Systems

Secure Software Development Models/Methods

Lecture 1

Jan 16, 2007



Contact

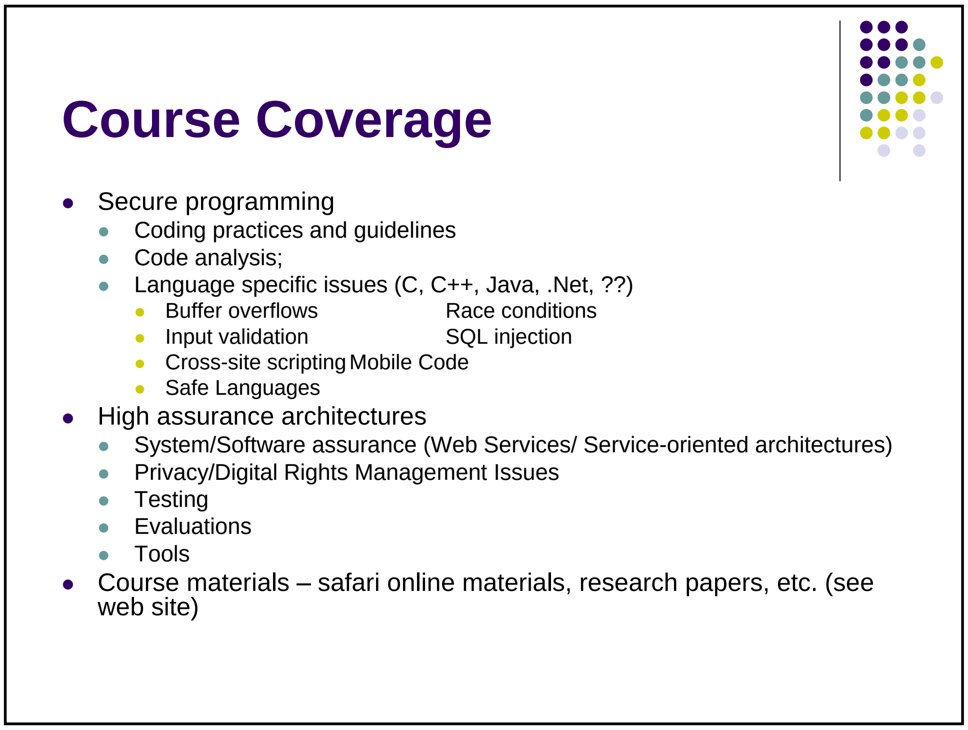
- James Joshi
- 706A, IS Building
- Phone: 412-624-9982
- E-mail: jjoshi@mail.sis.pitt.edu
- Web:http://www.sis.pitt.edu/~jjoshi/courses/IS2620/Spring07/
- Office Hours:
  - Thursdays: 1.00 – 3.00 p.m. or By appointments
- GSA: [Saubhagya Joshi]

Course Objective

- The objective of the course
  - To learn the principles and practice of secure information system design
    - Life cycle models/ security engineering principles
  - To learn about how to implement secure and high assurance information systems
    - Secure programming (e.g., C, C++, Java)
  - To learn about the tools and techniques to conduct testing and analysis of systems

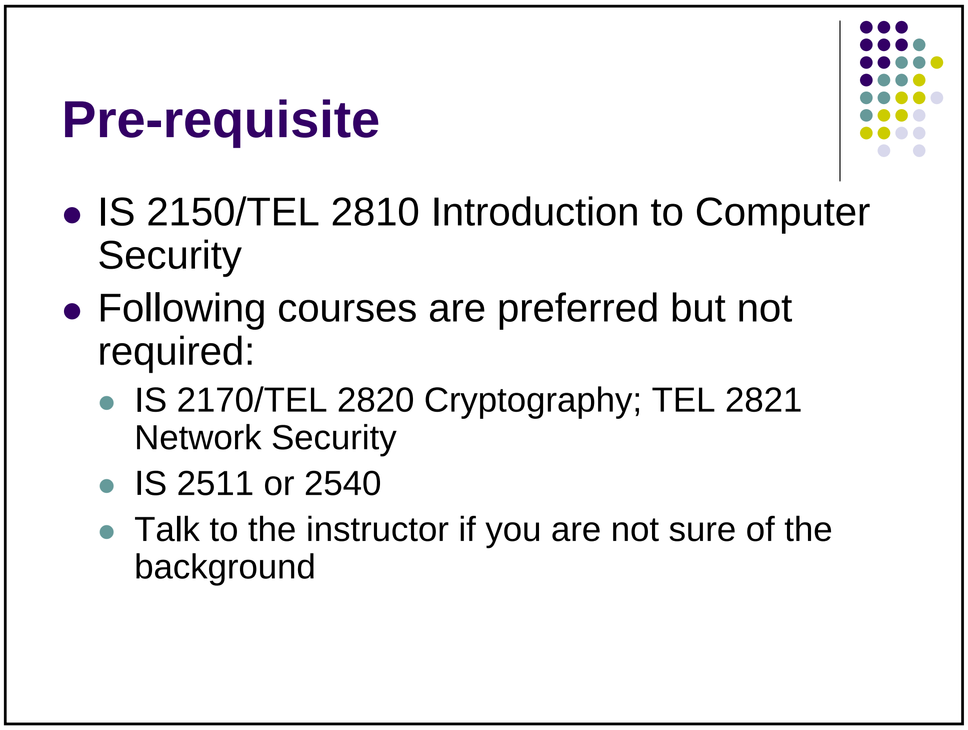
Course Coverage

- Secure software development process
  - Security Engineering/Lifecycle models
    - Software Development Models
    - Capability Maturity Models and Extensions
    - Trustworthy computing Security Engineering Lifecycle
  - Secure Design/Implementation Principles
    - Systems / software
    - Formal methods
      - UMLSec, Model Checking (code, protocols)



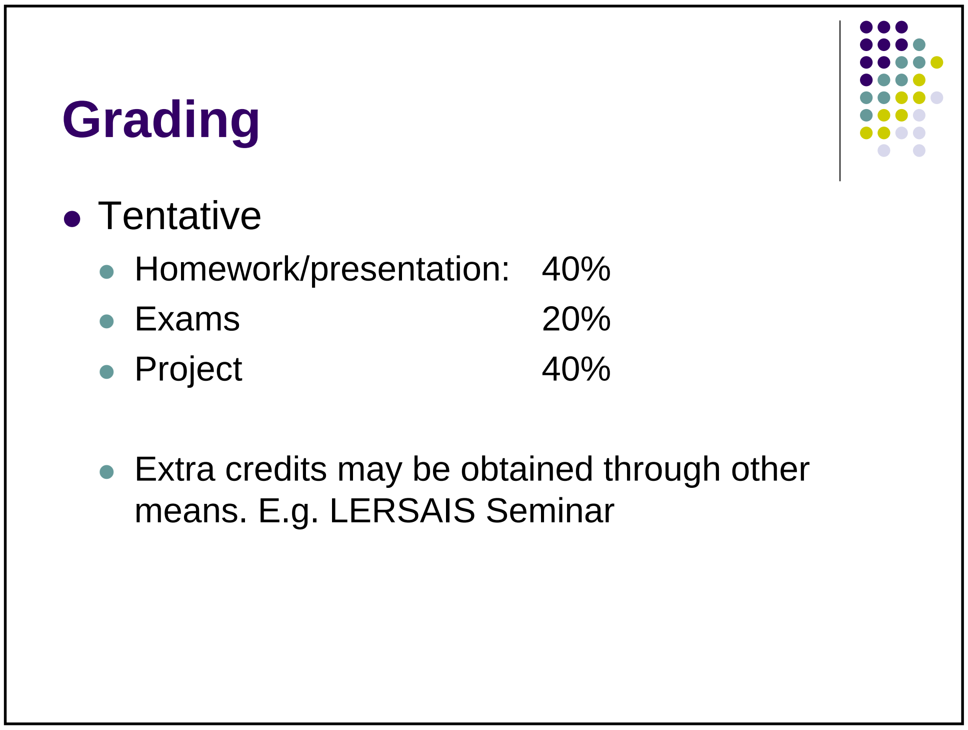
Course Coverage

- Secure programming
  - Coding practices and guidelines
  - Code analysis;
  - Language specific issues (C, C++, Java, .Net, ??)
    - Buffer overflows Race conditions
    - Input validation SQL injection
    - Cross-site scripting Mobile Code
    - Safe Languages
- High assurance architectures
  - System/Software assurance (Web Services/ Service-oriented architectures)
  - Privacy/Digital Rights Management Issues
  - Testing
  - Evaluations
  - Tools
- Course materials – safari online materials, research papers, etc. (see web site)



Pre-requisite

- IS 2150/TEL 2810 Introduction to Computer Security
- Following courses are preferred but not required:
  - IS 2170/TEL 2820 Cryptography; TEL 2821 Network Security
  - IS 2511 or 2540
  - Talk to the instructor if you are not sure of the background

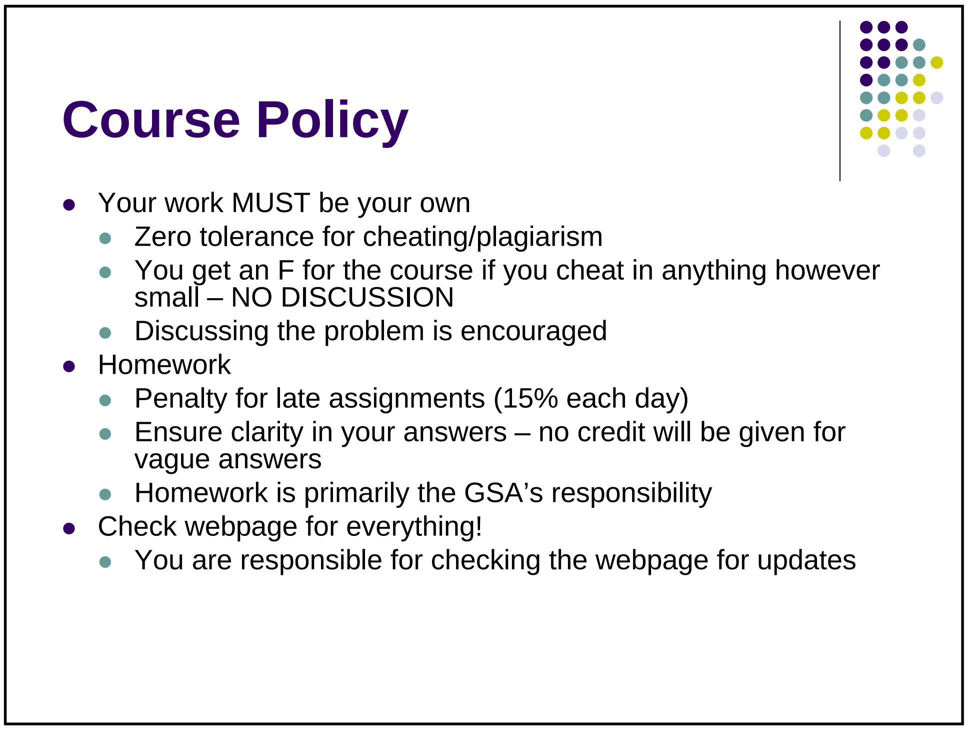


Grading

- Tentative

Homework/presentation:	40%
Exams	20%
Project	40%

  - Extra credits may be obtained through other means. E.g. LERSAIS Seminar



Course Policy

- Your work MUST be your own
  - Zero tolerance for cheating/plagiarism
  - You get an F for the course if you cheat in anything however small – NO DISCUSSION
  - Discussing the problem is encouraged
- Homework
  - Penalty for late assignments (15% each day)
  - Ensure clarity in your answers – no credit will be given for vague answers
  - Homework is primarily the GSA's responsibility
- Check webpage for everything!
  - You are responsible for checking the webpage for updates

Some Terms: Process

- Process
  - A sequence of steps performed for a given purpose [IEEE]
- Secure Process
  - Set of activities performed to develop, maintain, and deliver a secure software solution
  - Activities could be concurrent or iterative

Process Models

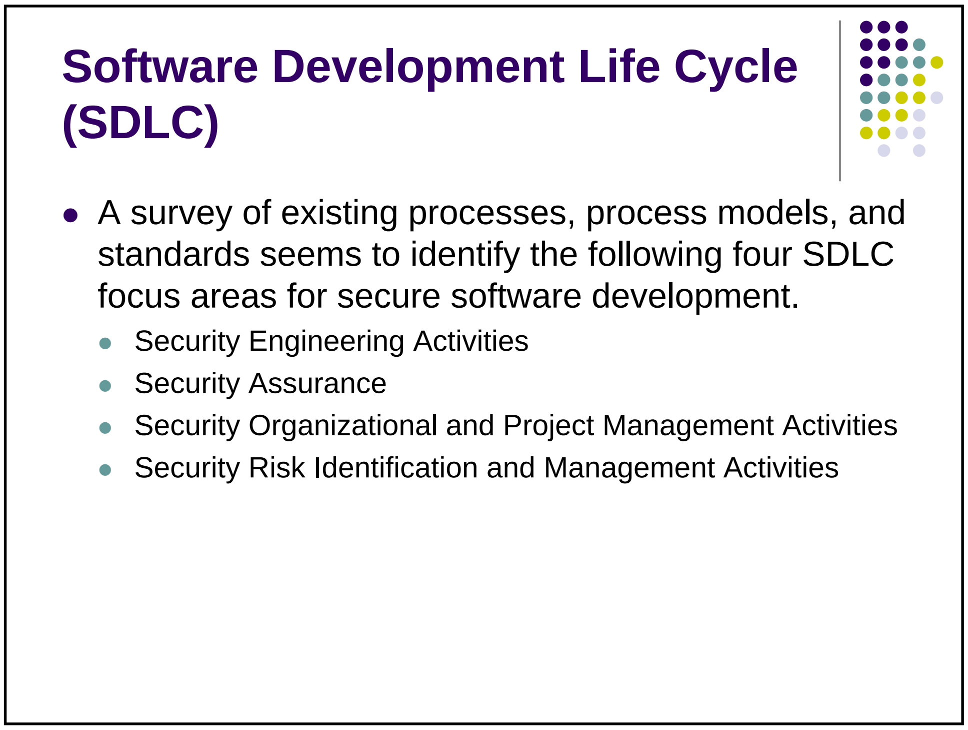
- **Process model**
  - provides a reference set of best practices that can be used for both
    - process improvement and
    - process assessment.
  - defines the characteristics of processes.
  - Usually have an architecture or a structure.
- Most process models also have a *capability* or *maturity* dimension, that can be used for
  - assessment and
  - evaluation purposes.

Process Models

- Process Models
  - have been produced to create
    - common measures of organizational processes throughout the software development lifecycle (SDLC).
  - identify many technical and management practices
  - primarily address good software engineering practices to manage and build software
  - Do not, however, guarantee software developed is bug free

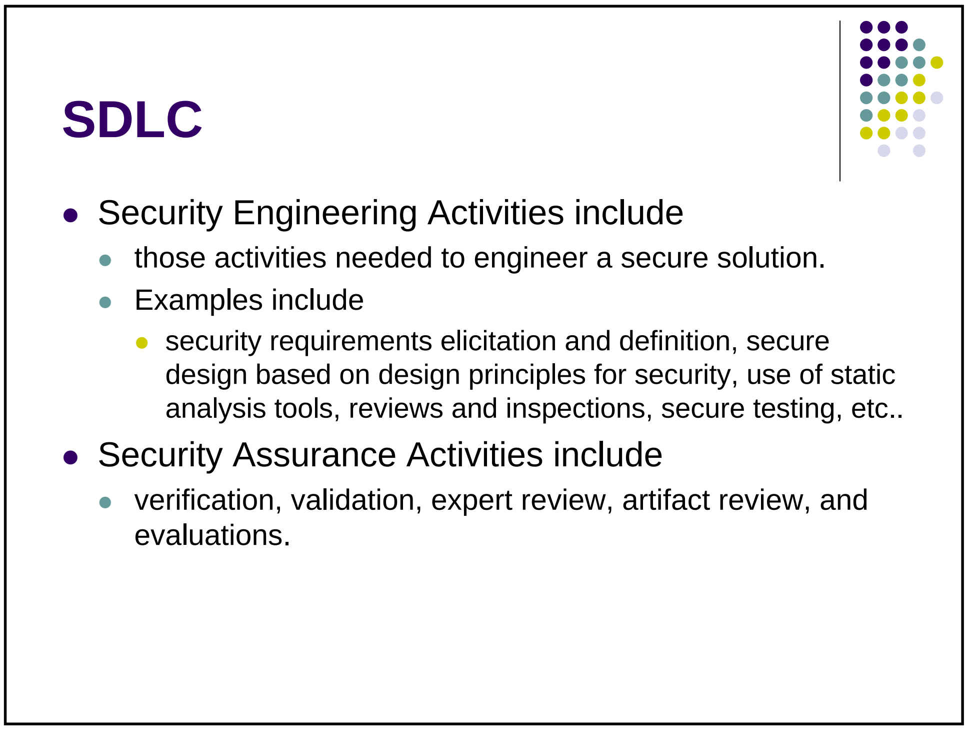
Assessments

- **Assessments, evaluations, appraisals**
  - Imply comparison of a process being practiced to a reference process model or standard.
  - used to understand process capability in order to improve processes.
  - help determine if the processes being practiced are
    - adequately specified, designed, integrated, and implemented sufficiently to support the needs



Software Development Life Cycle (SDLC)

- A survey of existing processes, process models, and standards seems to identify the following four SDLC focus areas for secure software development.
  - Security Engineering Activities
  - Security Assurance
  - Security Organizational and Project Management Activities
  - Security Risk Identification and Management Activities



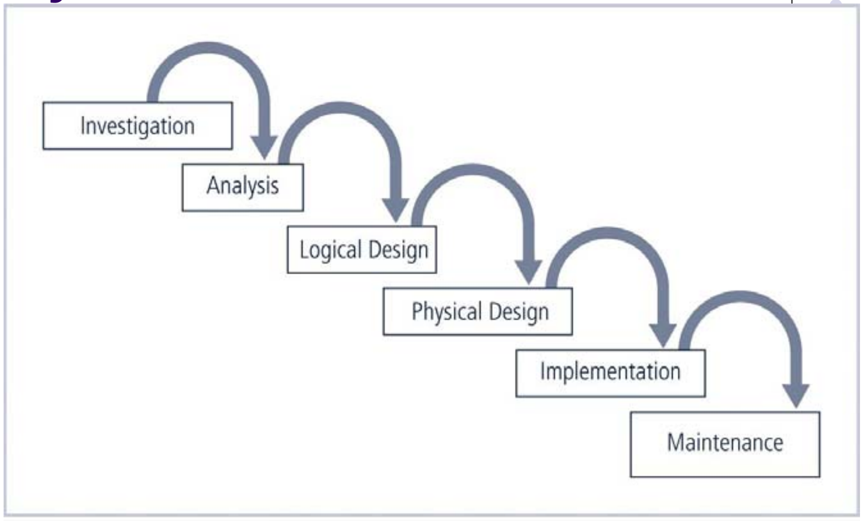
SDLC

- Security Engineering Activities include
  - those activities needed to engineer a secure solution.
  - Examples include
    - security requirements elicitation and definition, secure design based on design principles for security, use of static analysis tools, reviews and inspections, secure testing, etc..
- Security Assurance Activities include
  - verification, validation, expert review, artifact review, and evaluations.

SDLC

- Security Organizational and Project Management Activities include
  - Organizational management
    - organizational policies, senior management sponsorship and oversight, establishing organizational roles, and other organizational activities that support security.
  - Project management
    - project planning and tracking, resource allocation and usage to ensure that the security engineering, security assurance, and risk identification activities are planned, managed, and tracked.
- Security Risk Identification and Management Activities
  - identifying and managing security risks is one of the most important activities in a secure SDLC

System DLC



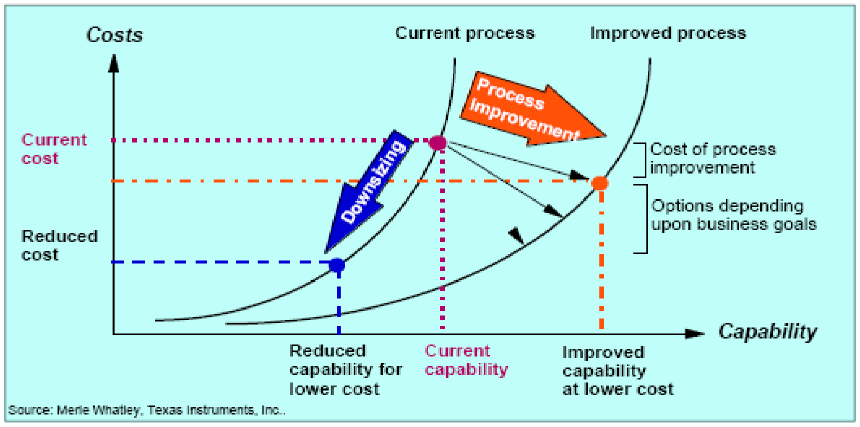
Capability Maturity Models (CMM)

- CMM
  - Provides reference model of mature practices
  - Helps identify the potential areas of improvement
  - Provides goal-level definition for and key attributes for specific processes
  - No operational guidance
  - *Defines process characteristics*

CMM

- Three CMMs
  - Capability Maturity Model Integration® (CMMI®),
  - The integrated Capability Maturity Model (iCMM), and the
  - Systems Security Engineering Capability Maturity Model (SSE-CMM)
    - Specifically to develop security

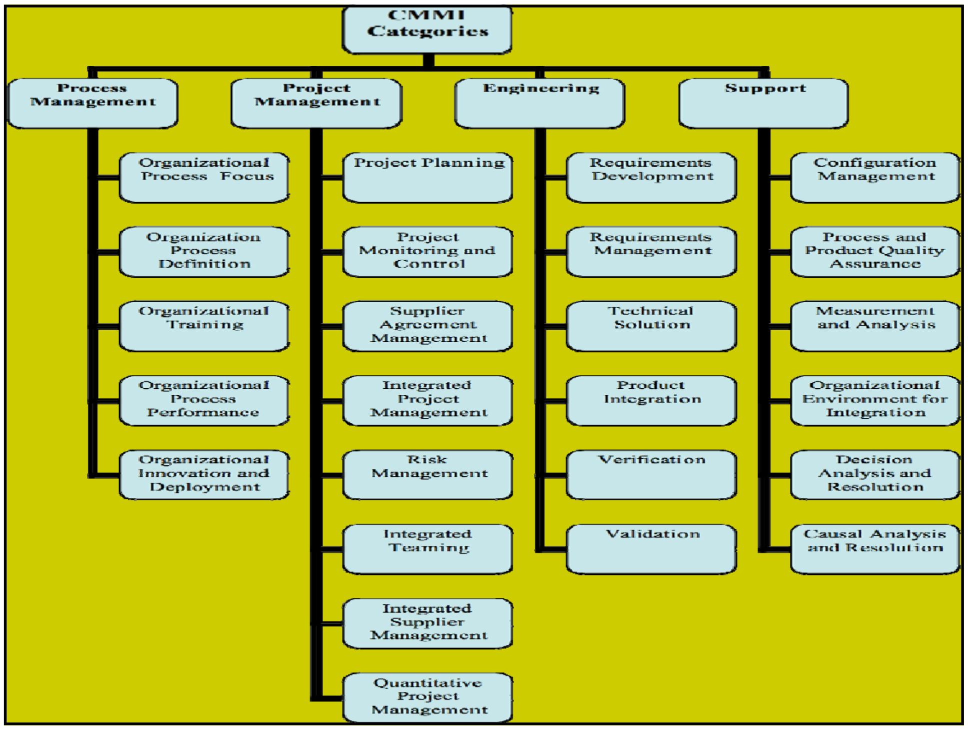
Why CMM?


Costs
Current process
Improved process
Process Improvement
Downsizing
Current cost
Reduced cost
Cost of process improvement
Options depending upon business goals
Capability
Reduced capability for lower cost
Current capability
Improved capability at lower cost
Source: Merle Whatley, Texas Instruments, Inc..

Source: http://www.secat.com/download/locked_pdf/SSEovrw_lkd.pdf

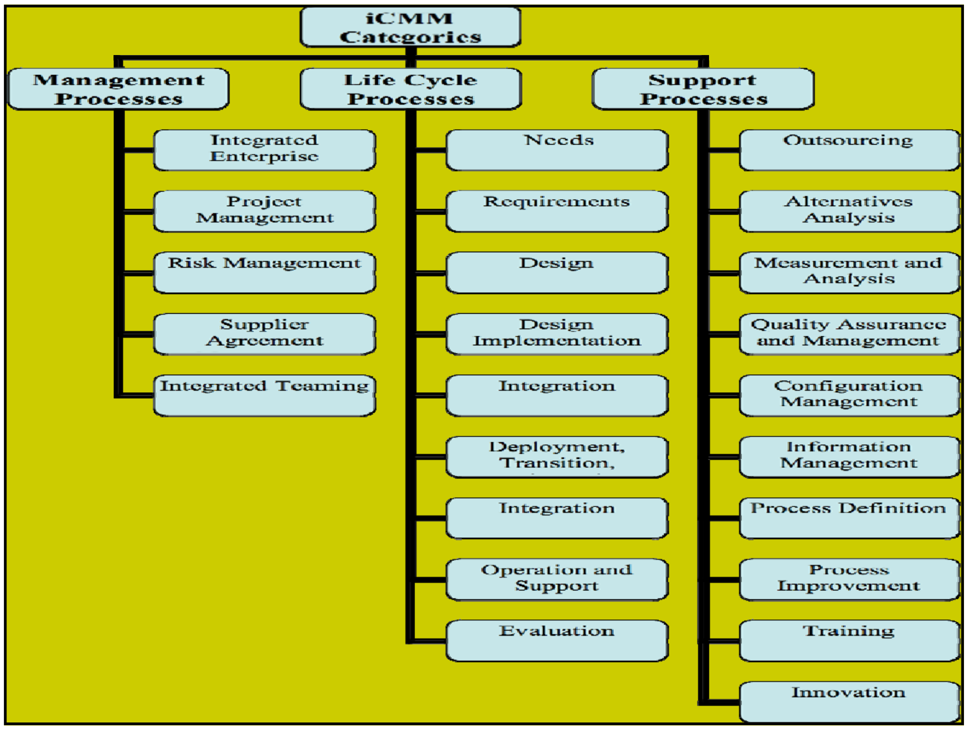
CMMI

- CMM Integration (CMMI) provides
  - the latest best practices for product and service development, maintenance, and acquisition, including mechanisms to help organizations improve their processes and provides criteria for evaluating process capability and process maturity.
- As of Dec 2005, the SEI reports
  - 1106 organizations and 4771 projects have reported results from CMMI-based appraisals
- its predecessor, the software CMM (SW-CMM)
  - Since 80s – Dec, 2005
    - 3049 Organizations + 16,540 projects


CMMI Categories
Process Management
Project Management
Engineering
Support
Organizational Process Focus
Organization Process Definition
Organizational Training
Organizational Process Performance
Organizational Innovation and Deployment
Project Planning
Project Monitoring and Control
Supplier Agreement Management
Integrated Project Management
Risk Management
Integrated Teaming
Integrated Supplier Management
Quantitative Project Management
Requirements Development
Requirements Management
Technical Solution
Product Integration
Verification
Validation
Configuration Management
Process and Product Quality Assurance
Measurement and Analysis
Organizational Environment for Integration
Decision Analysis and Resolution
Causal Analysis and Resolution

Integrated CMM

- iCMM is widely used in the Federal Aviation Administration (FAA-iCMM)
  - Provides a single model for enterprise-wide improvement
  - integrates the following standards and models:
    - ISO 9001:2000, EIA/IS 731,
    - Malcolm Baldrige National Quality Award and President's Quality Award criteria,
    - CMMI-SE/SW/IPPD and
    - CMMI-A, ISO/IEC TR 15504, ISO/IEC 12207, and ISO/IEC CD 15288.



Trusted CMM

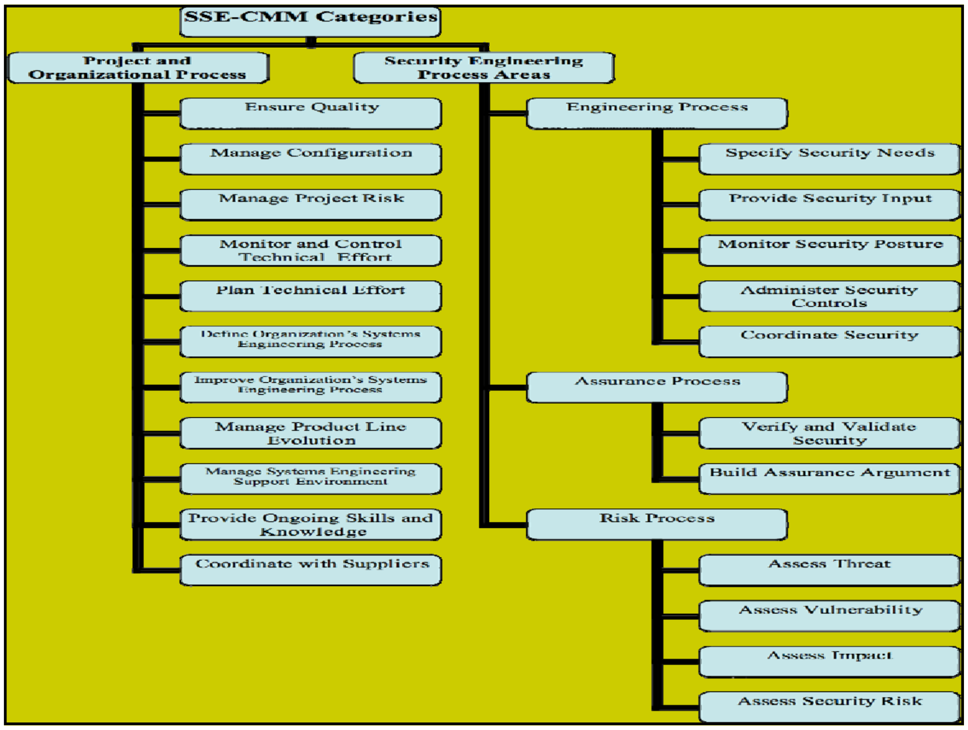
- Trusted CMM
  - In early 1990 as Trusted Software Methodology (TSM)
  - TSM defines trust levels
    - *Low* emphasizes resistance to unintentional vulnerabilities
    - *High* adding processes to counter malicious developers
  - TSM was later harmonized with CMM
    - Not much in use

Systems Security Engineering CMM

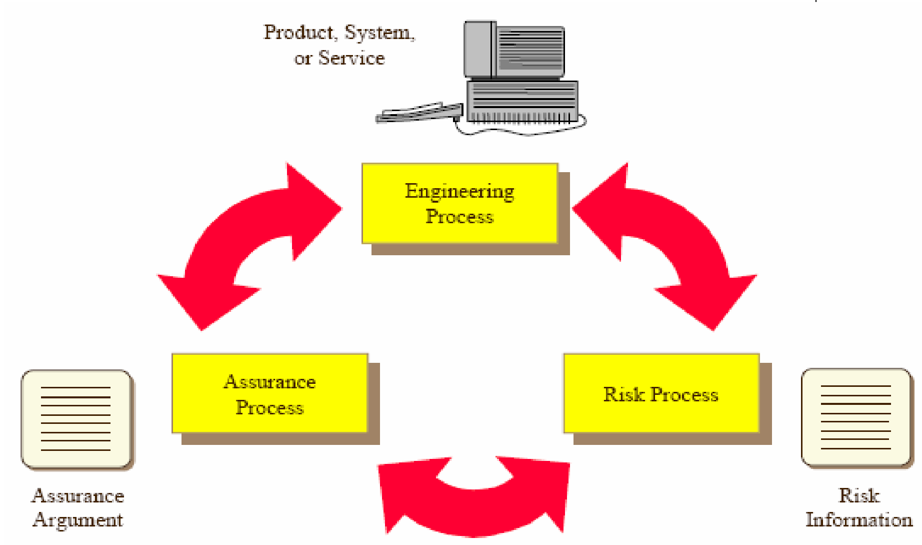
- The SSE-CMM
  - is a process model that can be used to improve and assess
    - the security engineering capability of an organization.
  - provides a comprehensive framework for
    - evaluating security engineering practices against the generally accepted security engineering principles.
  - provides a way to measure and improve performance in the application of security engineering principles.

SSE-CMM

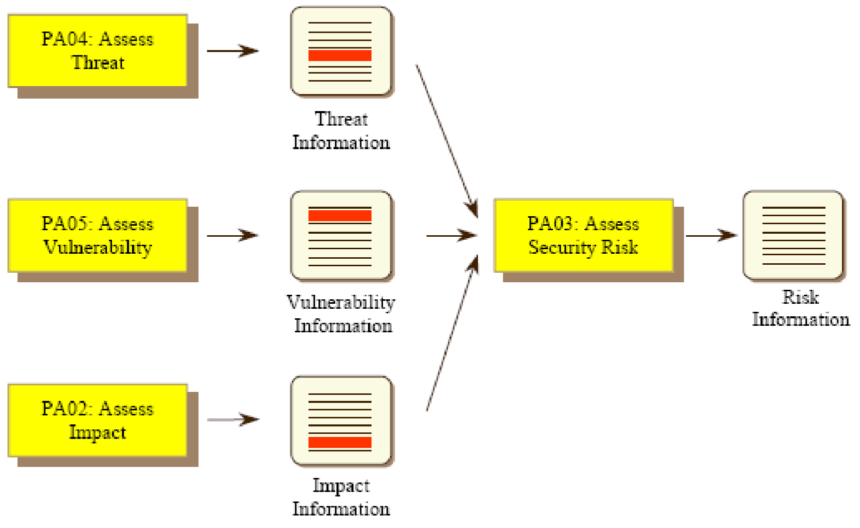
- Purpose for SSE-CMM
  - although the field of security engineering has several generally accepted principles, it lacks a comprehensive framework for evaluating security engineering practices against the principles.
- The SSE-CMM also
  - describes the essential characteristics of an organization's security engineering processes.
- The SSE-CMM is now ISO/IEC 21827 standard (version 3 is available)


SSE-CMM Categories
Project and Organizational Process
Security Engineering Process Areas
Ensure Quality
Manage Configuration
Manage Project Risk
Monitor and Control Technical Effort
Plan Technical Effort
Define Organization's Systems Engineering Process
Improve Organization's Systems Engineering Process
Manage Product Line Evolution
Manage Systems Engineering Support Environment
Provide Ongoing Skills and Knowledge
Coordinate with Suppliers
Engineering Process
Specify Security Needs
Provide Security Input
Monitor Security Posture
Administer Security Controls
Coordinate Security
Assurance Process
Verify and Validate Security
Build Assurance Argument
Risk Process
Assess Threat
Assess Vulnerability
Assess Impact
Assess Security Risk

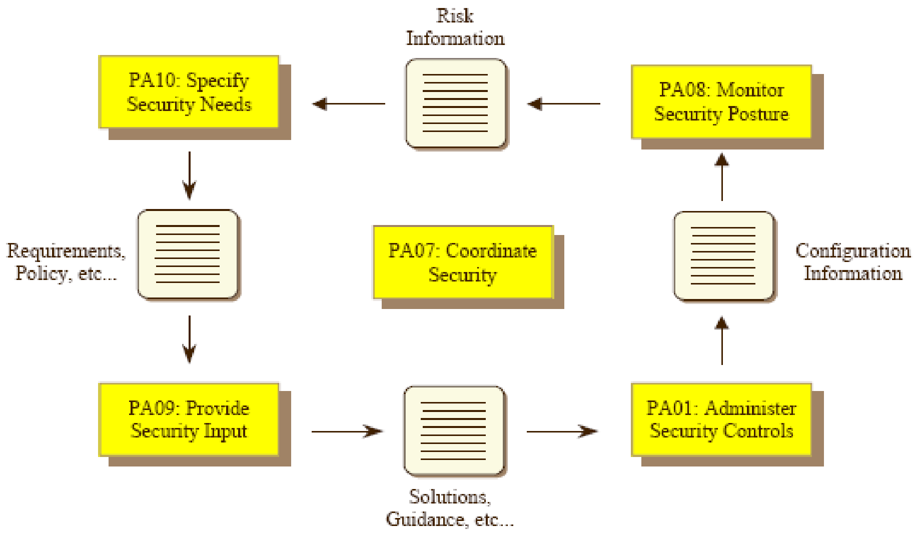
Security Engineering Process



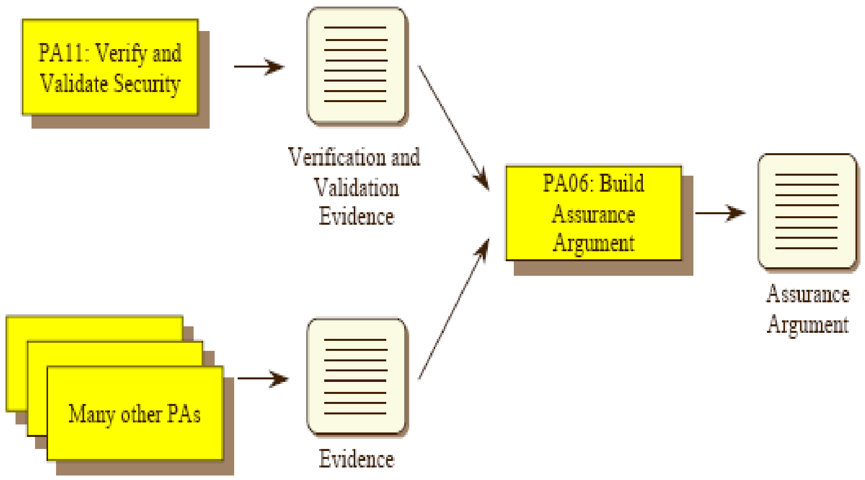
Security Risk Process



Security is part of Engineering

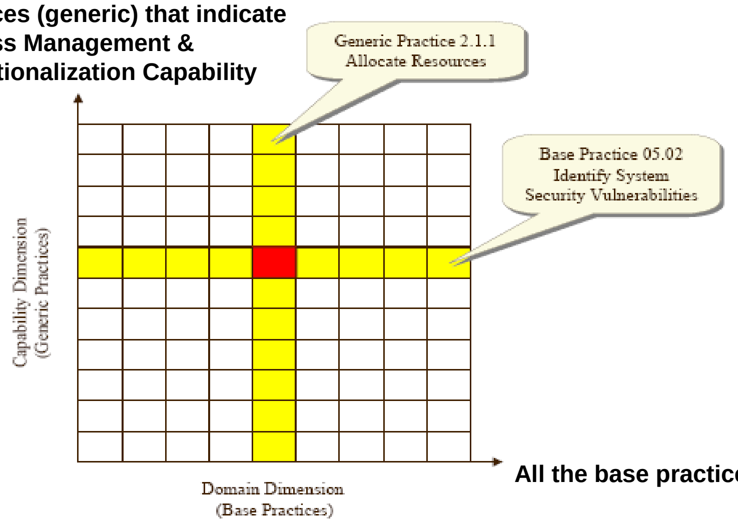

Risk Information
PA10: Specify Security Needs
PA08: Monitor Security Posture
Requirements, Policy, etc...
PA07: Coordinate Security
Configuration Information
PA09: Provide Security Input
PA01: Administer Security Controls
Solutions, Guidance, etc...

Assurance


PA11: Verify and Validate Security
Verification and Validation Evidence
Many other PAs
Evidence
PA06: Build Assurance Argument
Assurance Argument

SSE-CMM Dimensions

Practices (generic) that indicate Process Management & Institutionalization Capability


Generic Practice 2.1.1 Allocate Resources
Base Practice 05.02 Identify System Security Vulnerabilities
Capability Dimension (Generic Practices)
Domain Dimension (Base Practices)

All the base practices

SSE-CMM

- 129 base practices Organized into 22 process areas
  - 61 of these, organized in 11 process areas, cover all major areas of security engineering
    - Remaining relates to project and organization domains
- Base practice
  - Applies across the life cycle of the enterprise
  - Does not overlap with other base practices
  - Represents a “best practice” of the security community
  - Does not simply reflect a state of the art technique
  - Is applicable using multiple methods in multiple business context
  - Does not specify a particular method or tool

Process Area

- Assembles related activities in one area for ease of use
- Relates to valuable security engineering services
- Applies across the life cycle of the enterprise
- Can be implemented in multiple organization and product contexts
- Can be improved as a distinct process
- Can be improved by a group with similar interests in the process
- Includes all base practices that are required to meet the goals of the process area

Process Areas

Process Areas related to Security Engineering process areas

- PA01 Administer Security Controls
- PA02 Assess Impact
- PA03 Assess Security Risk
- PA04 Assess Threat
- PA05 Assess Vulnerability
- PA06 Build Assurance Argument
- PA07 Coordinate Security
- PA08 Monitor Security Posture
- PA09 Provide Security Input
- PA10 Specify Security Needs
- PA11 Verify and Validate Security

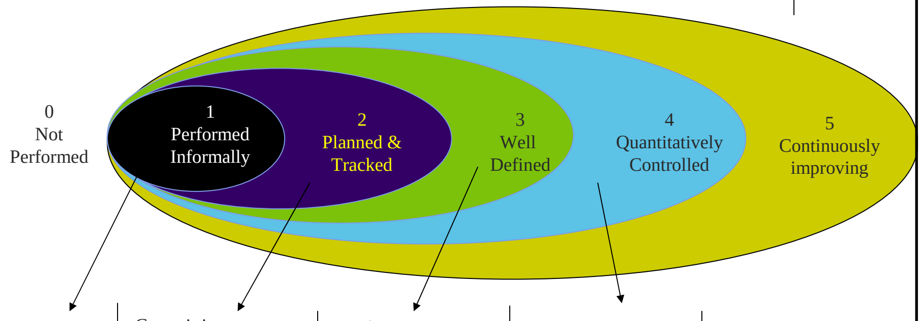
Process Areas related to project and Organizational practices

- PA12 – Ensure Quality
- PA13 – Manage Configuration
- PA14 – Manage Project Risk
- PA15 – Monitor and Control Technical Effort
- PA16 – Plan Technical Effort
- PA17 – Define Organization's Systems Engineering Process
- PA18 – Improve Organization's Systems Engineering Process
- PA19 – Manage Product Line Evolution
- PA20 – Manage Systems Engineering Support Environment
- PA21 – Provide Ongoing Skills and Knowledge
- PA22 – Coordinate with Suppliers

Generic Process Areas

- Activities that apply to all processes
- They are used during
  - Measurement and institutionalization
- Capability levels
  - Organize common features
  - Ordered according to maturity

Capability Levels



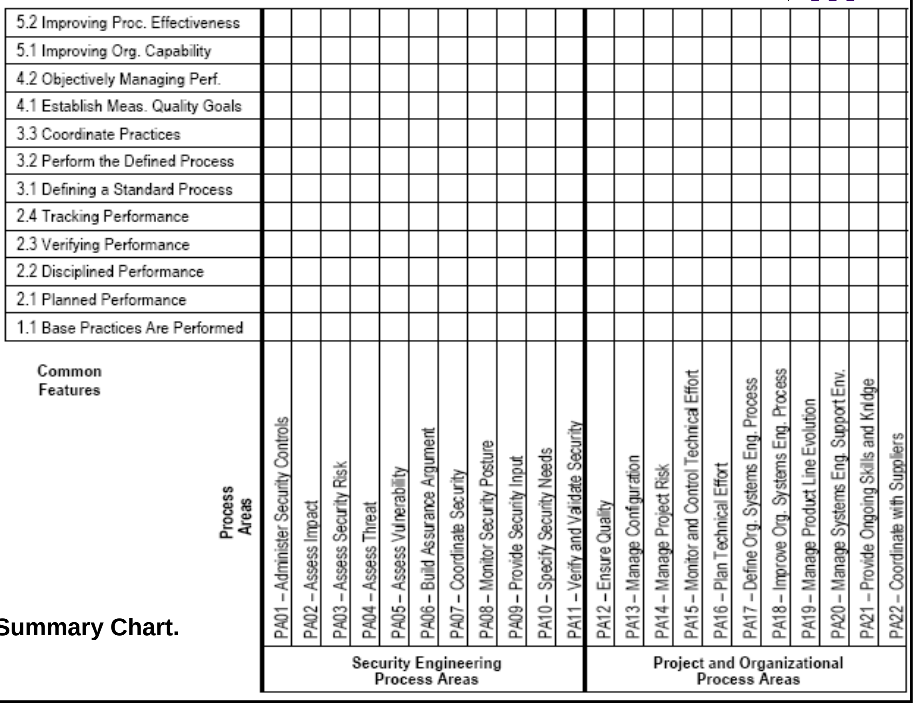
Base Practices Performed

Committing to perform
Planning performance
Disciplined performance
Tracking performance
Verifying performance

Defining a standard process
Tailoring standard process
Using data
Perform a defined process

Establishing measurable quality goals
Determining process capability to achieve goals
Objectively managing performance

Establishing quantitative process goals
Improving process effectiveness



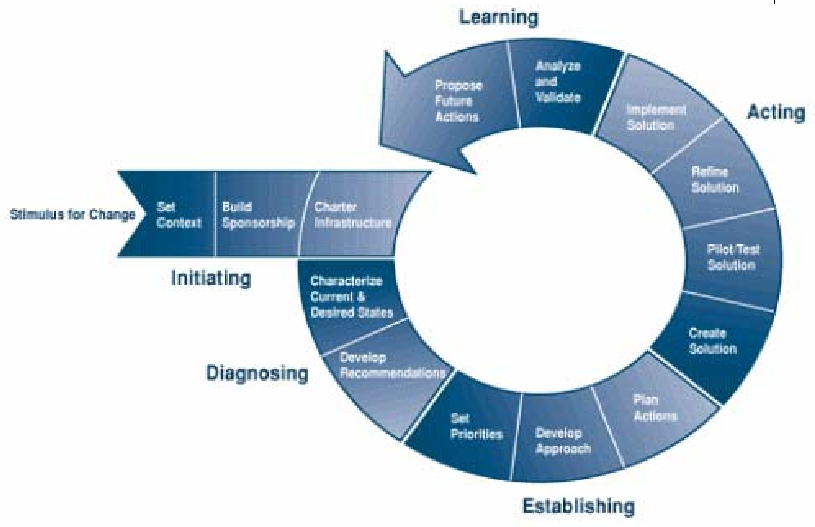
Common Features / Process Areas	Security Engineering Process Areas											Project and Organizational Process Areas										
	PA01 – Administer Security Controls	PA02 – Assess Impact	PA03 – Assess Security Risk	PA04 – Assess Threat	PA05 – Assess Vulnerability	PA06 – Build Assurance Argument	PA07 – Coordinate Security	PA08 – Monitor Security Posture	PA09 – Provide Security Input	PA10 – Specify Security Needs	PA11 – Verify and Validate Security	PA12 – Ensure Quality	PA13 – Manage Configuration	PA14 – Manage Project Risk	PA15 – Monitor and Control Technical Effort	PA16 – Plan Technical Effort	PA17 – Define Org. Systems Eng. Process	PA18 – Improve Org. Systems Eng. Process	PA19 – Manage Product Line Evolution	PA20 – Manage Systems Eng. Support Env.	PA21 – Provide Ongoing Skills and Knldge	PA22 – Coordinate with Suppliers
5.2 Improving Proc. Effectiveness																						
5.1 Improving Org. Capability																						
4.2 Objectively Managing Perf.																						
4.1 Establish Meas. Quality Goals																						
3.3 Coordinate Practices																						
3.2 Perform the Defined Process																						
3.1 Defining a Standard Process																						
2.4 Tracking Performance																						
2.3 Verifying Performance																						
2.2 Disciplined Performance																						
2.1 Planned Performance																						
1.1 Base Practices Are Performed																						

Summary Chart.

Using SSE-CMM

- Can be used in one of the three ways
  - Process improvement
    - Facilitates understanding of the level of security engineering process capability
  - Capability evaluation
    - Allows a consumer organization to understand the security engineering process capability of a provider
  - Assurance
    - Increases the confidence that product/system/service is trustworthy

Process Improvement



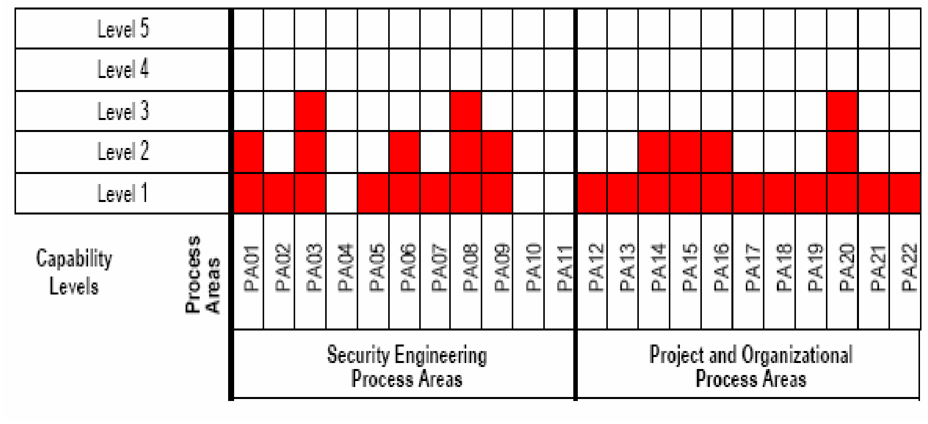
Capability Evaluation

- No need to use any particular appraisal method
- SSE-CMM Appraisal (SSAM) method has been developed if needed
- SSAM purpose
  - Obtain the baseline or benchmark of actual practice related to security engineering within the organization or project
  - Create or support momentum for improvement within multiple levels of the organizational structure

SSAM Overview

- Planning phase
  - Establish appraisal framework
- Preparation phase
  - Prepare team for onsite phase through information gathering (questionnaire)
  - Preliminary data analysis indicate what to look for / ask for
- Onsite phase
  - Data gathering and validation with the practitioner
  - interviews
- Post-appraisal
  - Present final data analysis to the sponsor

Capability Evaluation



Assurance

- A mature organization is significantly more likely to create a product or system with appropriate assurance
- Process evidence can be used to support claims for the trustworthiness of those products
- It is conceivable that
  - An immature organization could produce high assurance product.

CMI/iCMM/SSE-CMM

- Because of the integration of process disciplines and coverage of enterprise issues,
  - the CMMI and the iCMM are used by more organizations than the SSE-CMM;
- CMMI and iCMM have gaps in their coverage of safety and security.
- FAA and the DoD have sponsored a joint effort to identify best safety and security practices for use in combination with the iCMM and the CMMI.

Safety/Security additions

- The proposed Safety and Security additions include the following four goals:
  - Goal 1 – An infrastructure for safety and security is established and maintained.
  - Goal 2 – Safety and security risks are identified and managed.
  - Goal 3 – Safety and security requirements are satisfied.
  - Goal 4 – Activities and products are managed to achieve safety and security requirements and objectives.

Goal 1 related practices

1. Ensure safety and security awareness, guidance, and competency.
2. Establish and maintain a qualified work environment that meets safety and security needs.
3. Ensure integrity of information by providing for its storage and protection, and controlling access and distribution of information.
4. Monitor, report and analyze safety and security incidents and identify potential corrective actions.
5. Plan and provide for continuity of activities with contingencies for threats and hazards to operations and the infrastructure

Goal 2 related practices

1. Identify risks and sources of risks attributable to vulnerabilities, security threats, and safety hazards.
2. For each risk associated with safety or security, determine the causal factors, estimate the consequence and likelihood of an occurrence, and determine relative priority.
3. For each risk associated with safety or security, determine, implement and monitor the risk mitigation plan to achieve an acceptable level of risk.

Goal 3 related practices

1. Identify and document applicable regulatory requirements, laws, standards, policies, and acceptable levels of safety and security.
2. Establish and maintain safety and security requirements, including integrity levels, and design the product or service to meet them.
3. Objectively verify and validate work products and delivered products and services to assure safety and security requirements have been achieved and fulfill intended use.
4. Establish and maintain safety and security assurance arguments and supporting evidence throughout the lifecycle.

Goal 4 related practices

1. Establish and maintain independent reporting of safety and security status and issues.
2. Establish and maintain a plan to achieve safety and security requirements and objectives.
3. Select and manage products and suppliers using safety and security criteria.
4. Measure, monitor and review safety and security activities against plans, control products, take corrective action, and improve processes.

Team Software Process for Secure SW/Dev

- TSP
  - provides a framework, a set of processes, and disciplined methods for applying software engineering principles at the team and individual level
- TSP for Secure Software Development (TSP-Secure)
  - focus more directly on the security of software applications.

Team Software Process for Secure SW/Dev

- TSP-Secure addresses secure software development (three ways).
  1. Secure software is not built by accident,
     - TSP-Secure addresses planning for security.
     - Since schedule pressures and people issues get in the way of implementing best practices, TSP-Secure helps to build self-directed development teams, and then put these teams in charge of their own work.

TSP-Secure

1. Since security and quality are closely related,
   - TSP-Secure helps manage quality throughout the product development life cycle.
2. Since people building secure software must have an awareness of software security issues,
   - TSP-Secure includes security awareness training for developers.

TSP-Secure

- Teams
  - Develop their own plans
  - Make their own commitments
  - Track and manage their own work
  - Take corrective action when needed

TSP-Secure

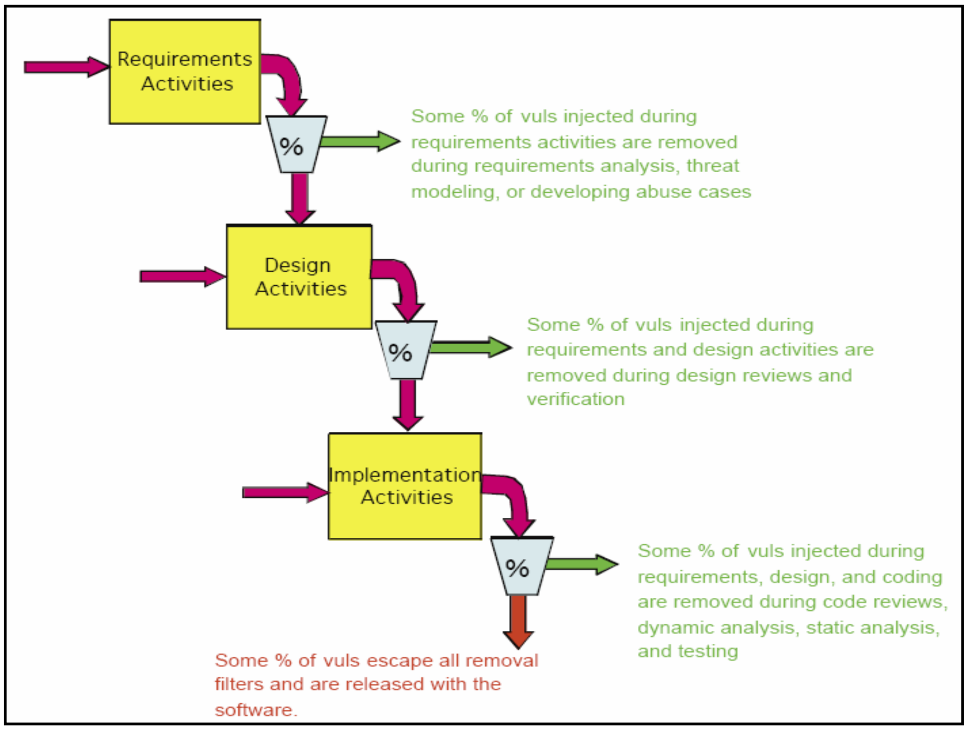
- Initial planning – “project launch” (3-4 days)
  - Tasks include
    - identifying security risks,
    - eliciting and defining security requirement, secure design, and code reviews,
    - use of static analysis tools, unit tests, and Fuzz testing.
- Next, the team executes its plan, and ensures all security related activities are taking place.
  - Security status is presented and discussed during every management status briefing.

TSP-Secure

- Basis
  - Defective software is seldom secure
  - Defective software is not inevitable
    - Consider cost of reducing defects
    - Manage defects throughout the lifecycle
  - Defects are leading cause of vulnerabilities
    - Use multiple defect removal points in the SD
      - Defect filters

TSP-Secure

- Key questions in managing defects
  - What type of defects lead to security vulnerabilities?
  - Where in the software development life cycle should defects be measured?
  - What work products should be examined for defects?
  - What tools and methods should be used to measure the defects?
  - How many defects can be removed at each step?
  - How many estimated defects remain after each removal step?
- TSP-Secure includes training for developers, managers, and other team members.


Requirements Activities
%
Some % of vuls injected during requirements activities are removed during requirements analysis, threat modeling, or developing abuse cases
Design Activities
%
Some % of vuls injected during requirements and design activities are removed during design reviews and verification
Implementation Activities
%
Some % of vuls injected during requirements, design, and coding are removed during code reviews, dynamic analysis, static analysis, and testing
Some % of vuls escape all removal filters and are released with the software.

Correctness by Construction

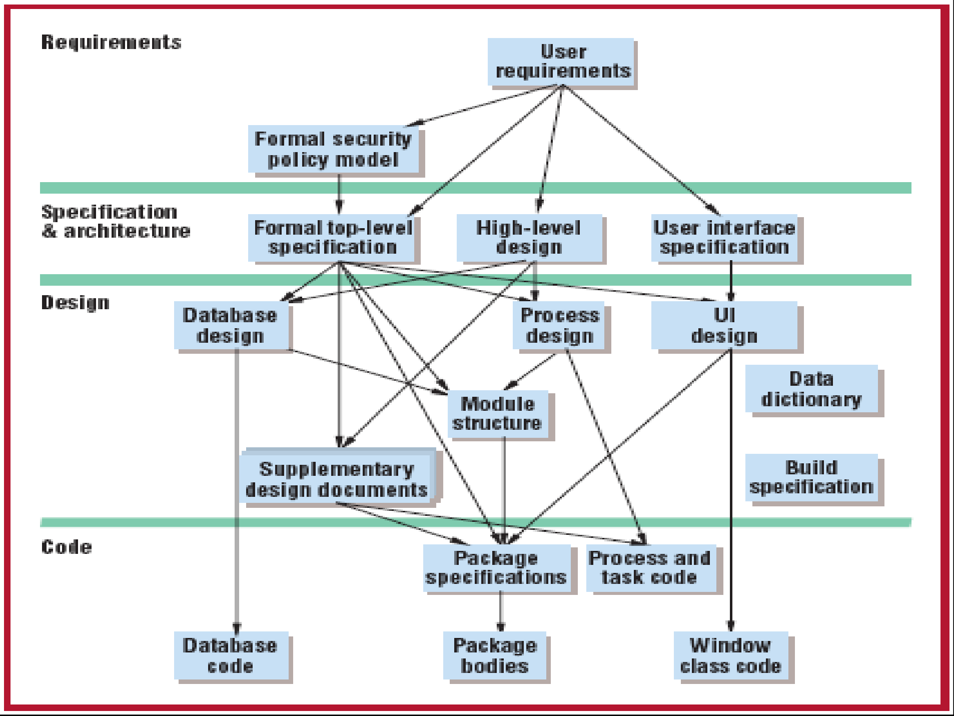
- CbC Methodology from Praxis Critical Systems
  - Process for developing high integrity software
  - Has been successfully used to develop safety-critical systems
  - Removes defects at the earliest stages
  - the process almost always uses formal methods to specify behavioral, security and safety properties of the software.

Correctness by Construction

- The seven key principles of Correctness-by-Construction are:
  - Expect requirements to change.
  - Know why you're testing (debug + verification)
  - Eliminate errors before testing
  - Write software that is easy to verify
  - Develop incrementally
  - Some aspects of software development are just plain hard.
  - Software is not useful by itself.

Correctness by Construction

- Correctness-by-Construction is
  - one of the few secure SDLC processes that incorporate formal methods into many development activities.
  - Requirements are specified using Z, and verified.
  - Code is checked by verification software, and is written in Spark, a subset of Ada which can be statically assured.



Correctness by Construction

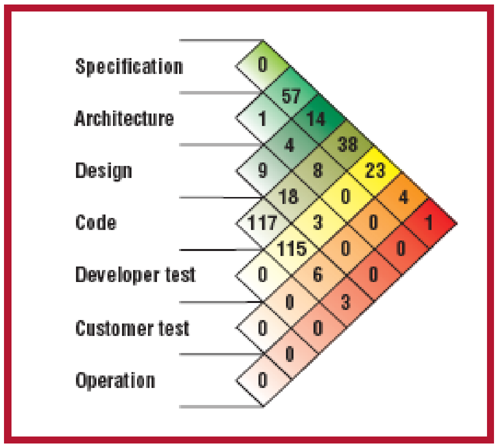


Table 1

Distribution of effort.

Activity	Effort (%)
Requirements	2
Specification and architecture	25
Code	14
Test	34
Fault fixing	6
Project management	10
Training	3
Design authority	3
Development- and target-environment	3

Agile Methods

- Agile manifesto
  - "We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:
    - *Individuals and interactions* over processes and tools
    - *Working software* over comprehensive documentation
    - *Customer collaboration* over contract negotiation
    - *Responding to change* over following a plan
  - That is, while there is value in the items on the right, we value the items on the left more."

Agile manifesto principles

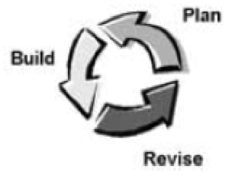
- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Business people and developers work together daily throughout the project.
- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
- Working software is the primary measure of progress.
- Agile processes promote sustainable development. The sponsors, developers and users should be able to maintain a constant pace indefinitely.
- Continuous attention to technical excellence and good design enhances agility.
- Simplicity—the art of maximizing the amount of work not done—is essential.
- The best architectures, requirements and designs emerge from self-organizing teams.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Agile Processes

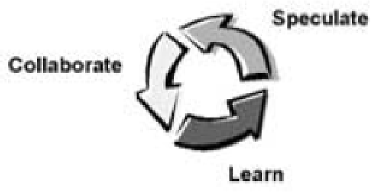
- Among many variations
  - Adaptive software development (ASP)
  - Extreme programming (XP)
  - Crystal
  - Rational Unified Process (RUP)

Adaptive software development (ASP)

- Premise:
  - Unpredictable outcomes
  - Not possible to plan successfully in a fast moving and unpredictable business environment
- Instead of evolutionary life cycle model use adaptive life cycle



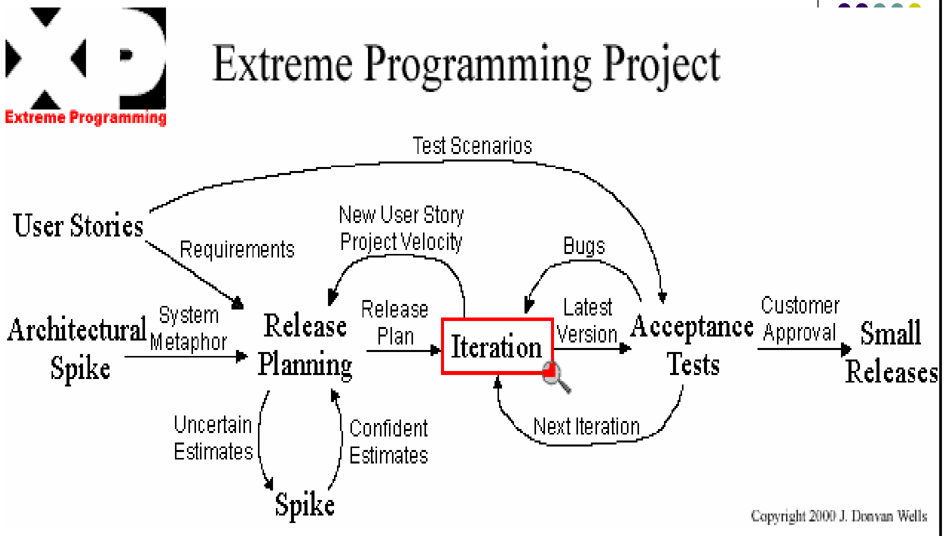
Evolutionary Life Cycle



Adaptive Life Cycle

Extreme Programming

- A high profile agile process
- Four basic values
  - Communication
  - Feedback
    - Check results
  - Simplicity
    - Avoid unnecessary artifacts/activities to a project
  - Courage
    - More faith on people than in processes



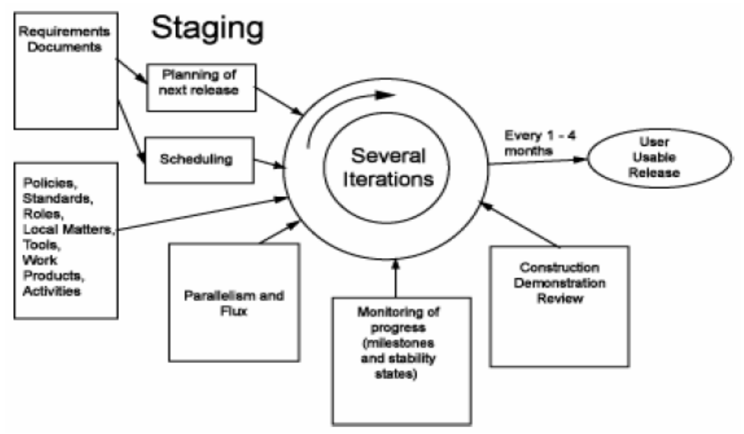
Crystal

- A family of processes each applied to different kinds of projects
- Selecting crystal process that matches
  - Comfort
    - System failure means loss of comfort
  - Discretionary money
  - Essential money
  - Life
    - Most rigorous process needed

Crystal

- Each of the process shares common policy standards
  - Incremental delivery
  - Progress tracking by milestones based on software deliveries and major decisions rather than written documents
  - Direct user involvement
  - Automated regression testing of functionality
  - Two user viewings per release
  - Workshops for product and methodology – tuning at the beginning and in the middle of each increment

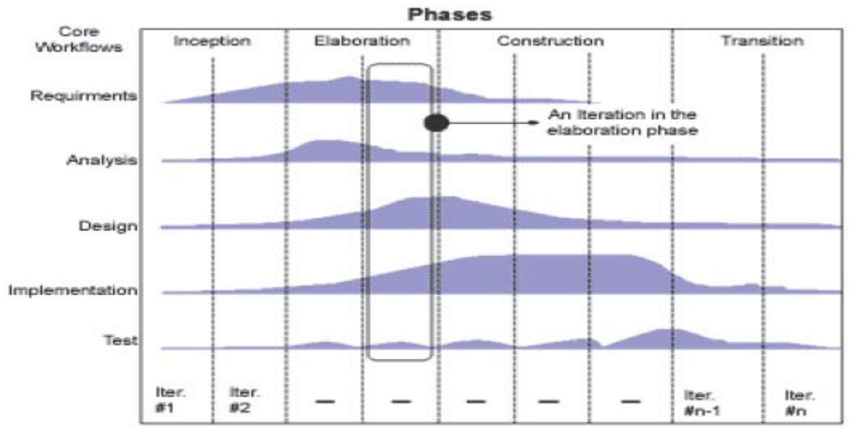
Crystal


Requirements Documents
Staging
Planning of next release
Scheduling
Policies, Standards, Roles, Local Matters, Tools, Work Products, Activities
Several Iterations
Every 1 - 4 months
User Usable Release
Parallelism and Flux
Monitoring of progress (milestones and stability states)
Construction Demonstration Review

Rational Unified Process

- A generic process framework that uses a specific methodology to accomplish the tasks associated with it
  - Uses UML language to develop use cases for the software system design
- In its simplest form
  - Mimics the waterfall model

Rational Unified Process



The five workflows take place over four phases [11]

TSP Revisited
- How TSP Relates to Agile ..

- *Individuals and interactions* over processes and tools

- TSP holds that the individual is key to product quality and effective member interactions are necessary to the team's success.
  - Project launches strive to create gelled teams.
  - Weekly meetings and communication are essential to sustain them.
  - Teams define their own processes in the launch.

How TSP Relates

- *Working software* over comprehensive documentation

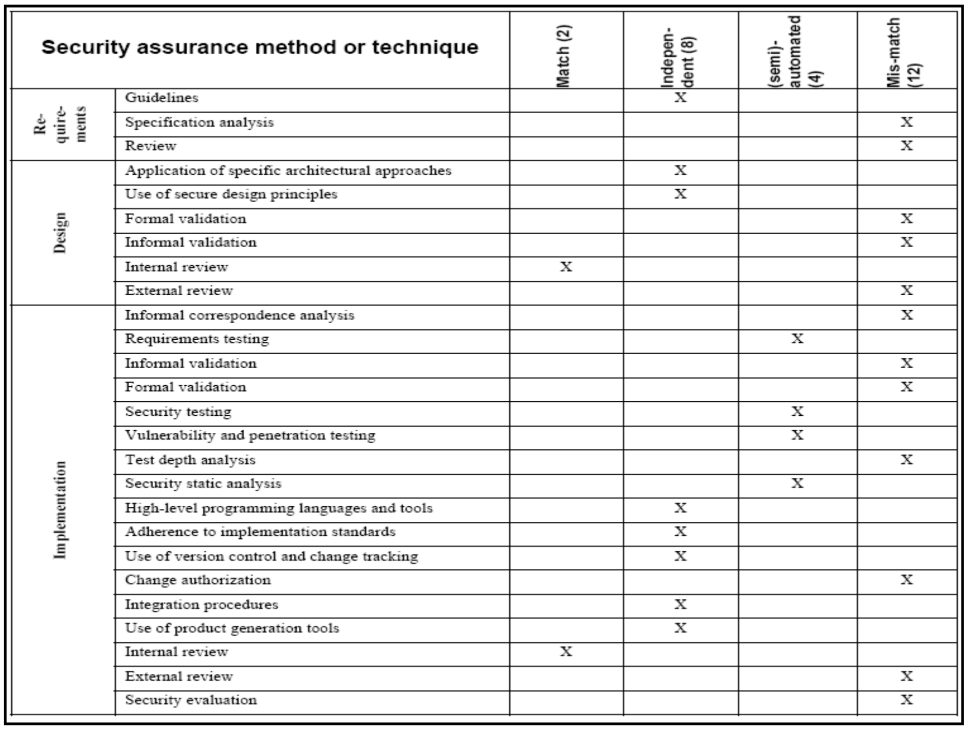
- TSP teams can choose evolutionary or iterative lifecycle models to deliver early functionality—the focus is on high quality from the start. TSP does not require heavy documentation.
  - Documentation should merely be sufficient to facilitate effective reviews and information sharing.

How TSP Relates

- *Customer collaboration* over contract negotiation
- Learning what the customer wants is a key focus of the “launch”. Sustaining customer contact is one reason for having a customer interface manager on the team.
  - Focus on negotiation of a contract is more a factor of the organization than of whether TSP is used.

How TSP Relates

- *Responding to change* over following a plan
- TSP teams expect and plan for change by:
  - Adjusting the team's process through process improvement proposals and weekly meetings.
  - Periodically re-launching and re-planning whenever the plan is no longer a useful guide.
  - Adding new tasks as they are discovered; removing tasks that are no longer needed.
  - Dynamically rebalancing the team workload as required to finish faster.
  - Actively identifying and managing risks.



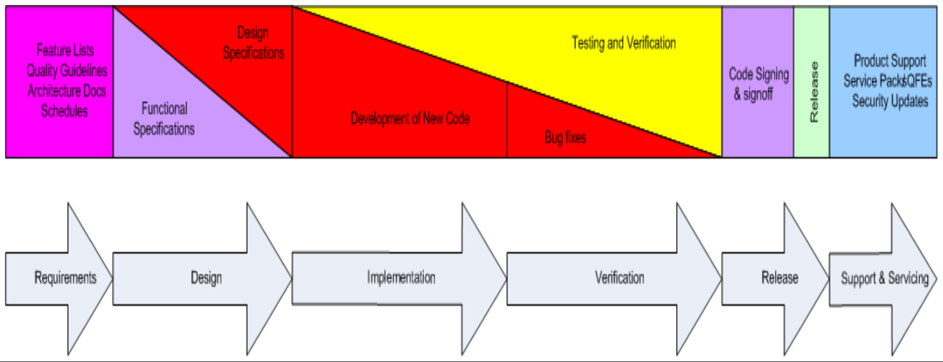
	Security assurance method or technique	Match (2)	Indepen-dent (8)	(semi)-automated (4)	Mis-match (12)
Re-quire-ments	Guidelines		X		
	Specification analysis				X
	Review				X
Design	Application of specific architectural approaches		X		
	Use of secure design principles		X		
	Formal validation				X
	Informal validation				X
	Internal review	X			
	External review				X
Implementation	Informal correspondence analysis				X
	Requirements testing			X	
	Informal validation				X
	Formal validation				X
	Security testing			X	
	Vulnerability and penetration testing			X	
	Test depth analysis				X
	Security static analysis			X	
	High-level programming languages and tools		X		
	Adherence to implementation standards		X		
	Use of version control and change tracking		X		
	Change authorization				X
	Integration procedures		X		
	Use of product generation tools		X		
	Internal review	X			
	External review				X
	Security evaluation				X

Besnosov Comparison

- 50% of traditional security assurance activities are not compatible with Agile methods (12 out of 26),
- less than 10% are natural fits (2 out of 26),
- about 30% are independent of development method, and
- slightly more than 10% (4 out of 26) could be semi-automated and thus integrated more easily into the Agile methods.

Microsoft Trustworthy Computing SDLC

- Generally accepted SDL process at MS
- (actually spiral not “waterfall” as it indicates)



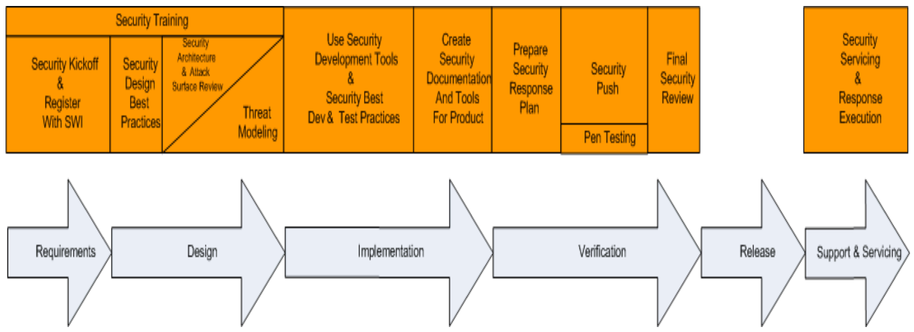
SDL Overview

- MS’s SD^3 + C paradigm
  - Secure by Design
  - Secure by Default
  - Secure by Deployment
  - Communications
    - software developers should be prepared for the discovery of product vulnerabilities and should communicate openly and responsibly

The SDL is updated as shown next

SDL at MS

- Add the SD^3 + C praradigm


Security Training
Security Kickoff & Register With SWI
Security Design Best Practices
Security Architecture & Attack Surface Review
Threat Modeling
Use Security Development Tools & Security Best Dev & Test Practices
Create Security Documentation And Tools For Product
Prepare Security Response Plan
Security Push
Pen Testing
Final Security Review
Security Servicing & Response Execution
Requirements
Design
Implementation
Verification
Release
Support & Servicing

Design Phase

- Define Security architecture and design guidelines
  - Identify tcb; use layering etc.
- Document the elements of the software attack surface
  - Find out default security
- Conduct threat modeling
- Define supplemental ship criteria

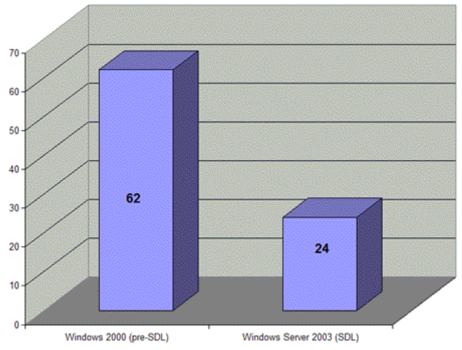
Implementation phase

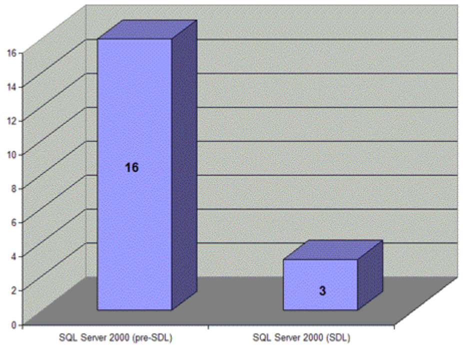
- Apply coding and testing standards
- Apply security testing tools including fuzzing tools
- Apply static analysis code scanning tools
- Conduct code reviews

Verification Phase

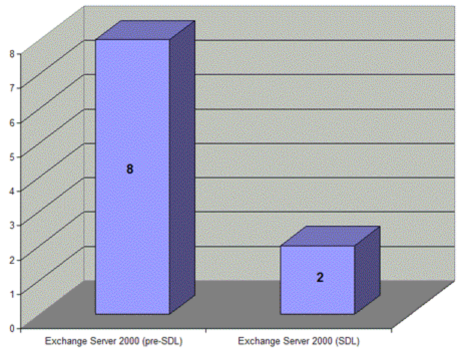
- “Security push” for Windows server 2003
  - Includes code review beyond those in implementation phase and
  - Focused testing
- Two reasons for “security push”
  - Products had reached the verification phase
  - Opportunity to review both code that was developed or updated during the implementation phase and “legacy code” that was not modified

Results





Results


Exchange Server 2000 (pre-SDL)
Exchange Server 2000 (SDL)