CISSP® Common Body of Knowledge Review:

Software Development Security Domain

Version: 5.10

Learning Objective

Software Development Security Domain

Software Development Security domain refers to the controls that are included within systems and application software and the steps used in their development (e.g., SDLC).

Software refers to system software (operating systems) and application programs such as agents, applets, software, databases, data warehouses, and knowledge-based systems. These applications may be used in distributed or centralized environments.

The candidate should fully understand the security and controls of the system development process, system life cycle, application controls, change controls, data warehousing, data mining, knowledge-based systems, program interfaces, and concepts used to ensure data and application integrity, security, and availability.

Reference: *CISSP CIB*, January 2012 (4.17.14 Rev. 13)

Current State of Insecurity in Federal Agencies

- "The 25 major agencies of Federal government continue to improve information security performance relative to C&A rate and testing of contingency plans and security controls." – OMB *FY 2008 Report to Congress on Implementation of FISMA*.

% of System with a:	FY 2005	FY 2006	FY 2007	FY 2008
Certification and Accreditation (C&A)	85%	88%	92%	**96%**
Tested Contingency Plan	61%	77%	86%	**92%**
Tested Security Controls	72%	88%	95%	**93%**
Total Systems Reported	10,289	10,595	10,304	10,679

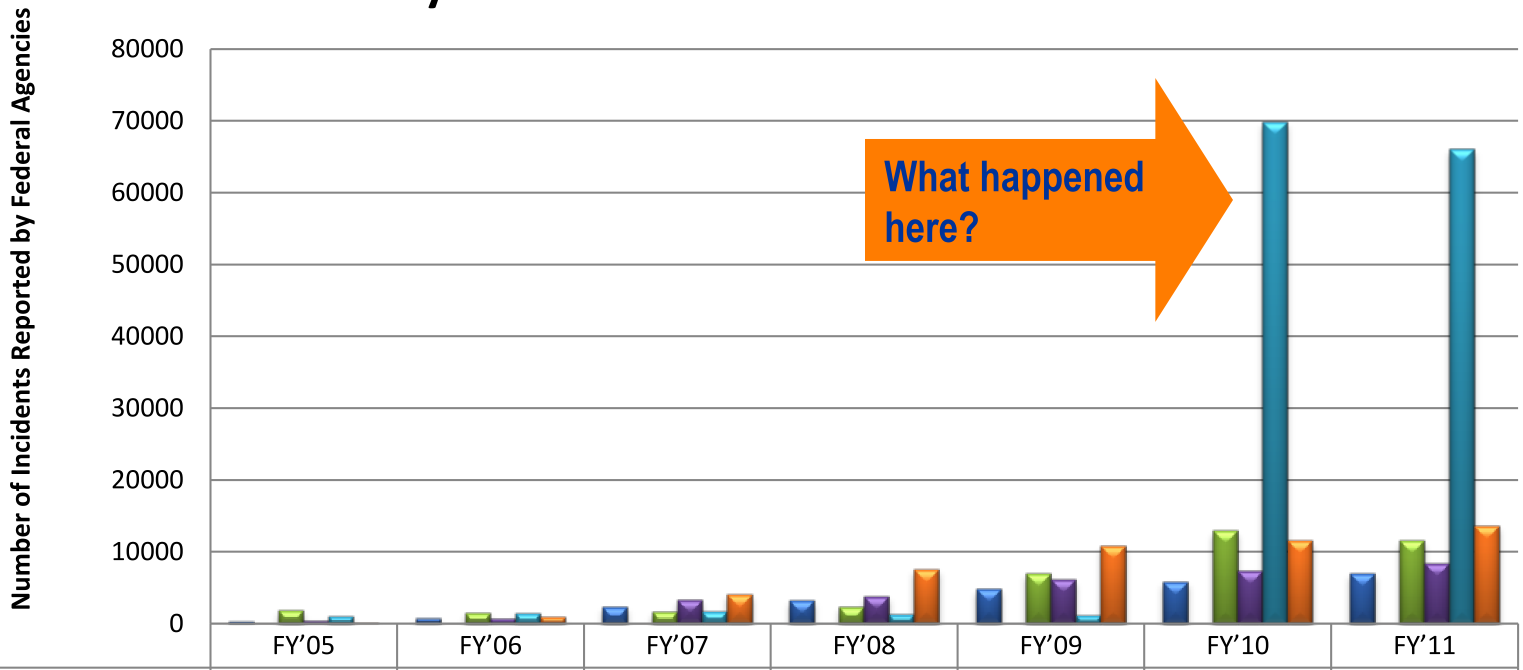
- Yet, "20 of 24 major agencies indicated that inadequate information security controls were either a significant deficiency or a material weakness."*

* **Source:** GAO-08-496, *Information Security– Although Progress Reported, Federal Agencies Need to Resolve Significant Deficiencies*, February 14, 2008

Current State of Insecurity in Federal Agencies

- # of security incidents keeps growing*…

Security Incidents - FY'05 to FY'11



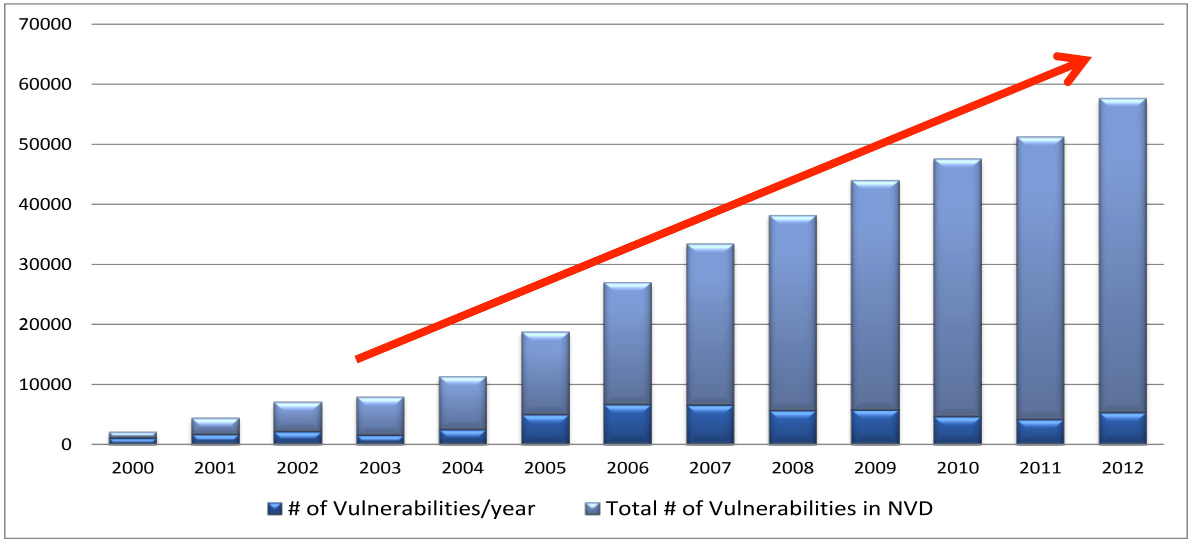
What happened here?

	FY'05	FY'06	FY'07	FY'08	FY'09	FY'10	FY'11
🟦 1. Unauthorized Access	304	706	2,321	3,214	4,848	5,782	6,959
🟥 2. Denial of Service	31	37	36	26	48	28	33
🟩 3. Malicious Code	1,806	1,465	1,607	2,274	6,977	12,926	11,556
🟪 4. Improper Usage	370	638	3,305	3,762	6,148	7,334	8,372
🟦 5. Scans/Probes/Attempted Access	976	1,388	1,661	1,272	1,152	69,832	66,057
🟧 6. Under Investigation	82	912	4,056	7,502	10,826	11,534	13,601

*** Source:** US-CERT

Current State of Insecurity in COTS Software

- The software flaw statistics are also trending upward…



- According to an analysis by Software Engineering Institute (SEI): "Most software security vulnerabilities arise from common causes; more than 90 percent are caused by known software defect types." Where the top 10 causes account for about 75 percent of all vulnerabilities.

* **Source:** National Vulnerability Database (http://nvd.nist.gov)

2011 CWE/SANS Top 25 Most Dangerous Programming Errors

Rank	Score	ID	Name
[1]	93.8	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
[2]	83.3	CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
[3]	79.0	CWE-120	Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')
[4]	77.7	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
[5]	76.9	CWE-306	Missing Authentication for Critical Function
[6]	76.8	CWE-862	Missing Authorization
[7]	75.0	CWE-798	Use of Hard-coded Credentials
[8]	75.0	CWE-311	Missing Encryption of Sensitive Data
[9]	74.0	CWE-434	Unrestricted Upload of File with Dangerous Type
[10]	73.8	CWE-807	Reliance on Untrusted Inputs in a Security Decision
[11]	73.1	CWE-250	Execution with Unnecessary Privileges
[12]	70.1	CWE-352	Cross-Site Request Forgery (CSRF)
[13]	69.3	CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')
[14]	68.5	CWE-494	Download of Code Without Integrity Check
[15]	67.8	CWE-863	Incorrect Authorization
[16]	66.0	CWE-829	Inclusion of Functionality from Untrusted Control Sphere
[17]	65.5	CWE-732	Incorrect Permission Assignment for Critical Resource
[18]	64.6	CWE-676	Use of Potentially Dangerous Function
[19]	64.1	CWE-327	Use of a Broken or Risky Cryptographic Algorithm
[20]	62.4	CWE-131	Incorrect Calculation of Buffer Size
[21]	61.5	CWE-307	Improper Restriction of Excessive Authentication Attempts
[22]	61.1	CWE-601	URL Redirection to Untrusted Site ('Open Redirect')
[23]	61.0	CWE-134	Uncontrolled Format String
[24]	60.3	CWE-190	Integer Overflow or Wraparound
[25]	59.9	CWE-759	Use of a One-Way Hash without a Salt

Reference: http://cwe.mitre.org/top25/

Today’s problems are about same as yesterday’s

Open Web Application Security Project (OWASP) Top 10	
2010	**2013**
A1 – Injection	A1 – Injection
A3 – Broken Authentication and Session Management	A2 – Broken Authentication and Session Management
A2 – Cross-Site Scripting (XSS)	A3 – Cross-Site Scripting (XSS)
A4 – Insecure Direct Object References	A4 – Insecure Direct Object References
A6 – Security Misconfiguration	A5 – Security Misconfiguration
A7 – Insecure Cryptographic Storage – Merged with A9 →	A6 – Sensitive Data Exposure
A8 – Failure to Restrict URL Access – Broadened into →	A7 – Missing Function Level Access Control
A5 – Cross-Site Request Forgery (CSRF)	A8 – Cross-Site Request Forgery (CSRF)
<buried in A6: Security Misconfiguration>	A9 – Using Known Vulnerability Components
A10 – Un-validated Redirects and Forwards	A10 – Un-validated Redirects and Forwards
A9 – Insufficient Transport Layer Protection	Merged with 2010-A7 into new 2013-A6

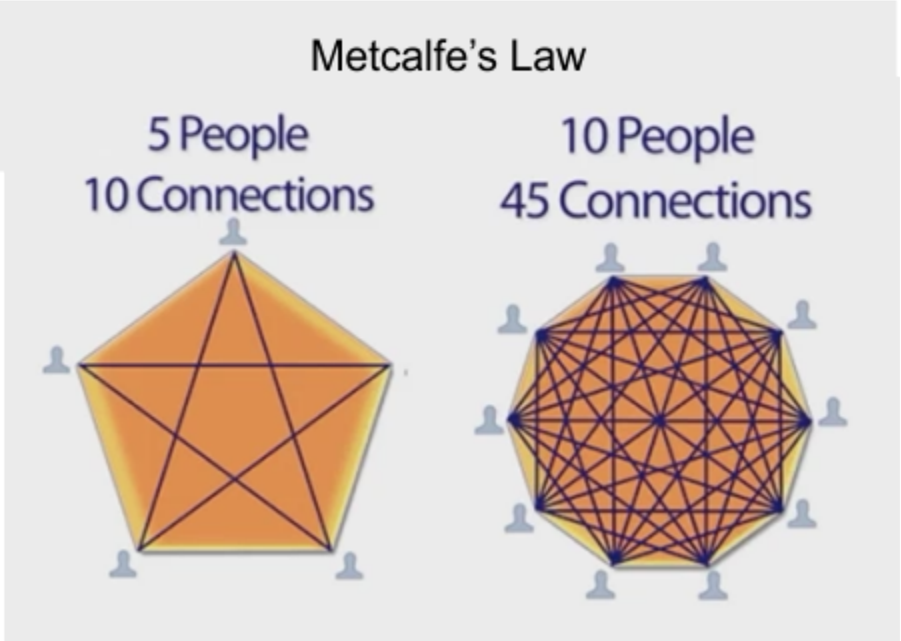
Source: OWASP Top Ten Project (https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project)

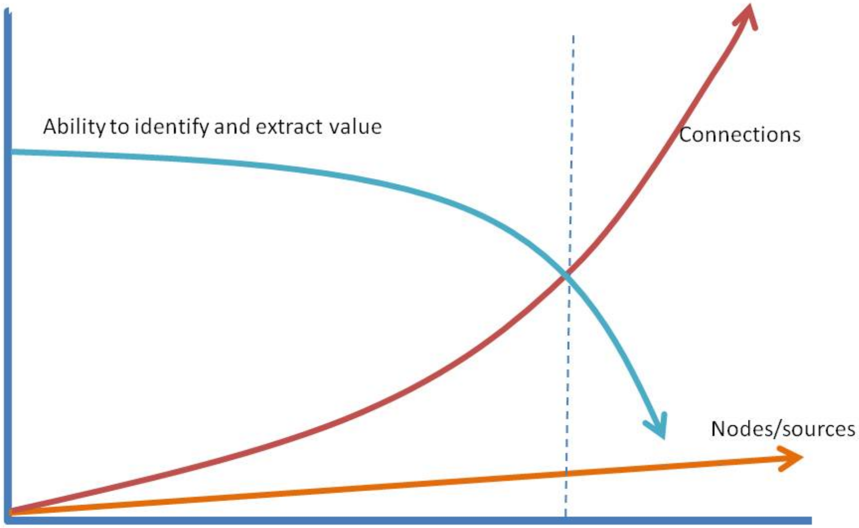
Topics

Software Development Security Domain

- ➡ Governance & Management
- System Life Cycle and Security
- Software Environment and Security Controls
- Programming Languages
- Database and DB Warehousing Vulnerabilities, Threats, and Protections
- Software Vulnerabilities and Threats

Size Matters… (1/2)



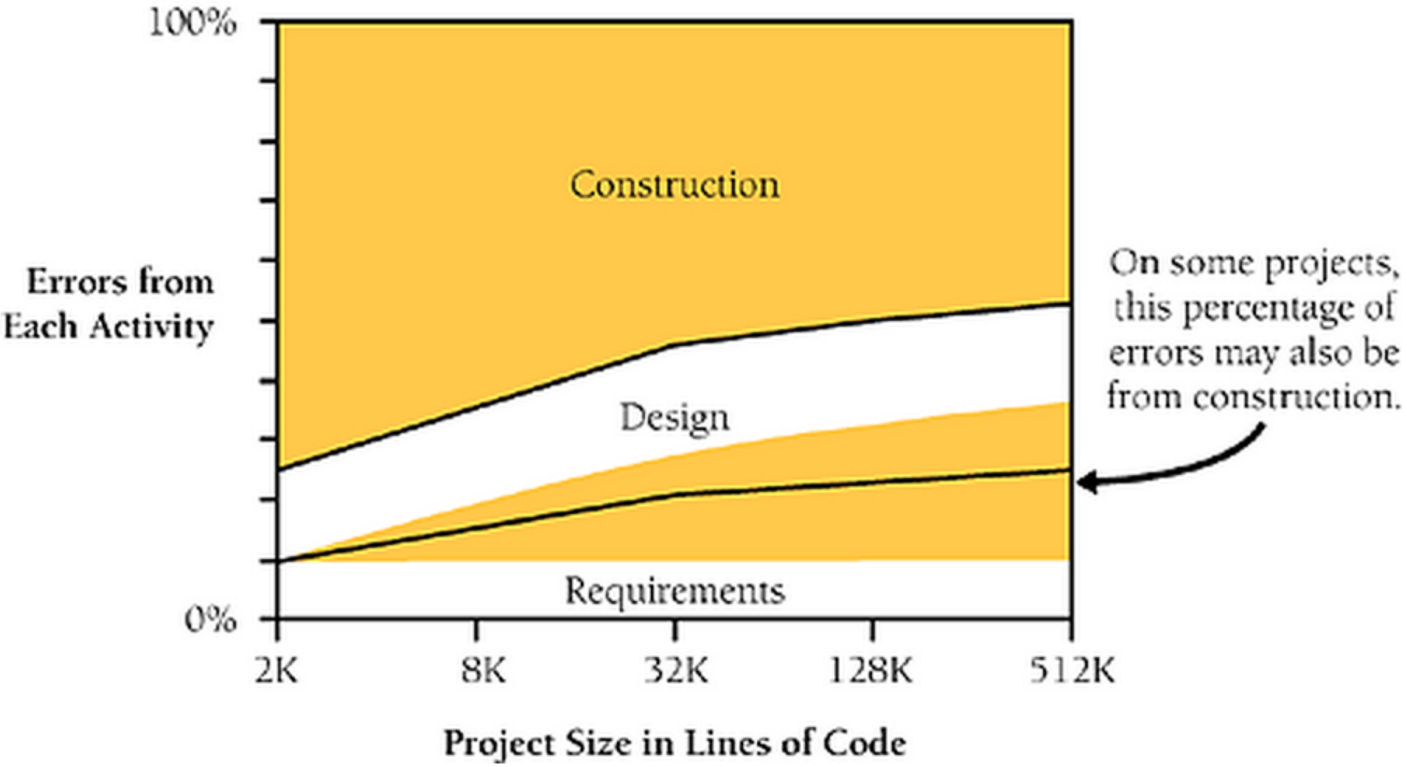


Number of connections (or interfaces) = n * (n – 1) / 2

Reference: *Code Complete: A Practical Handbook of Software Construction*, 2nd Edition, 2004

Size Matters… (2/2)

- “*As project size increases, errors usually come more from requirements and design…* (Boehm 1981, Grady 1987, Jones 1998)”

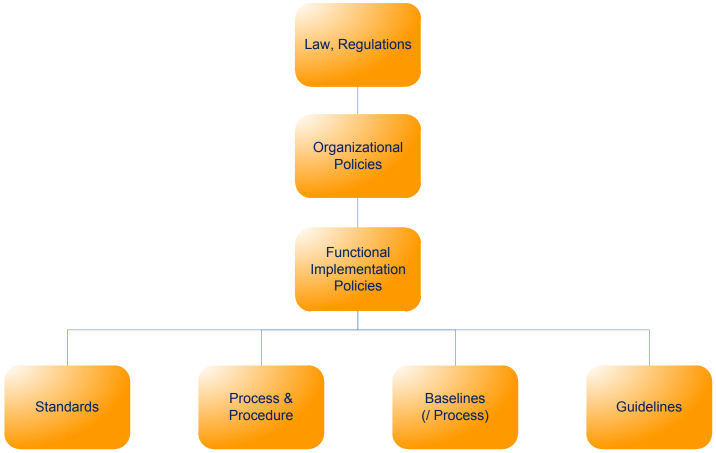


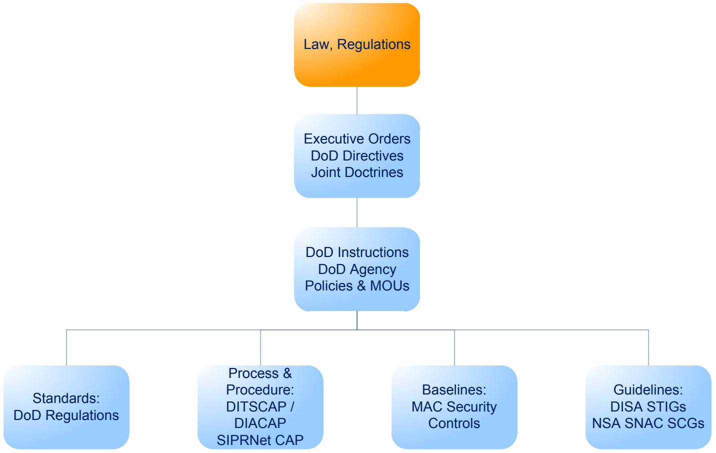
On some projects, this percentage of errors may also be from construction.

Reference: *Code Complete: A Practical Handbook of Software Construction*, 2nd Edition, 2004

Information Security Governance

- Policy. Management directives that establish expectations (goals & objectives), and assign roles & responsibilities.
- Standards. Functional specific mandatory activities, actions, and rules.
- Procedure. Step-by-step implementation instructions.
- Baseline (or Process). Mandatory description of how to implement security packages to ensure consist security posture.
- Guidelines. General statement, framework, or recommendations to augment baselines or procedures.




Law, Regulations
Executive Orders DoD Directives Joint Doctrines
DoD Instructions DoD Agency Policies & MOUs
Standards: DoD Regulations
Process & Procedure: DITSCAP / DIACAP SIPRNet CAP
Baselines: MAC Security Controls
Guidelines: DISA STIGs NSA SNAC SCGs

Clinger-Cohen Act of 1996 (CCA)

- The Clinger-Cohen Act of 1996 (a.k.a. ITMRA) defined the Federal agencies and DoD's acquisition, management, and usage of IT.
- Key Elements
  - Defines the roles & responsibilities of Federal agencies and their executives (i.e. directors and CIOs.)
  - Requires Federal agencies to implement performance and result-based management for capital planning and investment control (CPIC).
  - Defines the IT acquisition process.
  - Requires IT architecture be defined for all Federal agencies. (i.e. Federal Enterprise Architecture (FEA)).

Why CCA (/ ITMRA) necessary?

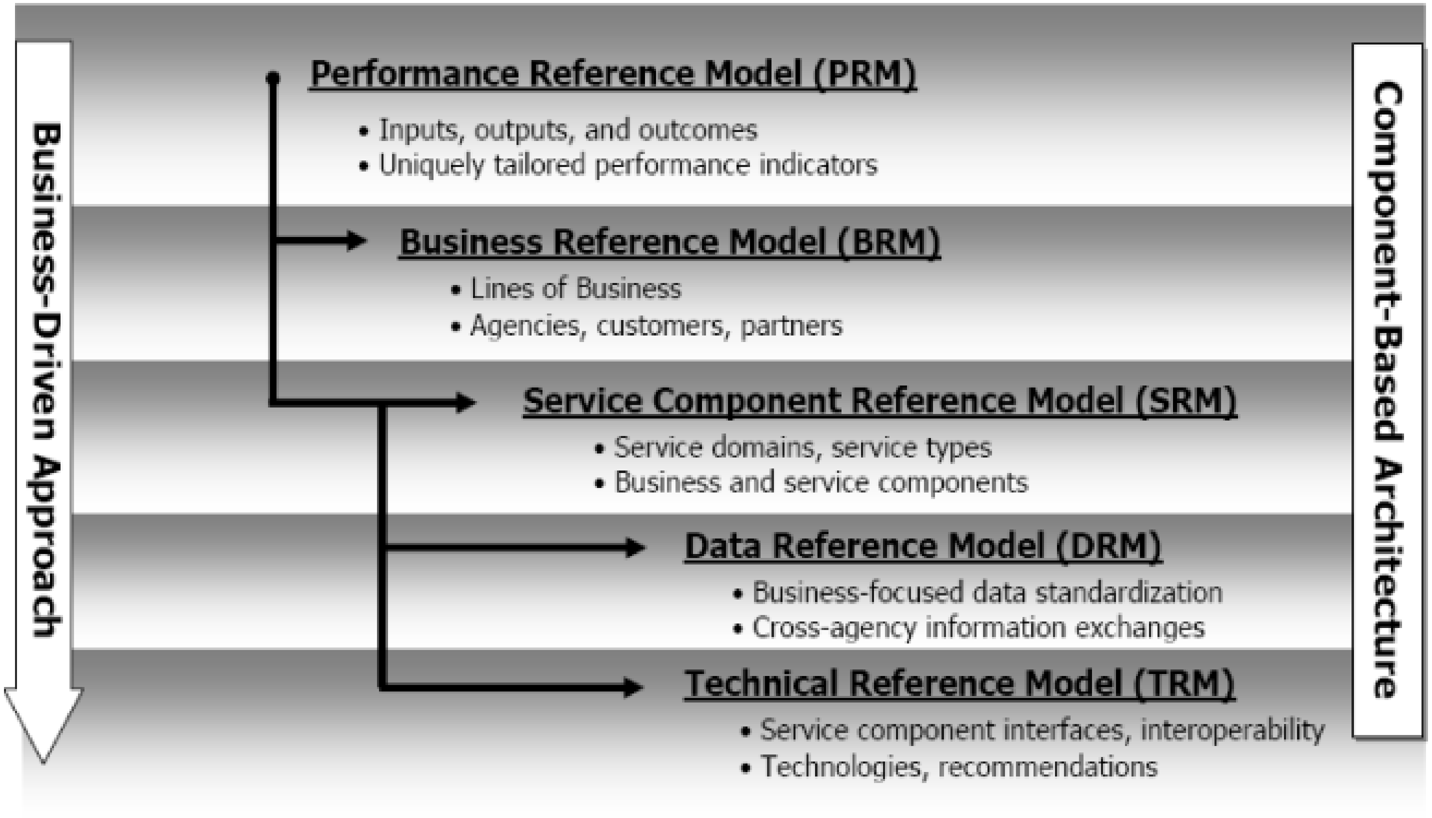
In 1992, GAO reported: "*Defense's mission-critical systems continue to have significant software development problems. Numerous GAO reports and Defense studies have identified many problems, including a lack of management attention, ill-defined system requirements, and inadequate testing. The highly complex nature of mission-critical systems and millions of lines of software required to support them contribute to the continuation of serious software development problems.*" E.g., *

- Cheyenne Mountain Upgrade (CMU), etc.
- Strategic Defense Initiative (SDI)
- Patriot surface-to-air missile system (Patriot)
- Army Tactical Command and Control System (ATCCS)
- AN/BSY-2 combat system for SSN-21 Seawolf submarine (BSY-2)
- AN/FQ-93 computer for the North American Aerospace Defense Command
- C-17 transport aircraft
- F-14D Tomcat fighter aircraft, etc.

Reference:
* GAO/IMTEC-93-13, *Defense Attempting to Address Major Software Challenges*, December 24, 1992

Federal Enterprise Architecture (FEA) Framework

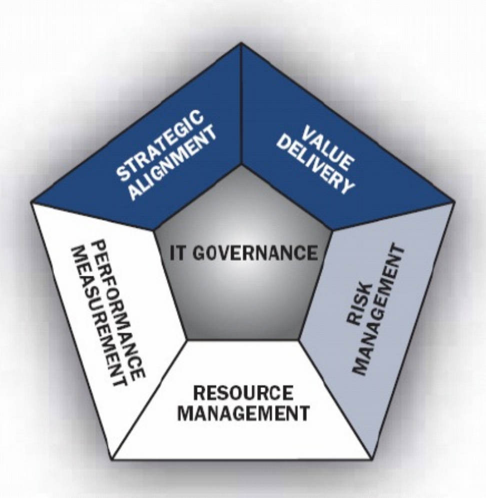
- Federal Enterprise Architecture Framework (FEAF) focuses on BUSINESS


Business-Driven Approach
Performance Reference Model (PRM)
• Inputs, outputs, and outcomes
• Uniquely tailored performance indicators
Business Reference Model (BRM)
• Lines of Business
• Agencies, customers, partners
Service Component Reference Model (SRM)
• Service domains, service types
• Business and service components
Data Reference Model (DRM)
• Business-focused data standardization
• Cross-agency information exchanges
Technical Reference Model (TRM)
• Service component interfaces, interoperability
• Technologies, recommendations
Component-Based Architecture

Reference: *Federal Enterprise Architecture Consolidated Reference Model, May 2005*

COBIT Governance Framework

- Control Objectives for Information and related Technology (COBIT) is an IT Governance Framework created by Information Systems Audit and Control Association (ISACA)
- COBIT controls can encompass:
  - Information security controls (e.g., NIST SP 800-53, CNSS 1253, ISO/IEC 27001:2005)
  - IT processes management frameworks (e.g., ITIL, CMMI, ISO/IEC 27000 IT Service Management, PMBOK)
- COBIT governance is composed of 5 focus areas:
  - Strategic alignment
  - Value delivery
  - Resource management
  - Risk management
  - Performance measurement



Reference: *COBIT 4.1* (http://www.isaca.org/)

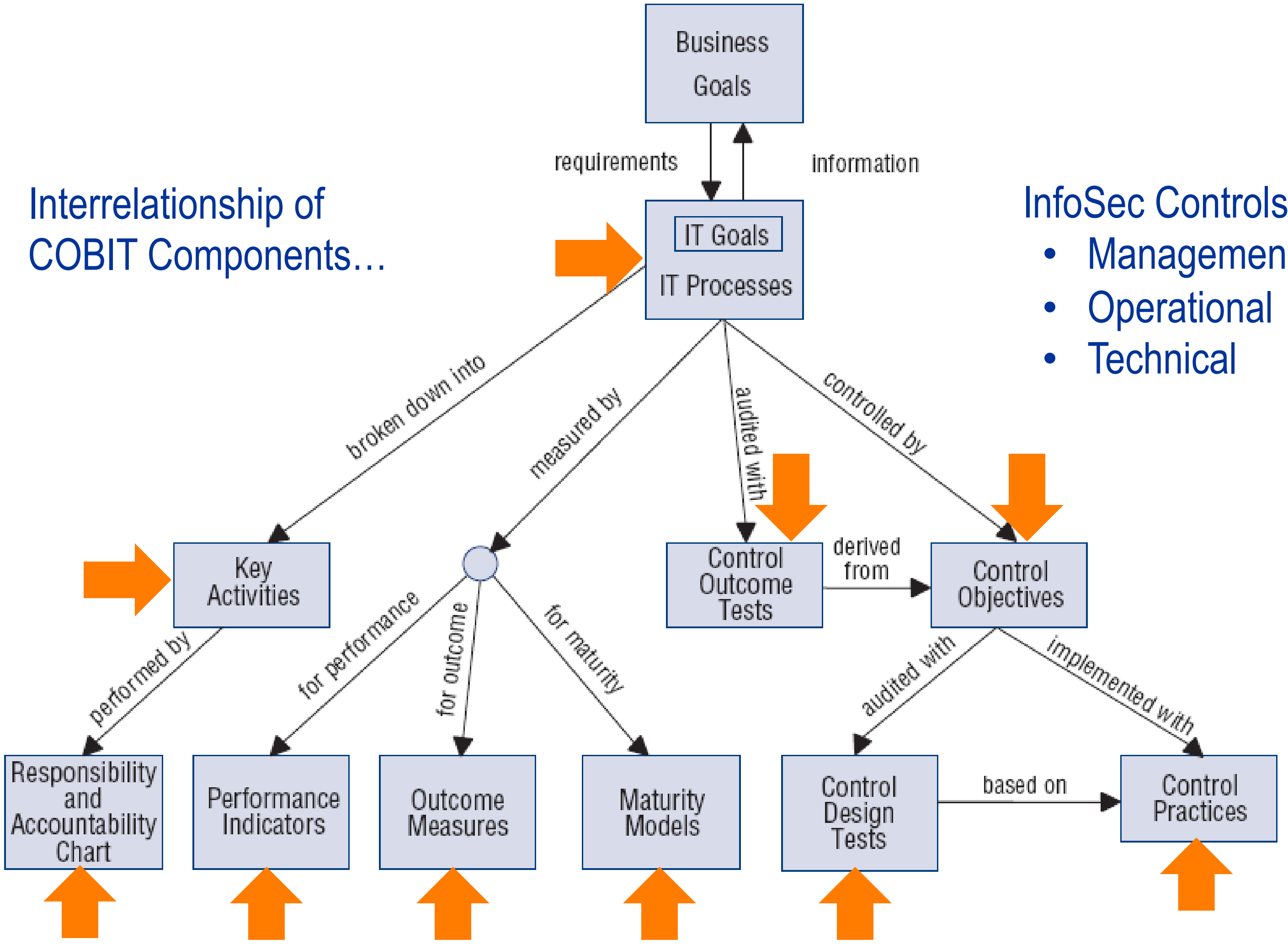
Augment IT Governance with Information Security

- Information security is an ubiquitous practice…

Interrelationship of COBIT Components…

InfoSec Controls:

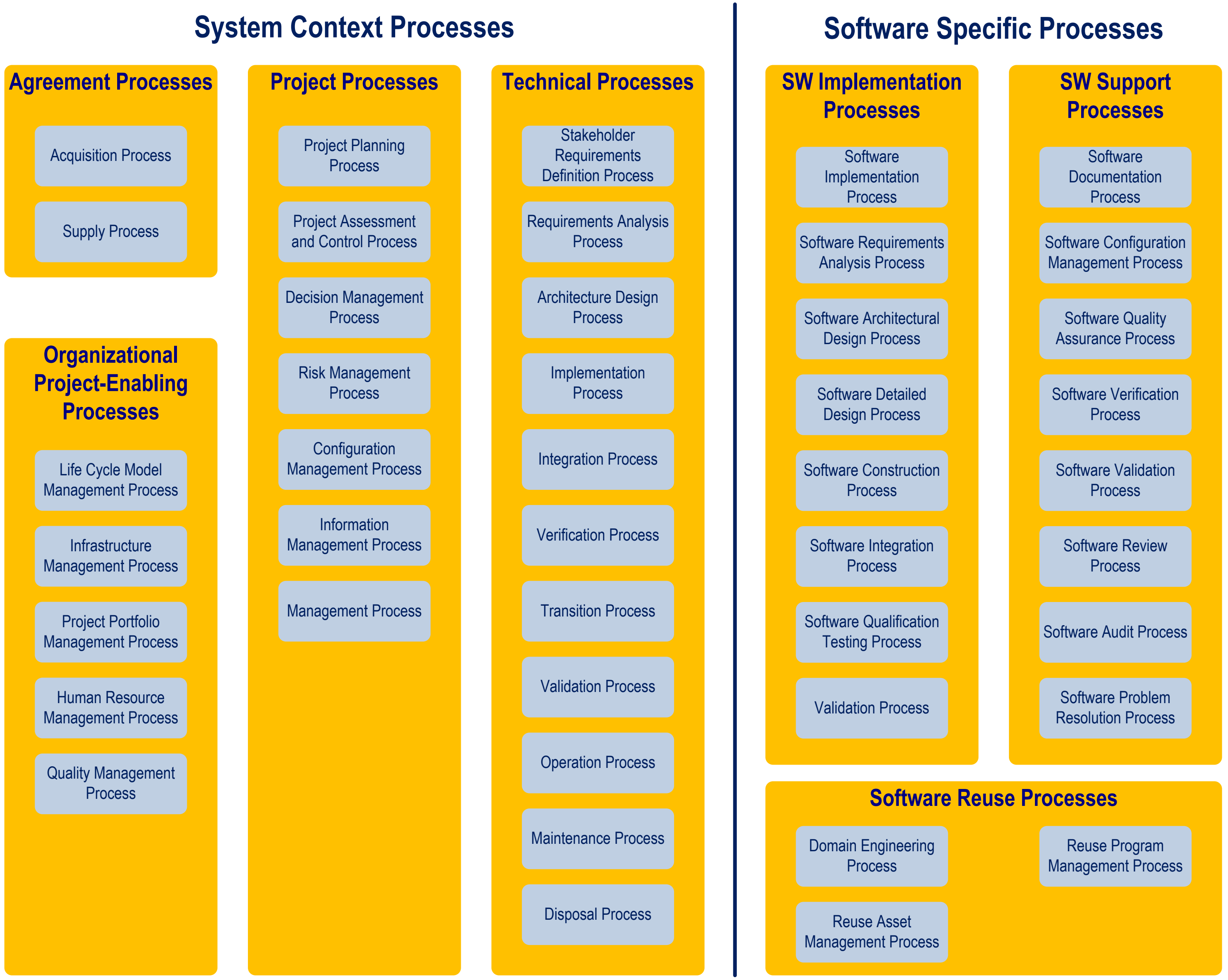
- Management
- Operational
- Technical



Reference: *COBIT 4.1* (http://www.isaca.org/)

System Life Cycle (SLC) and System Development Life Cycle (SDLC)

ISO/IEC 12207:2008, Software Life Cycle Processes



System Context Processes

Agreement Processes

- Acquisition Process
- Supply Process

Organizational Project-Enabling Processes

- Life Cycle Model Management Process
- Infrastructure Management Process
- Project Portfolio Management Process
- Human Resource Management Process
- Quality Management Process

Project Processes

- Project Planning Process
- Project Assessment and Control Process
- Decision Management Process
- Risk Management Process
- Configuration Management Process
- Information Management Process
- Management Process

Technical Processes

- Stakeholder Requirements Definition Process
- Requirements Analysis Process
- Architecture Design Process
- Implementation Process
- Integration Process
- Verification Process
- Transition Process
- Validation Process
- Operation Process
- Maintenance Process
- Disposal Process

Software Specific Processes

SW Implementation Processes

- Software Implementation Process
- Software Requirements Analysis Process
- Software Architectural Design Process
- Software Detailed Design Process
- Software Construction Process
- Software Integration Process
- Software Qualification Testing Process
- Validation Process

SW Support Processes

- Software Documentation Process
- Software Configuration Management Process
- Software Quality Assurance Process
- Software Verification Process
- Software Validation Process
- Software Review Process
- Software Audit Process
- Software Problem Resolution Process

Software Reuse Processes

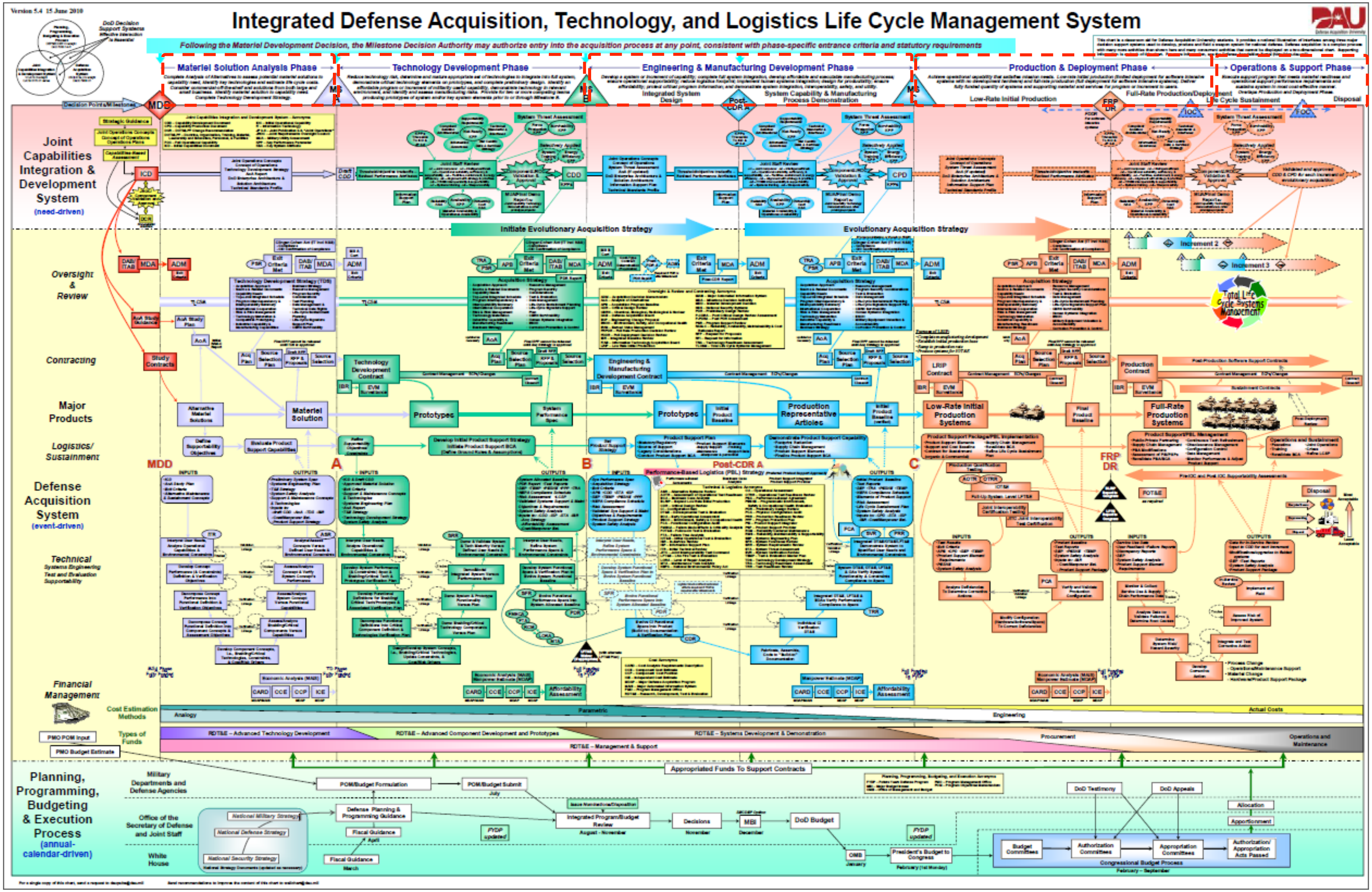
- Domain Engineering Process
- Reuse Program Management Process
- Reuse Asset Management Process

* Note: ISO/IEC 12207is identical to IEEE Std 12207

Reference: IEEE/IEC 12207:2008, *Information Technology Software Life Cycle Processes*

Governance & Management

Life Cycle Stages in Defense Acquisition System


Version 5.4 15 June 2010
Integrated Defense Acquisition, Technology, and Logistics Life Cycle Management System
DAU
DoD Decision Support Systems
Following the Materiel Development Decision, the Milestone Decision Authority may authorize entry into the acquisition process at any point, consistent with phase-specific entrance criteria and statutory requirements
Materiel Solution Analysis Phase
Technology Development Phase
Engineering & Manufacturing Development Phase
Production & Deployment Phase
Operations & Support Phase
Decision Points/Milestones
MDD
MS A
MS B
MS C
Post-CDR A
FRP DR
IOC
FOC
Integrated System Design
System Capability & Manufacturing Process Demonstration
Low-Rate Initial Production
Full-Rate Production/Deployment
Life Cycle Sustainment
Disposal
Joint Capabilities Integration & Development System
(need-driven)
ICD
CDD
CPD
Initiate Evolutionary Acquisition Strategy
Evolutionary Acquisition Strategy
Increment 2
Increment 3
Oversight & Review
Total Life Cycle Systems Management
Contracting
Study Contracts
Technology Development Contract
Engineering & Manufacturing Development Contract
LRIP Contract
Production Contract
Major Products
Alternative Materiel Solutions
Materiel Solution
Prototypes
Production Representative Articles
Low-Rate Initial Production Systems
Full-Rate Production Systems
Logistics/Sustainment
Defense Acquisition System
(event-driven)
Technical
Systems Engineering
Test and Evaluation
Supportability
INPUTS
OUTPUTS
Performance-Based Logistics (PBL) Strategy
Financial Management
Cost Estimation Methods
Analogy
Parametric
Engineering
Actual Costs
Types of Funds
RDT&E – Advanced Technology Development
RDT&E – Advanced Component Development and Prototypes
RDT&E – Systems Development & Demonstration
Procurement
Operations and Maintenance
RDT&E – Management & Support
PMO POM Input
PMO Budget Estimate
Appropriated Funds To Support Contracts
Planning, Programming, Budgeting & Execution Process
(annual-calendar-driven)
Military Departments and Defense Agencies
Office of the Secretary of Defense and Joint Staff
White House
National Military Strategy
National Defense Strategy
National Security Strategy
POM/Budget Formulation
POM/Budget Submit
Defense Planning & Programming Guidance
Fiscal Guidance
Integrated Program/Budget Review
Decisions
MBI
DoD Budget
OMB
President's Budget to Congress
DoD Testimony
DoD Appeals
Allocation
Apportionment
Budget Committees
Authorization Committees
Appropriation Committees
Authorization/Appropriation Acts Passed
Congressional Budget Process

Each Life Cycle Stage has Milestone & Review

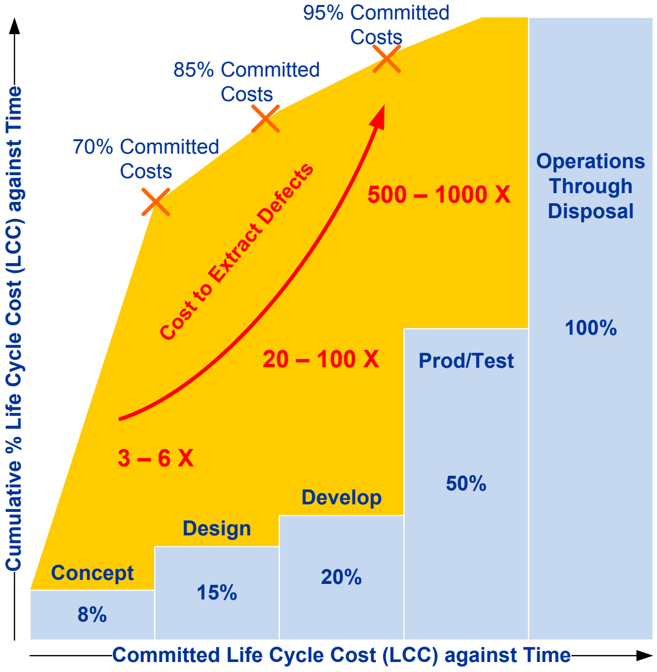
Defense Acquisition Life Cycle (DoD 5000)							
User needs & Technology Opportunities	Materiel Solution Analysis	Technology Development	Engineering & Manufacturing Development			Production and Deployment	Operations & Support
ISO/IEC 15288/IEEE 1220 Systems and Software Engineering Life Cycle							
Conceptual Design		Preliminary Design	Detailed Design & Development			Production Construction	Utilization System Support
Systems Engineering Life Cycle using Structured Analysis and Design Method							
Concept Development Stage			Engineering Development Stage			Post Development Stage	
Needs Analysis	Concept Exploration	Concept Definition	Advanced Development	Engineering Design	Integration & Evaluation	Production	Operations & Support
Typical Decision Gates	System Concept Review (SCR)	System Requirements Review (SRR)	Preliminary Design Review (PDR)	Critical Design Review (CDR)	Test Readiness Review (TRR)	Deployment Readiness Review (DRR)	Operations Readiness Review (ORR)
Information Systems Security Engineering (ISSE) Life Cycle							
Discover Information Protection Needs	Define Requirements	Design System Architecture	Develop Detailed System Design & Security Controls		Implement System & Security Controls		Continuous Monitoring
Typical C&A Decision Gates					System Certification	Security Test & Evaluation (ST&E)	System Security Authorization
Software Development: Rational Unified Process							
Inception		Elaboration		Construction		Transition	
Business Modeling	Requirements	Analysis & Design		Implementation		Deployment/CM	
McGraw's Software Security Touch Points							
Requirements and Use Cases		Architecture & Design	Test Plans	Code	Test & Test Results	Feedback From The Fields	

Focus on software structural defects

Focus on software weaknesses

Governance & SE reduces Acquisition Risks

- By Development Stage, 85% of LCC has already been committed.*
- Ratio of structural/design defects (flaws) vs. implementation weaknesses (bugs) is **50:50**.**
- If structural/design flaws have not been discovered, mitigating them will add **20 to 100 times** to the plan cost. (And up to 1000 x in Production/Test Stage.)*
- Running source code analysis tools doesn't help, because they are mostly for finding implementation weaknesses.**



Reference:

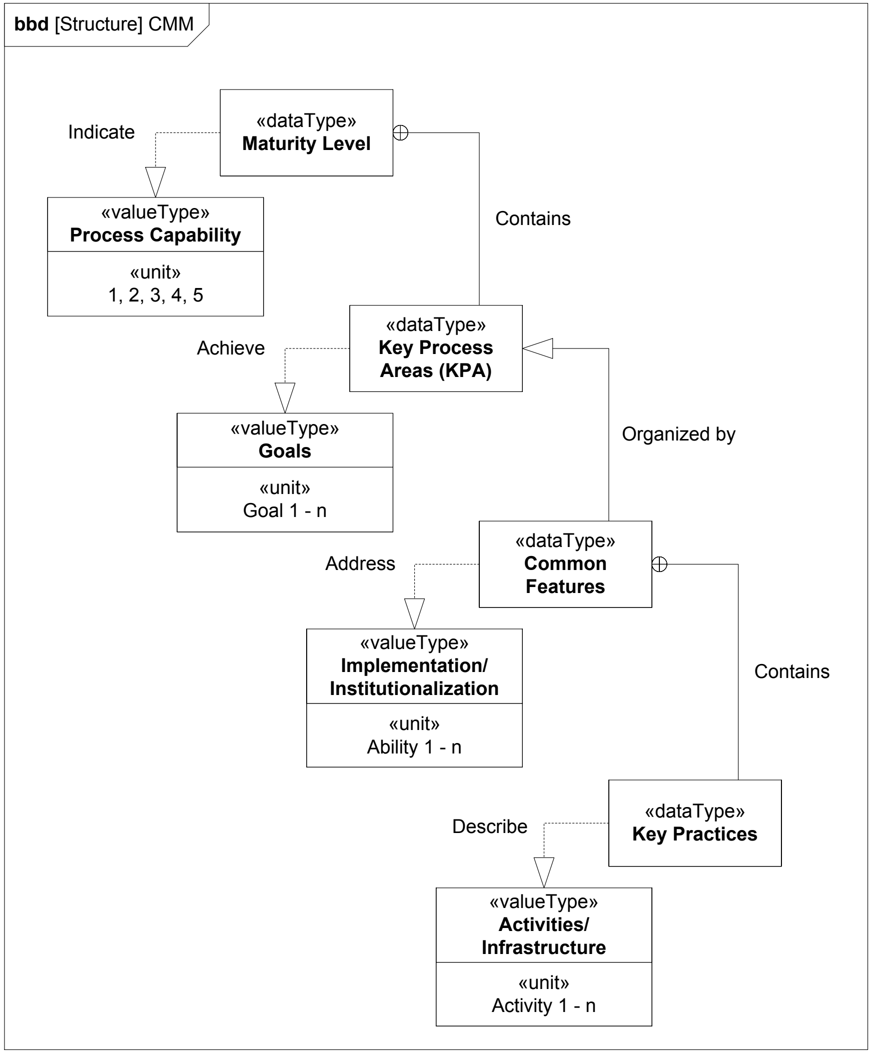
* *INCOSE Systems Engineering Handbook*, Version 3.2, 2010.

** G. McGraw, *Software Security: Building Security* In, Addison-Westley Professional, 2006. (ISBN: 978-0321356703)

Capability Maturity Model (CMM) – History

In 1986, Software Engineering Institute (SEI) and MITRE began developing an assessment framework for measuring the maturity of an organization's [system/] software engineering process.

- Process capability describes expected results.
- Process performance represents the actual results achieved.
- Process maturity is the degree which a process is explicitly defined, managed, measured, controlled, and effective.



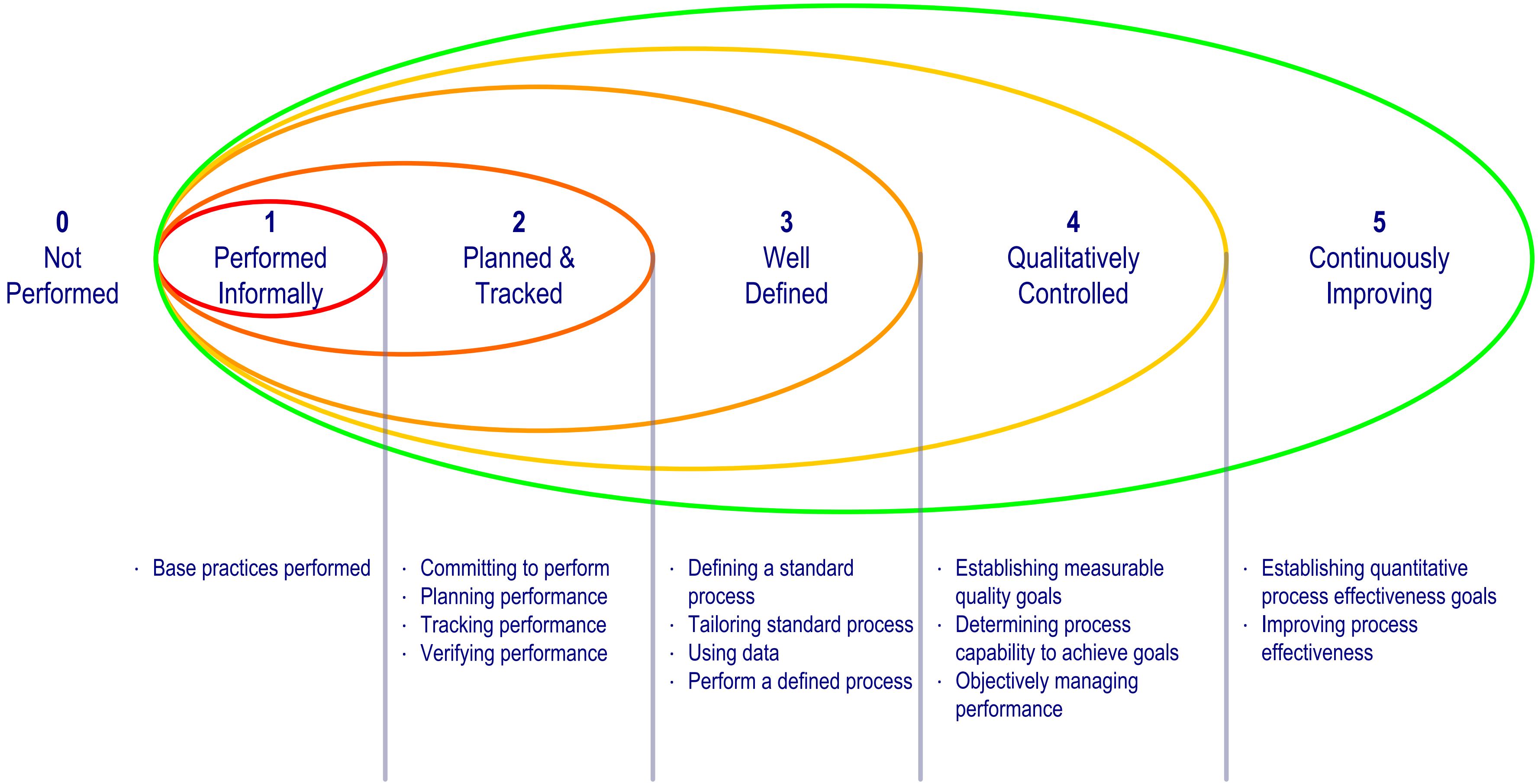
* **Reference:** M. Paulk, et al, *The Capability Maturity Model: Guidelines for Improving the Software Process*, Addison-Wesley, 1995. (ISBN: 0-201-54664-7)

Software Capability Maturity Model (SW-CMM)

- Level 1: Initial
  - The software development process is characterized as ad-hoc. Success depends on individual effort and heroics.
- Level 2: Repeatable
  - Basic project management (PM) processes are established to track performance, cost, and schedule.
- Level 3: Defined
  - Tailored software engineering and development processes are documented and used across the organization.
- Level 4: Managed
  - Detailed measures of product and process improvement are quantitatively controlled.
- Level 5: Optimizing
  - Continuous process improvement is institutionalized.

ISO/IEC 21827: SSE-CMM …(1/2)

- System Security Engineering – Capability Maturity Model (SSE-CMM)



0
Not Performed

1
Performed Informally

- Base practices performed

2
Planned & Tracked

- Committing to perform
- Planning performance
- Tracking performance
- Verifying performance

3
Well Defined

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

4
Qualitatively Controlled

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

5
Continuously Improving

- Establishing quantitative process effectiveness goals
- Improving process effectiveness

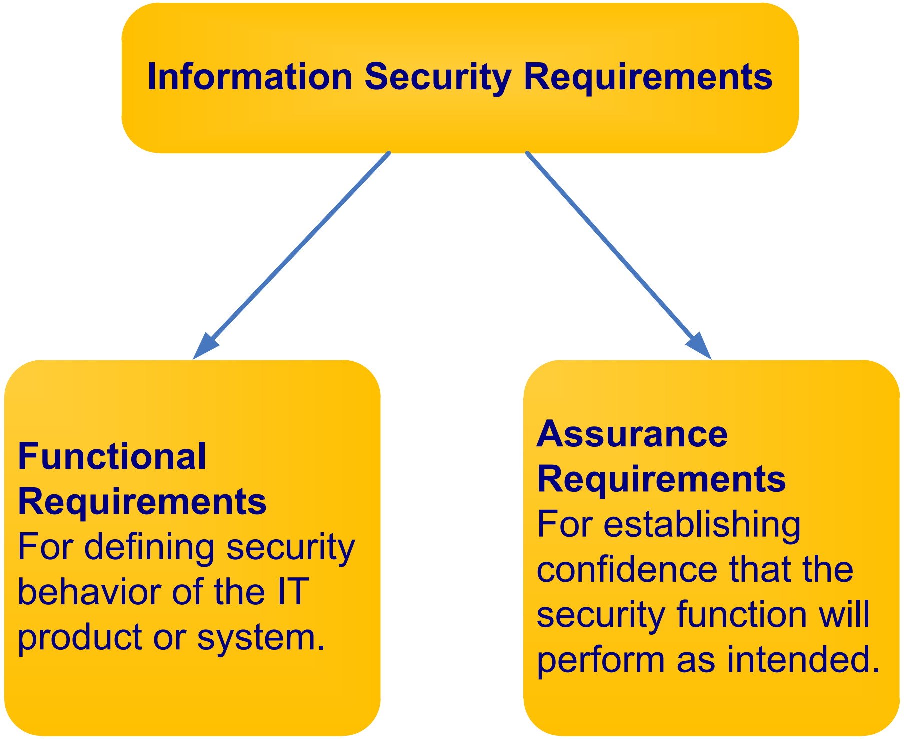
ISO/IEC 21827: SSE-CMM …(2/2)

- SSE-CMM is composed of two domains:
  - Security Base Practice (11 x Process Areas)
  - Project & Organizational Base Practice (11 x Process Areas)

- Security Base Practices
  - Administer Security Controls
  - Assess Impact
  - Assess Security Risk
  - Assess Threat
  - Assess Vulnerability
  - Build Assurance Argument
  - Coordinate Security
  - Monitor Security Posture
  - Provide Security Input
  - Specify Security Needs
  - Verify & Validate Security

- Project & Organizational Base Practices
  - Ensure Quality
  - Manage Configuration
  - Manage Project Risks
  - Monitor & Control Technical Effort
  - Plan Technical Effort
  - Define Organization’s SE Process
  - Improve Organization’s SE Process
  - Manage Product Line Evolution
  - Manage SE Support Environment
  - Provide Ongoing Skills & Knowledge
  - Coordinate with Suppliers

Measure of Effectiveness – Assurance Requirements



Information Security Requirements

Functional Requirements
For defining security behavior of the IT product or system.

Assurance Requirements
For establishing confidence that the security function will perform as intended.

- Meeting the assurance requirements is a part of "due diligence" processes.
  - Example:

    SC-3: Security Function Isolation. The information system isolates security functions from non-security functions.

- Meeting the functional requirements is a part of "due care" processes.
  - Example:
    - VLAN technology shall be created to partition the network into multiple mission-specific security domains.
    - The integrity of the internetworking architecture shall be preserved by the access control list (ACL).

Assurance Requirements – Federal Agencies

CLASS	FAMILY	IDENTIFIER
Management	Risk Assessment	RA
	Planning	PL
	System and Services Acquisition	SA
	Certification, Accreditation, and Security Assessment	CA
	Program Management	PM
Operational	Personnel Security	PS
	Physical and Environmental Protection	PE
	Contingency Planning	CP
	Configuration Management	CM
	Maintenance	MA
	System and Information Integrity	SI
	Media Protection	MP
	Incident Response	IR
	Awareness and Training	AT
Technical	Identification and Authentication	IA
	Access Control	AC
	Audit and Accountability	AU
	System and Communications Protection	SC

Reference: NIST SP800-53, Rev 3, *Recommended Security Controls for Federal Information Systems*

Assurance Requirements – DoD

DoDI 8500.2, *Information Assurance (IA) Implementation*

- Confidentiality Controls + Controls for Integrity & Availability (i.e. Mission Assurance Category (MAC))

CONFIDENTIALITY CONTROLS	INFORMATION CLASSIFICATION
E4.A4 (High)	Classified Information
E4.A5 (Medium)	Sensitive Information
E4.A6 (Basic)	Public Information

SUBJECT AREA NAME E4.A1 (MAC I) E4.A2 (MAC II) E4.A3 (MAC III)	ABBREVIATION	NUMBER OF CONTROLS IN SUBJECT AREA
Security Design & Configuration	DC	31
Identification & Authentication	IA	9
Enclave & Computing Environment	EC	48
Enclave Boundary Defense	EB	8
Physical & Environmental	PE	27
Personnel	PR	7
Continuity	CO	24
Vulnerability & Incident Management	VI	3

Assurance Requirements – Industry

ISO/IEC 27001:2005, *Information Technology – Security Techniques – Security Management System – Requirements*

CONTROL CATEGORY	SUB-CATEGORY OF CONTROLS
Security Policy	Information security policy
Organization of Information Security	Internal organization; External parties
Asset Management	Responsibility for assets; Information classification
Human Resource Security	Prior to employment; During employment; Termination or change of employment
Physical and Environmental Security	Secure areas; Equipment security
Communications and Operations Management	Operational procedures and responsibilities; Third party service delivery management; System planning and acceptance; Protection against malicious and mobile code; Back-up; Network security management; Media handling; Exchange of information; Electronic commerce services; Monitoring
Access Control	Business requirement for access control; User access management; User responsibilities; Network access control; Operating system access control; Application and information access control; Mobile computing and teleworking
Information Systems Acquisition, Development, and Maintenance	Security requirements of information systems; Correct processing in applications; Cryptographic controls; Security of system files; Security in development and support processes; Technical vulnerability management
Information Security Incident Management	Reporting information security events and weaknesses; Management of information security incidents and improvements
Business Continuity Management	Information security aspects of business continuity management
Compliance	Compliance with legal requirements; Compliance with security policies and standards, and technical compliance; Information system audit considerations

Assurance Requirements – Credit Card Payment Industry

Payment Card Industry – Data Security Standard (PCI-DSS), *Requirements and Security Assessment Procedures*, Version 2.0, October 2010

Assessment Procedures	Requirements
Build and Maintain a Secure Network	Req. 1: Install and maintain a firewall configuration to protect cardholder data. Req. 2: Do not use vendor-supplied defaults for system passwords and other security parameters.
Protect Cardholder Data	Req. 3: Protect stored cardholder data. Req. 4: Encrypt transmission of cardholder data across open, public networks.
Maintain a Vulnerability Management Program	Req. 5: Use and regularly update anti-virus software or programs. Req. 6: Develop and maintain secure systems and applications.
Implement Strong Access Control Measures	Req. 7: Restrict access to cardholder data by business need to know. Req. 8: Assign a unique ID to each person with computer access. Req. 9: Restrict physical access to cardholder data.
Regular Monitor and Test Network	Req. 10: Track and monitor all access to network resources and cardholder data. Req. 11: Regular test security systems and processes.
Maintain an Information Security Policy	Req. 12: Maintain a policy that addresses information security for all personnel.

Assurance Requirements – Other PCI Security Standards

- *Payment Application Data Security Standard (PA-DSS) Requirement and Security Assessment Procedure*, Version 2.0, October 2010
- Payment Card Industry PIN Transaction Security (PCI PTS)
  - *PIN Security Requirements*, Version 1.0, September 2011.
  - *Hardware Security Module (HSM)*, Version 1.0, April 2009.
  - *Point of Interaction (POI) Modular Security Requirements*, Version 3.1, October 2011.
- Payment Card Industry Point-to-Point Encryption (PCI P2PE)
  - *P2PE Hardware Solution Requirements and Testing Procedures*, April 2012.

Reference: *PCI Security Standards Council*, (https://www.pcisecuritystandards.org/)

Topics

Software Development Security Domain

- Governance & Management
- ➡ System/Software Life Cycle and Security
- Software Environment and Security Controls
- Programming Languages
- Database and DB Warehousing Vulnerabilities, Threats, and Protections
- Software Vulnerabilities and Threats

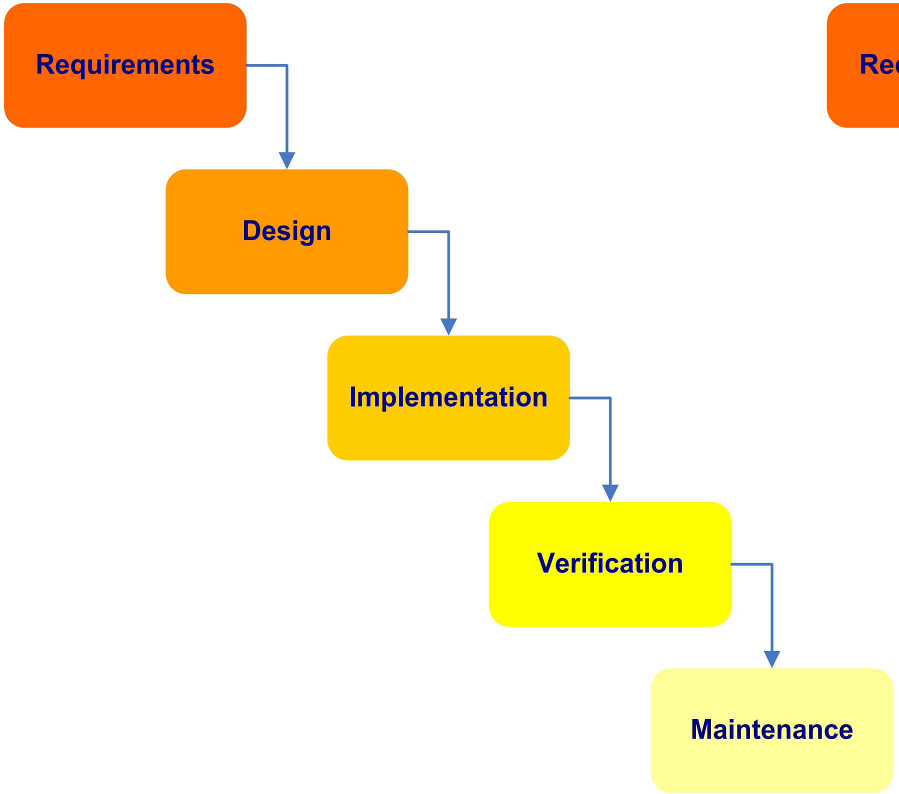
System Development Life Cycle (SDLC) Models and Processes

- Waterfall Development Models
  - Waterfall: DoD-STD-2167A (replaced by MIL-STD-498 on 11/1994).
  - Modified Waterfall: MIL-STD-498 (cancelled on 5/1998)
- Iterative Development Models
  - Boehm's Spiral Model.
  - Rapid Application Development (RAD) & Joint Application Development (JAD)
- SDLC Processes
  - ISO/IEC 12207, *Software Life Cycle Processes* (IEEE/EIA 12207 US implementation) (based on MIL-STD-499B)
  - ISO/IEC 15288, *Systems Engineering – System Life Cycle Processes* (IEEE std 1220 – 2005, US implementation)

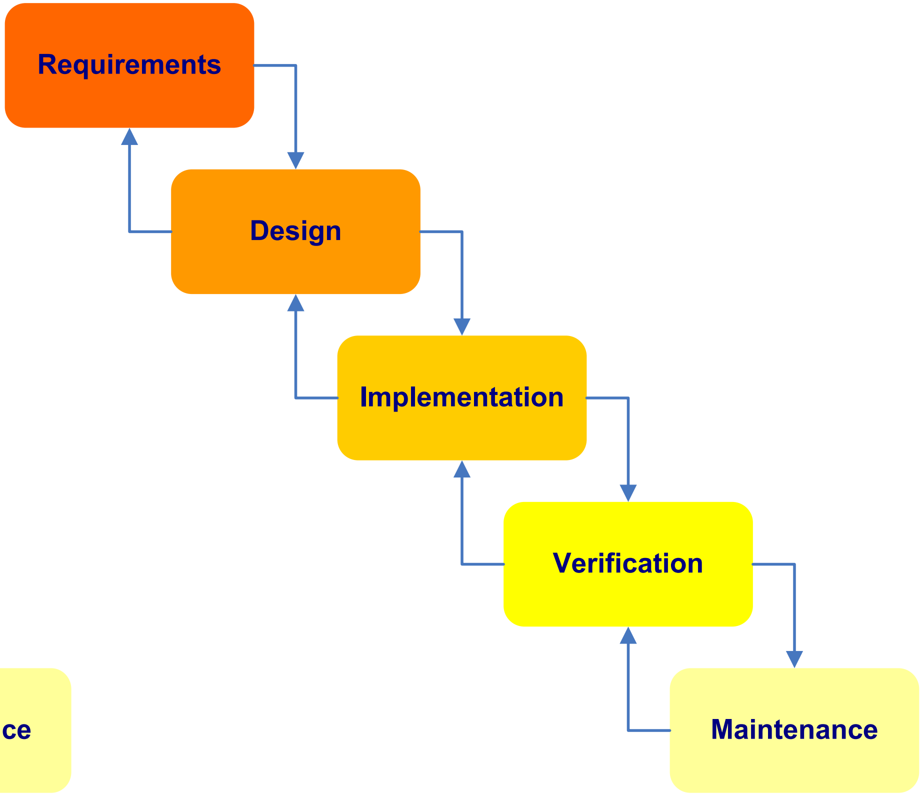
System/Software Development Life Cycle (SDLC)

Waterfall Development Models

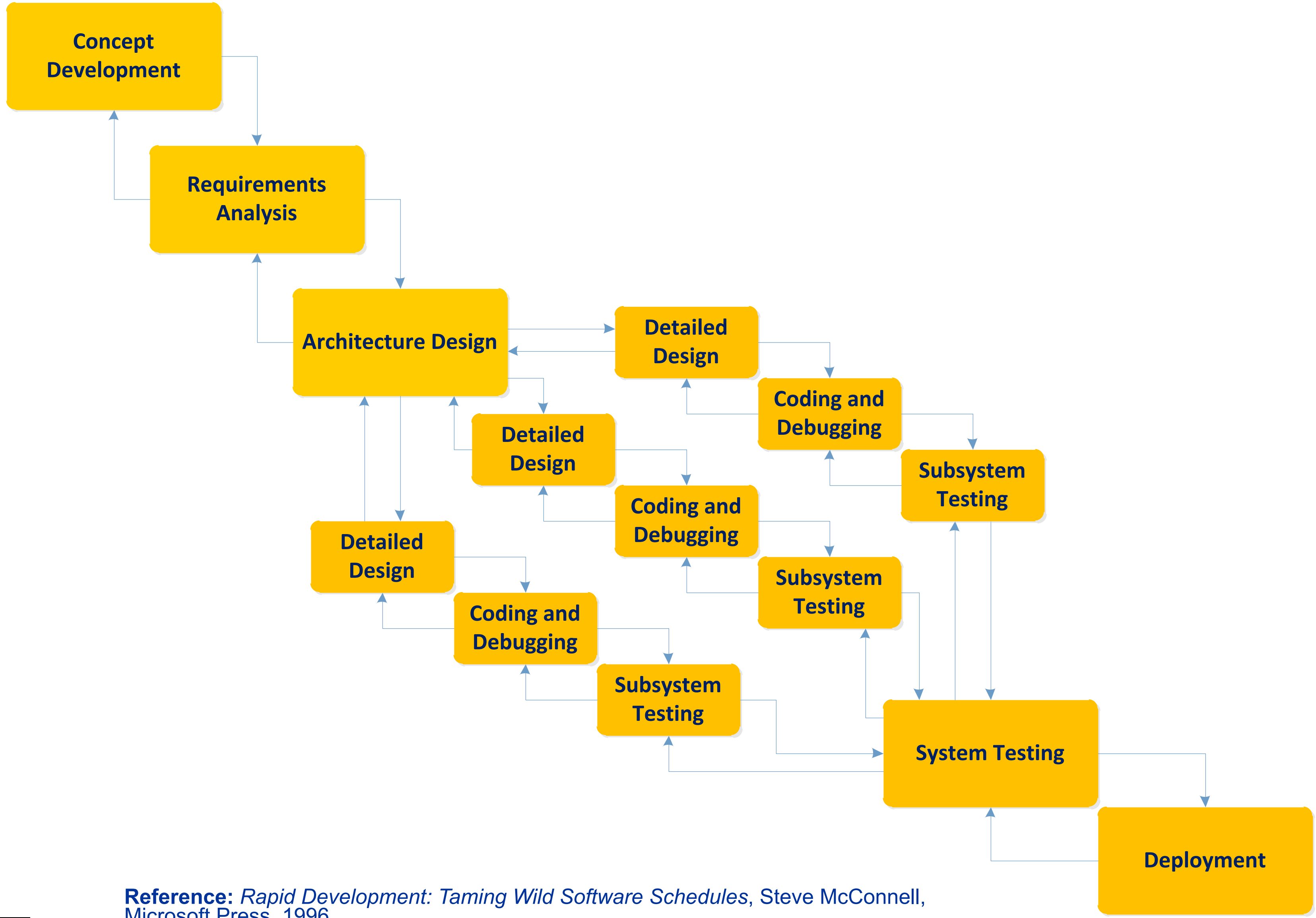
- Classic Waterfall: DoD-STD-2167A



- Modified Waterfall: MIL-STD-498

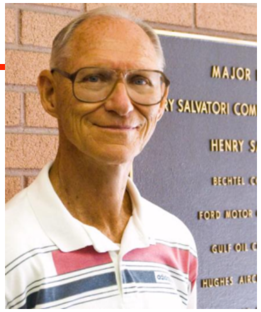


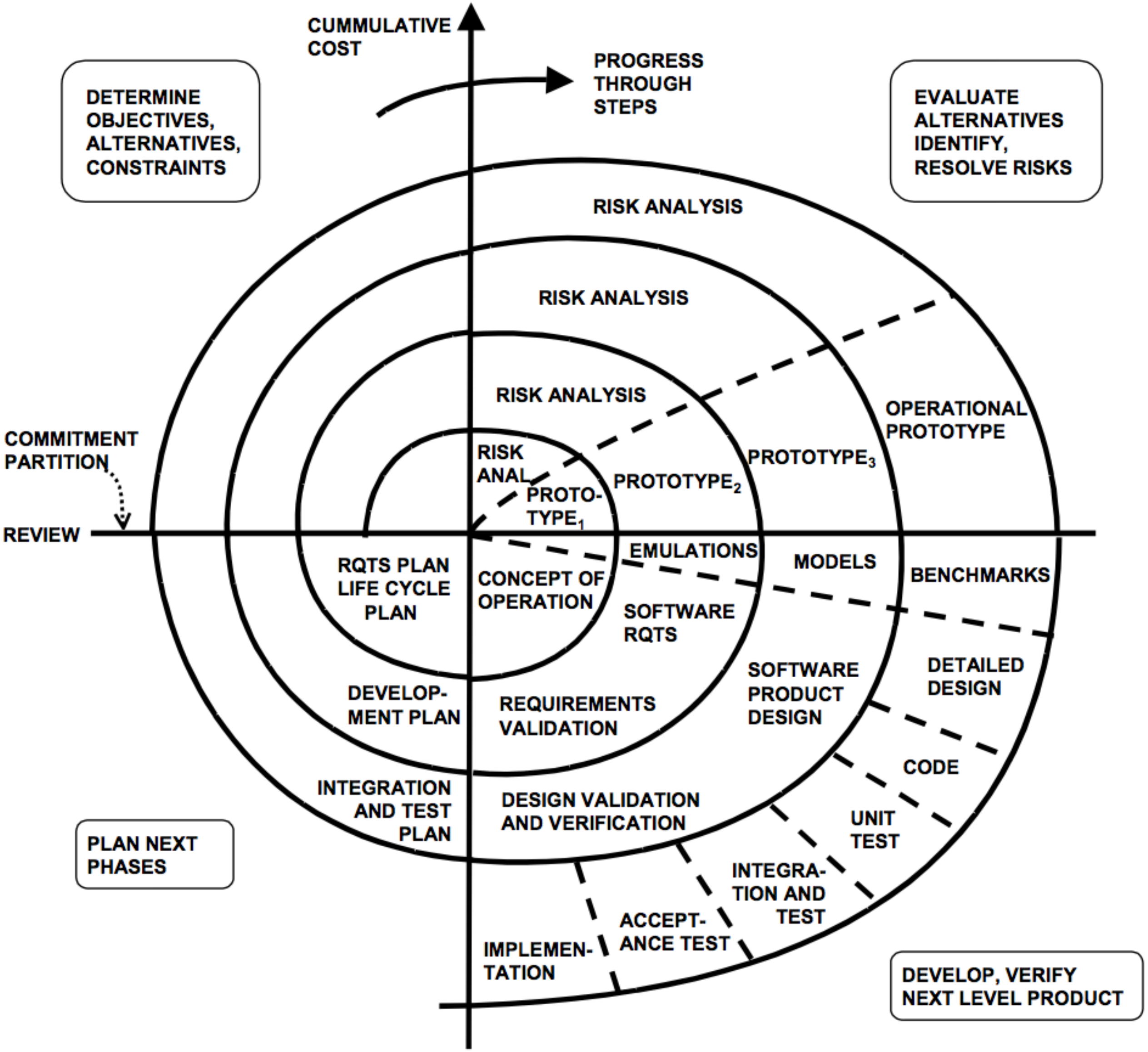
Other SDLC Models – Modified Waterfall w/ Subprojects



Reference: *Rapid Development: Taming Wild Software Schedules*, Steve McConnell, Microsoft Press, 1996

Boehm’s Spiral Model

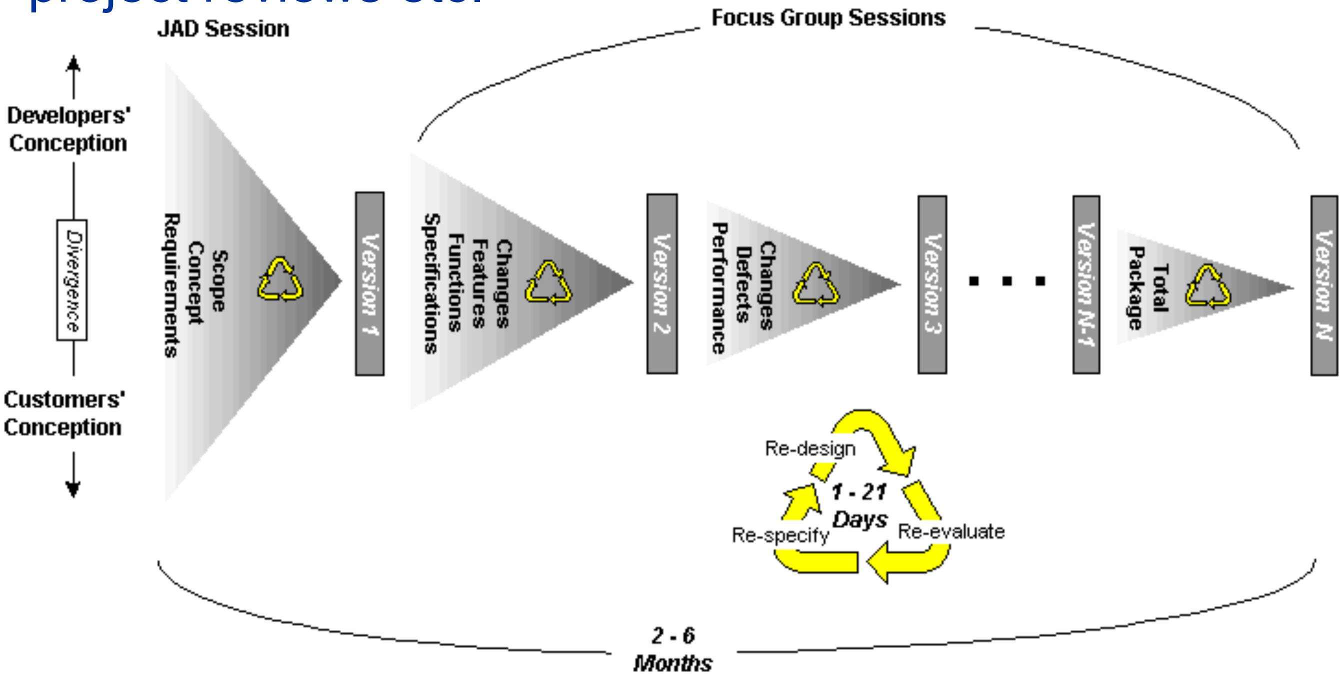




Reference: http://csse.usc.edu/people/barry.html

Rapid Application Development (RAD) Model

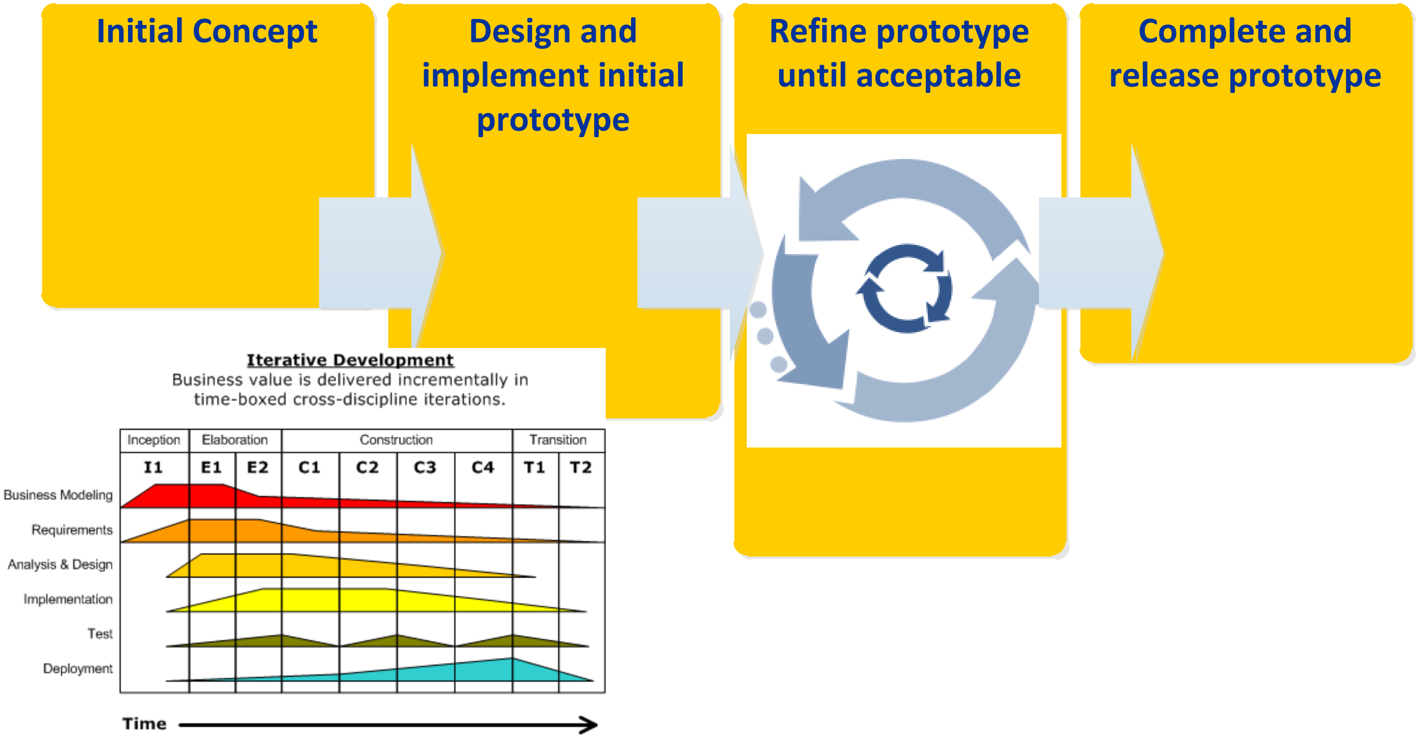
- Iterative, but spiral cycles are much smaller.
- Risk-based approach, but focus on “good enough” outcome.
- SDLC fundamentals still apply…
  - Requirements, configuration, and quality management, design process, coding, test & integration, technical and project reviews etc.



Reference:
- S. McConnel, *Rapid Development: Taming Wild Software Schedules*
- http://www.cs.bgsu.edu/maner/domains/RAD.htm

Evolutionary Prototyping Model

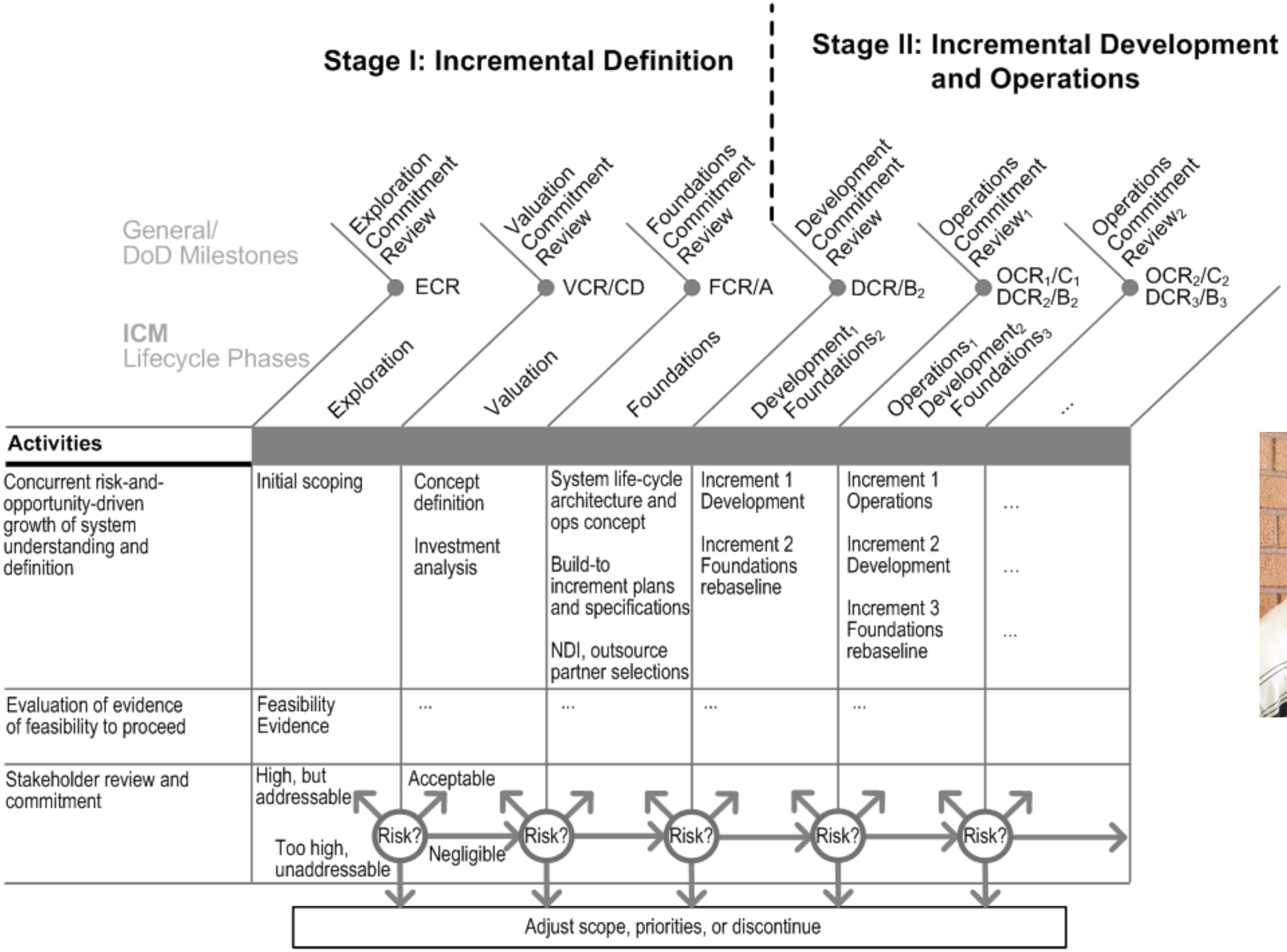
- The system concept is refined continuously…
  - The focus is on "good enough" concept, requirements, and prototype.
  - However, it is difficult to determine level of effort (LOE), cost, and schedule.


Initial Concept
Design and implement initial prototype
Refine prototype until acceptable
Complete and release prototype
Iterative Development
Business value is delivered incrementally in time-boxed cross-discipline iterations.
Inception
Elaboration
Construction
Transition
I1
E1
E2
C1
C2
C3
C4
T1
T2
Business Modeling
Requirements
Analysis & Design
Implementation
Test
Deployment
Time

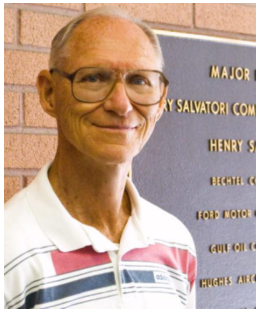
Reference: *Rapid Development: Taming Wild Software Schedules*, Steve McConnell, Microsoft Press, 1996

System/Software Development Life Cycle (SDLC)

Incremental Commitment Model


Stage I: Incremental Definition
Stage II: Incremental Development and Operations
General/ DoD Milestones
Exploration Commitment Review
Valuation Commitment Review
Foundations Commitment Review
Development Commitment Review
Operations Commitment Review$_1$
Operations Commitment Review$_2$
ECR
VCR/CD
FCR/A
DCR/B$_2$
OCR$_1$/C$_1$ DCR$_2$/B$_2$
OCR$_2$/C$_2$ DCR$_3$/B$_3$
ICM Lifecycle Phases
High, but addressable
Acceptable
Risk?
Negligible
Too high, unaddressable
Adjust scope, priorities, or discontinue

Activities	Exploration	Valuation	Foundations	Development$_1$ Foundations$_2$	Operations$_1$ Development$_2$ Foundations$_3$	...
Concurrent risk-and-opportunity-driven growth of system understanding and definition	Initial scoping	Concept definition Investment analysis	System life-cycle architecture and ops concept Build-to increment plans and specifications NDI, outsource partner selections	Increment 1 Development Increment 2 Foundations rebaseline	Increment 1 Operations Increment 2 Development Increment 3 Foundations rebaseline	... ...
Evaluation of evidence of feasibility to proceed	Feasibility Evidence	...	...	...	...	
Stakeholder review and commitment						

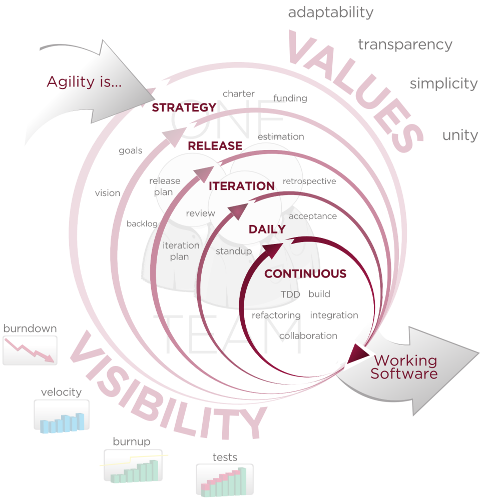


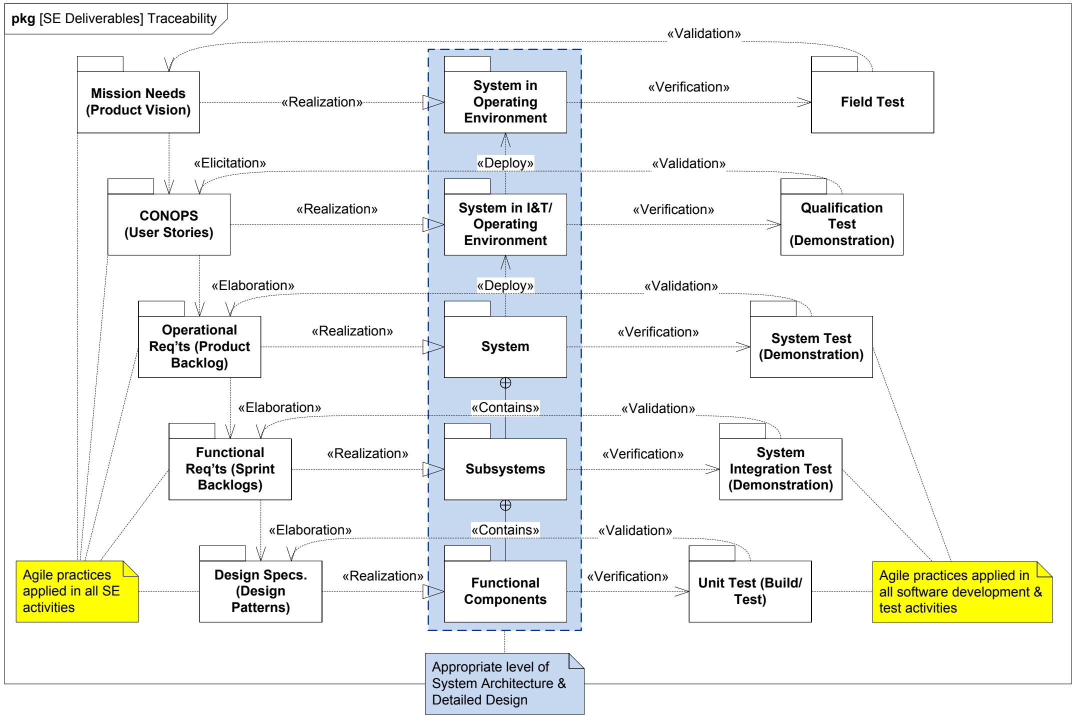
Reference: B. Boehm, J.A. Lane, *Using the Incremental Commitment Model to Integrate System Acquisition, Systems Engineering, and Software Engineering,* CrossTalk, October 2007.

System/Software Development Life Cycle (SDLC)

The need for speed... Agile Development Approach

Project Terms	Agile Terms
MNS	Vision
CONOPS	User Stories
SDP	Release & Iteration Plans, Backlogs
PMR/MS Reviews	Retrospectives, Product Demo





pkg [SE Deliverables] Traceability

Mission Needs (Product Vision)

CONOPS (User Stories)

Operational Req'ts (Product Backlog)

Functional Req'ts (Sprint Backlogs)

Design Specs. (Design Patterns)

System in Operating Environment

System in I&T/ Operating Environment

System

Subsystems

Functional Components

Field Test

Qualification Test (Demonstration)

System Test (Demonstration)

System Integration Test (Demonstration)

Unit Test (Build/ Test)

«Validation»

«Realization»

«Verification»

«Elicitation»

«Deploy»

«Elaboration»

«Contains»

Agile practices applied in all SE activities

Agile practices applied in all software development & test activities

Appropriate level of System Architecture & Detailed Design

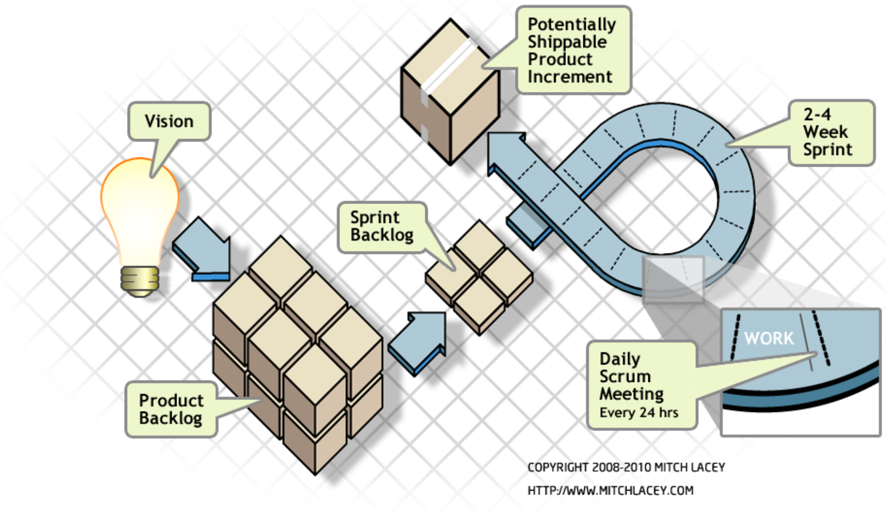
Agile SDLC Model – Scrum

- Scrum is an agile software development methodology and model that is both iterative and incremental.
- The concept derived from the development of commercial products, where:
  – Product owner provides the vision and roadmap;
  – Scrum master specifies activities and ensures deliverables meet the sprint and iteration goals;
  – Team executes the specified scrum activities.
- The process is executed in a series of "time-boxed" sprints and iterations, where:
  – A "sprint" is usually 2 to 4 weeks; and
  – The end-product is a "iteration".

Reference:

- T. Hirotaka, N. Ikujiro, *The New Product Development Game*, Harvard Business Review, January, 1986. (http://hbr.org/product/new-new-product-development-game/an/86116-PDF-ENG)
- J. Sutherland, *Agile Development: Lessons Learned from the First Scrum*, 2004-10. (http://www.scrumalliance.org/resources/35)
- R. Carlson, P.J. Matuzic, R.L. Simons, *Applying Scrum to Stabilize Systems Engineering Execution*, CrossTalk, May/June 2012.

Agile SDLC Model – Scrum


Potentially Shippable Product Increment
Vision
2-4 Week Sprint
Sprint Backlog
WORK
Product Backlog
Daily Scrum Meeting Every 24 hrs

COPYRIGHT 2008-2010 MITCH LACEY
HTTP://WWW.MITCHLACEY.COM

- The product vision is translated into a list of project requirements;
- This "list" is called the product backlog. It encompasses all the project requirements and work;
- The scrum master works with the product owner to plan and divide the product backlog into a series of sprint backlog.
- The self-organized team composed of domain and SMEs. The team is empowered to select, plan, and make decisions on its work task
- The daily stand-up team meeting is called the daily-scrum. It keeps the team members focused on their tasks. Both product owner and scrum master are required to participate.

Reference:

- R. Carlson, P.J. Matuzic, R.L. Simons, *Applying Scrum to Stabilize Systems Engineering Execution*, CrossTalk, May/June 2012.

Are there other SDLC models?

DevOps*

- Idea observed from cloud computing...
- 2009, Flickr reported doing 10 deployments per day
- Amazon EC2 reported in May 2011:**
  - Mean time between deployments: 11.6 seconds
  - Maximum # of deployments in an hour: 1,079
  - Mean # of hosts can simultaneously receive a deployment: 10k
  - Maximum # of hosts can simultaneously receive a deployment: 30k
  - http://youtu.be/o7-IuYS0iSE ***

Reference:

* J. Gorman, G. Kim, *Security is Dead. Long Live Rugged DevOps: IT at Ludicrous Speed*, RSA Conference 2012 (http://www.slideshare.net/realgenekim/security-is-dead-long-live-rugged-devops-it-at-ludicrous-speed)
** Jon Jenkins, Velocity Culture, O'Reilly Velocity 2011, (http://www.youtube.com/watch?v=dxk8b9rSKOo)
*** D. Edwards, *The (Short) History of DevOps*, Sept. 17, 2012. (http://youtu.be/o7-IuYS0iSE)

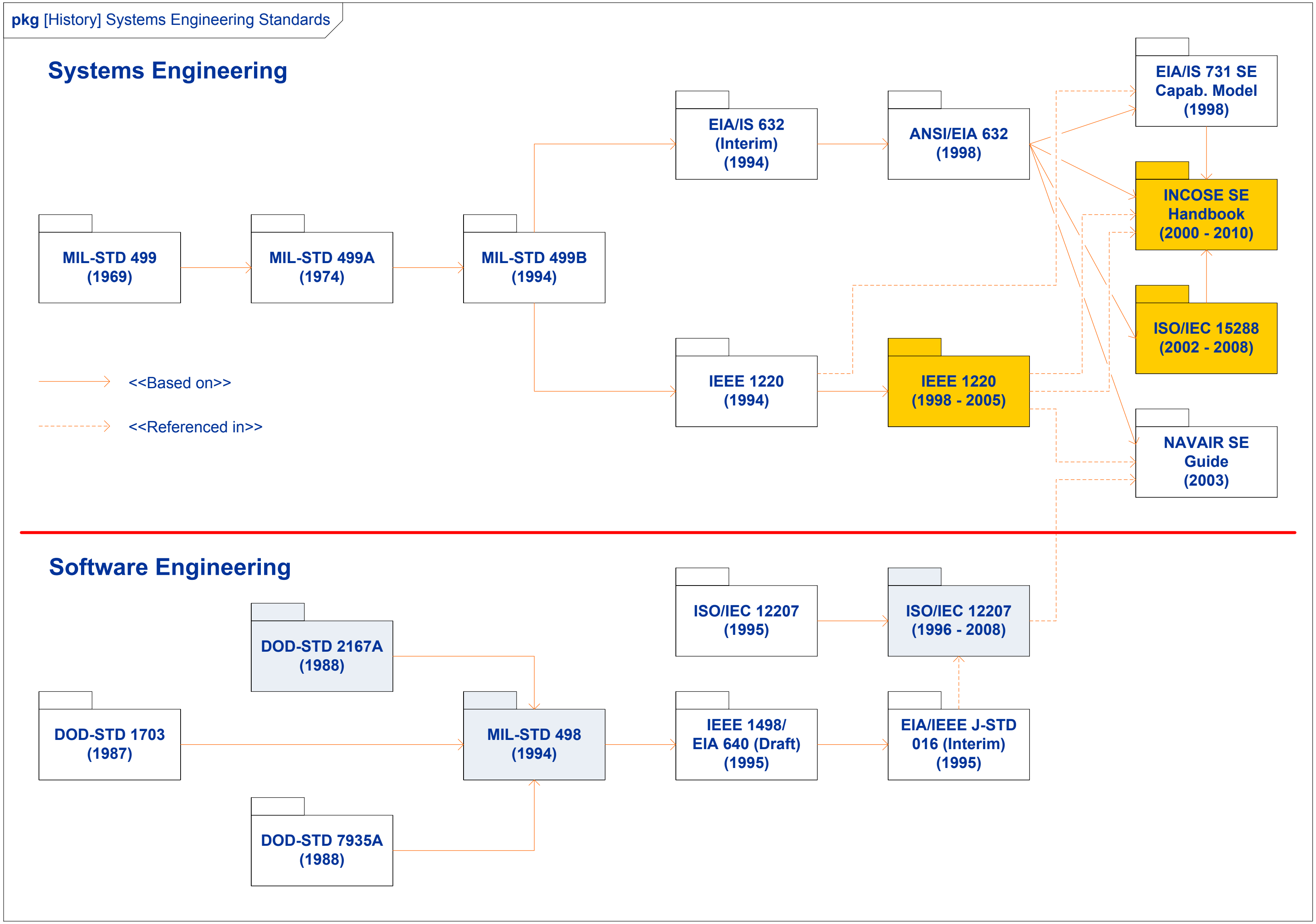
Philosophy behind the Rugged DevOps

- Seamless integration of software development and IT operations
- Focus on the “big picture” rather than security controls
  - Standard configuration
  - Process discipline
  - Controlled access to production systems
- Results
  - 75% reduction in outages triggered by software deployment since 2006
  - 90% reduction in outage minutes triggered by software deployments
  - Instantaneous automated rollback
  - Reduction in complexity
- Back to our study...

Reference:

- Jon Jenkins, *Velocity Culture*, O’Reilly Velocity 2011, (http://www.youtube.com/watch?v=dxk8b9rSKOo)

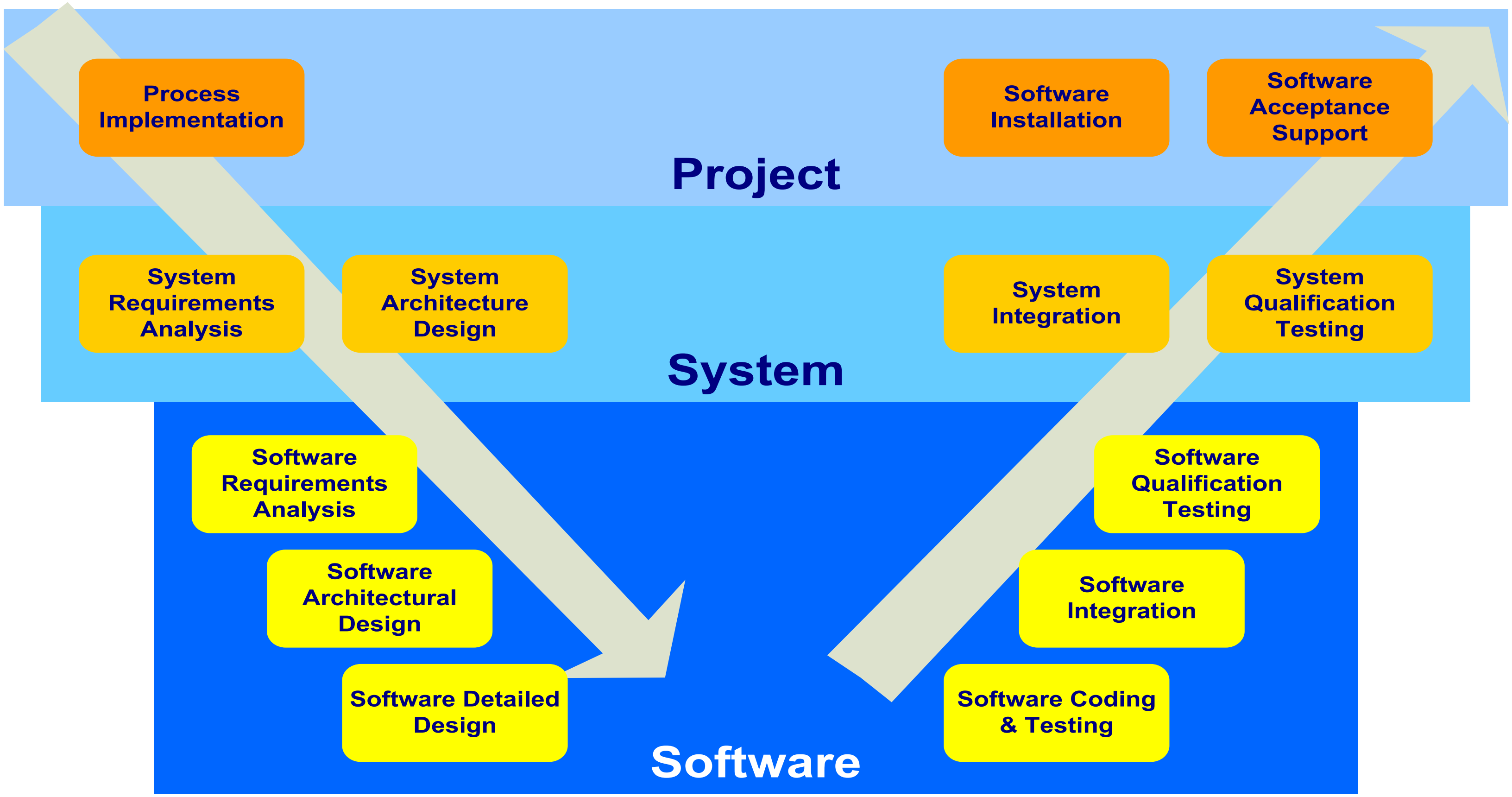
History of Systems/Software Engineering Process Standards



Software & System Engineering Management Processes

- There are more and more "software-intensive" systems…
  - Systems are getting more complex. Hardware problems are often addressed through software;
  - Operating environments are stochastic. Software are more flexible than hardware.
- As SDLC models evolves, management processes are evolving too…
  - DoD-STD-2167A: Waterfall SDLC + SE Process
  - MIL-STD-498: Modified Waterfall SDLC + SE Process
  - IEEE 1220: System Engineering Process
  - ISO 12207: Software + System Engineering Mgmt. Process
  - ISO 15288: System Engineering Mgmt. Process

DoD-STD-2167A – System Engineering Process

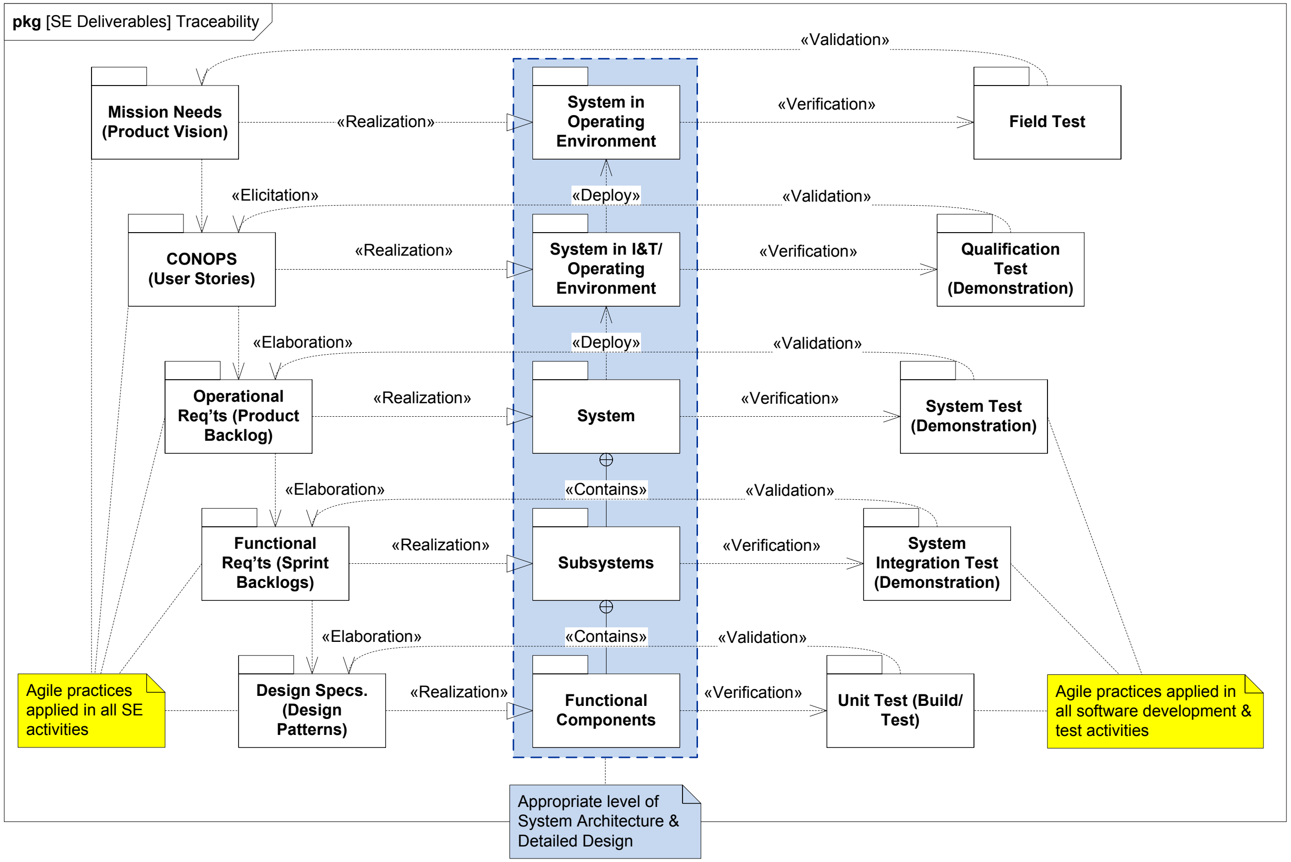

Process Implementation
Software Installation
Software Acceptance Support
Project
System Requirements Analysis
System Architecture Design
System Integration
System Qualification Testing
System
Software Requirements Analysis
Software Qualification Testing
Software Architectural Design
Software Integration
Software Detailed Design
Software Coding & Testing
Software

Reference: DoD-STD-2167A, *Defense System Software Development*, February 29, 1988

System/System Development Life Cycle (SDLC)

Everything must be traceable

- Verification: “The process of evaluating a system or component to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase.”
- Validation: “Confirmation, through the provision of objective evidence, that the requirements for a specific intended use or application have been fulfilled.”



pkg [SE Deliverables] Traceability

Mission Needs (Product Vision) «Realization» System in Operating Environment «Verification» Field Test

CONOPS (User Stories) «Realization» System in I&T/ Operating Environment «Verification» Qualification Test (Demonstration)

Operational Req’ts (Product Backlog) «Realization» System «Verification» System Test (Demonstration)

Functional Req’ts (Sprint Backlogs) «Realization» Subsystems «Verification» System Integration Test (Demonstration)

Design Specs. (Design Patterns) «Realization» Functional Components «Verification» Unit Test (Build/ Test)

«Validation» «Elicitation» «Elaboration» «Deploy» «Contains»

Agile practices applied in all SE activities

Agile practices applied in all software development & test activities

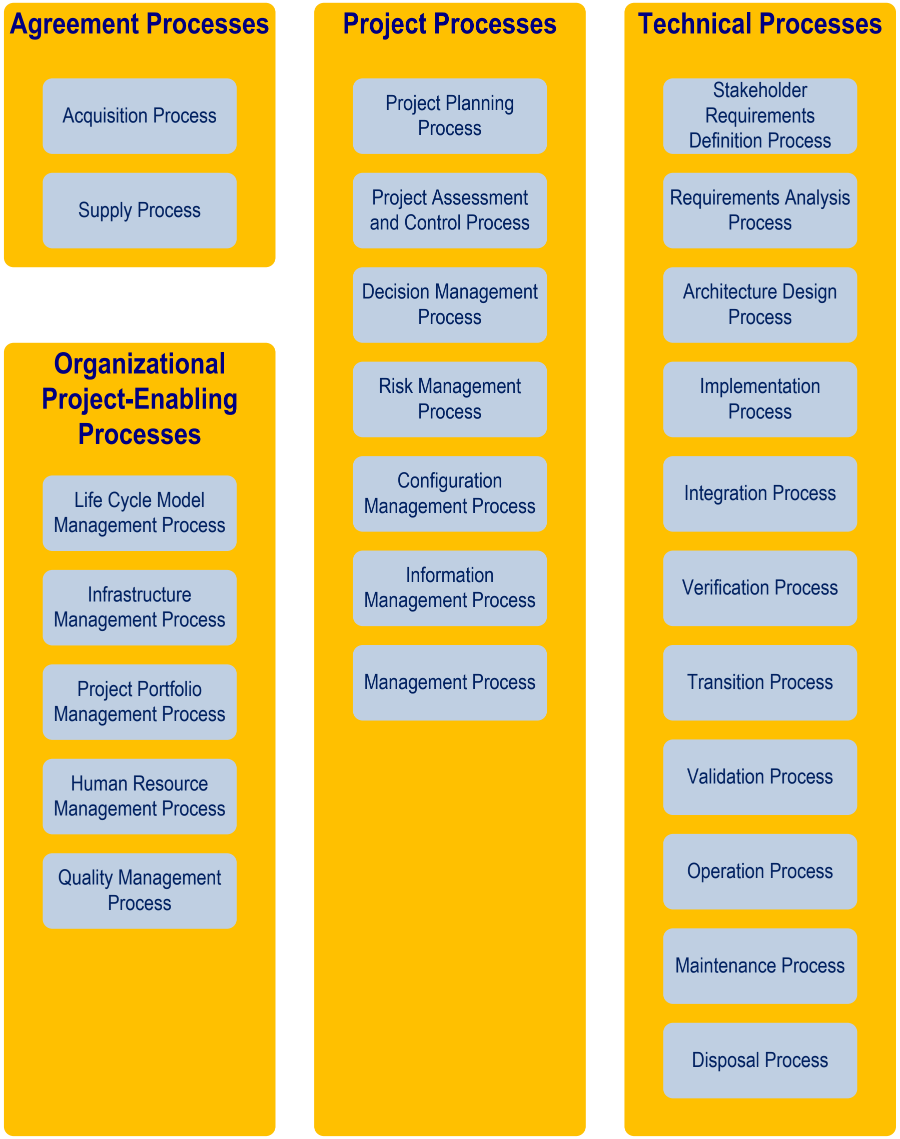
Appropriate level of System Architecture & Detailed Design

Reference: ISO/IEC/IEEE 24765:2010, *Systems and Software Engineering - Vocabulary*, 1st Ed. December 15, 2010.

System/Software Development Life Cycle (SDLC)

ISO/IEC 15288:2008, System Life Cycle Processes

- ISO/IEC 15288* encompasses:
  - Systems/software engineering processes (Technical Processes)
  - Project management processes
  - Project support infrastructure (Organizational Project-Enabling Processes)
  - Contract/business management processes (Agreement Processes)



Agreement Processes

- Acquisition Process
- Supply Process

Organizational Project-Enabling Processes

- Life Cycle Model Management Process
- Infrastructure Management Process
- Project Portfolio Management Process
- Human Resource Management Process
- Quality Management Process

Project Processes

- Project Planning Process
- Project Assessment and Control Process
- Decision Management Process
- Risk Management Process
- Configuration Management Process
- Information Management Process
- Management Process

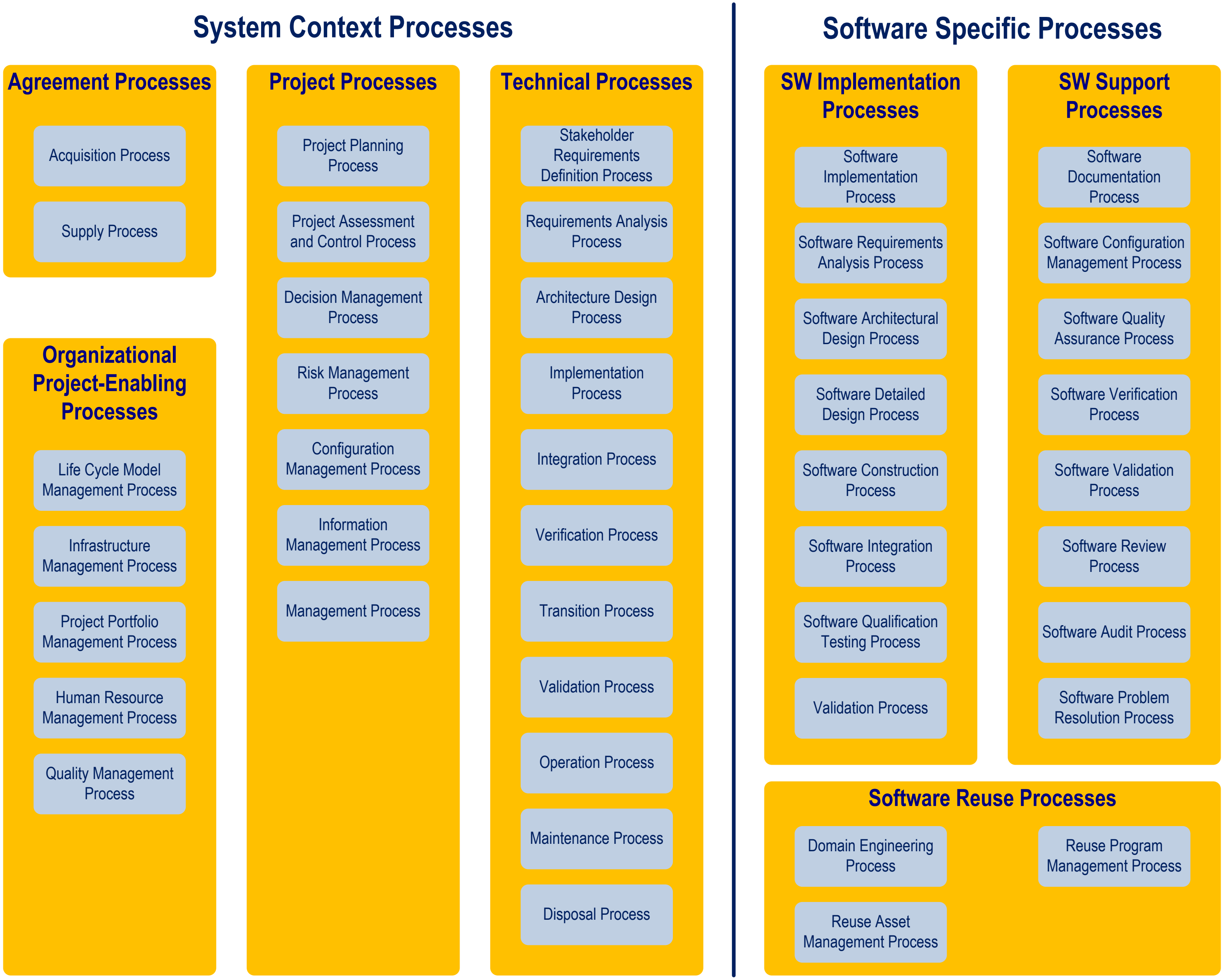
Technical Processes

- Stakeholder Requirements Definition Process
- Requirements Analysis Process
- Architecture Design Process
- Implementation Process
- Integration Process
- Verification Process
- Transition Process
- Validation Process
- Operation Process
- Maintenance Process
- Disposal Process

* Note: ISO/IEC 15288 is identical to IEEE Std 15288

System/Software Development Life Cycle (SDLC)

ISO/IEC 12207:2008, Software Life Cycle Processes



System Context Processes

Agreement Processes

- Acquisition Process
- Supply Process

Organizational Project-Enabling Processes

- Life Cycle Model Management Process
- Infrastructure Management Process
- Project Portfolio Management Process
- Human Resource Management Process
- Quality Management Process

Project Processes

- Project Planning Process
- Project Assessment and Control Process
- Decision Management Process
- Risk Management Process
- Configuration Management Process
- Information Management Process
- Management Process

Technical Processes

- Stakeholder Requirements Definition Process
- Requirements Analysis Process
- Architecture Design Process
- Implementation Process
- Integration Process
- Verification Process
- Transition Process
- Validation Process
- Operation Process
- Maintenance Process
- Disposal Process

Software Specific Processes

SW Implementation Processes

- Software Implementation Process
- Software Requirements Analysis Process
- Software Architectural Design Process
- Software Detailed Design Process
- Software Construction Process
- Software Integration Process
- Software Qualification Testing Process
- Validation Process

SW Support Processes

- Software Documentation Process
- Software Configuration Management Process
- Software Quality Assurance Process
- Software Verification Process
- Software Validation Process
- Software Review Process
- Software Audit Process
- Software Problem Resolution Process

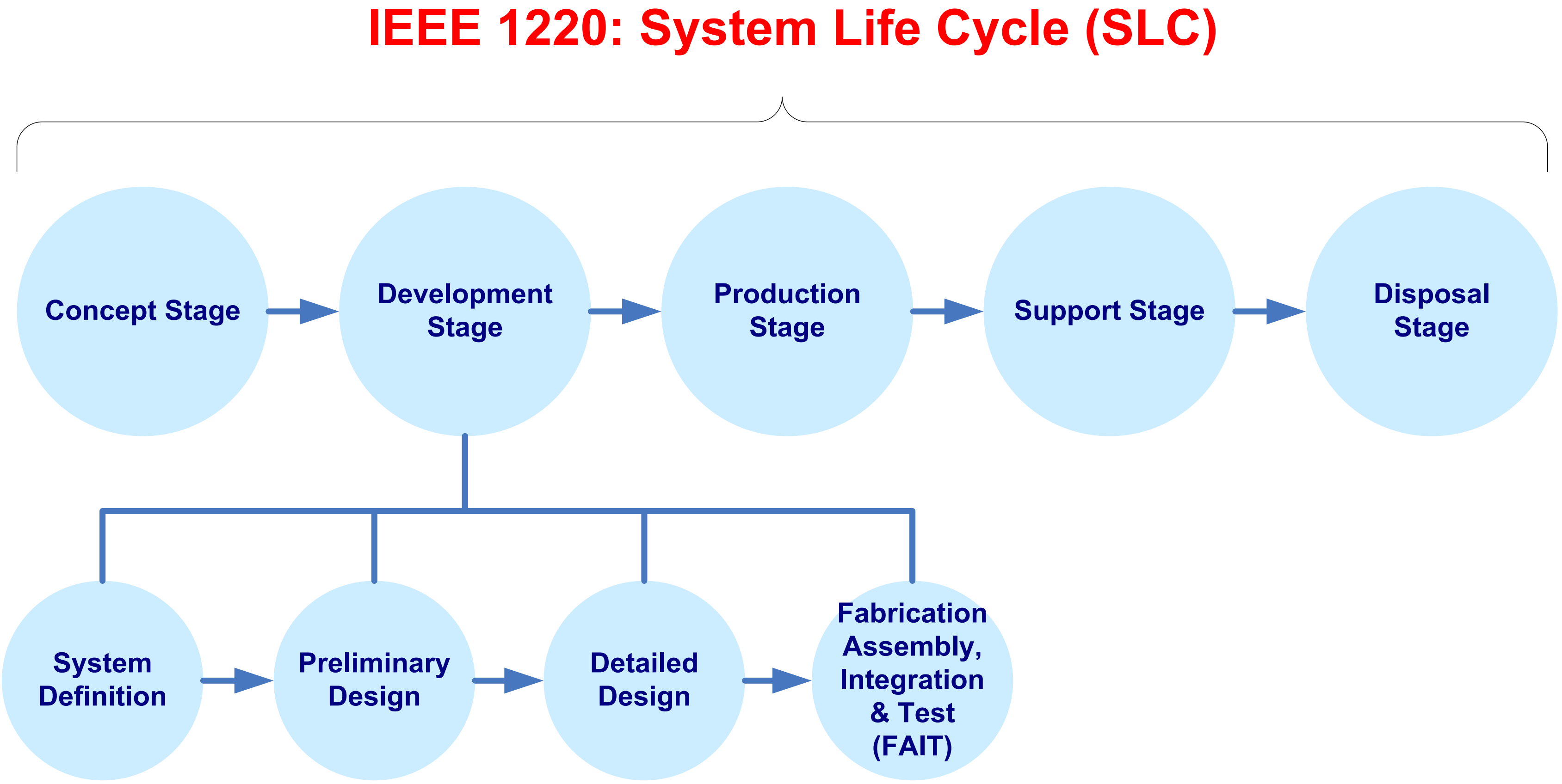
Software Reuse Processes

- Domain Engineering Process
- Reuse Program Management Process
- Reuse Asset Management Process

* Note: ISO/IEC 12207is identical to IEEE Std 12207

Reference: IEEE/IEC 12207:2008, *Information Technology Software Life Cycle Processes*

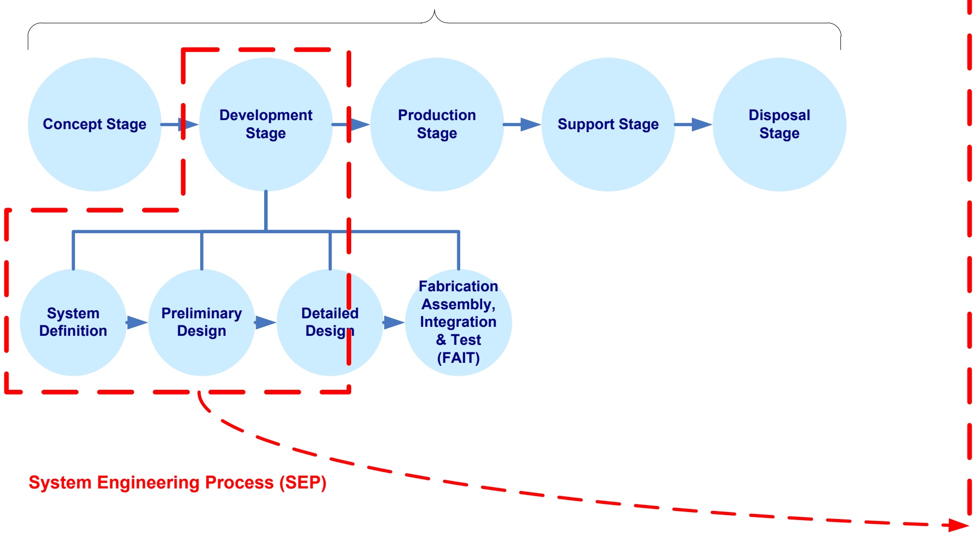
IEEE std 1220, System Engineering Process



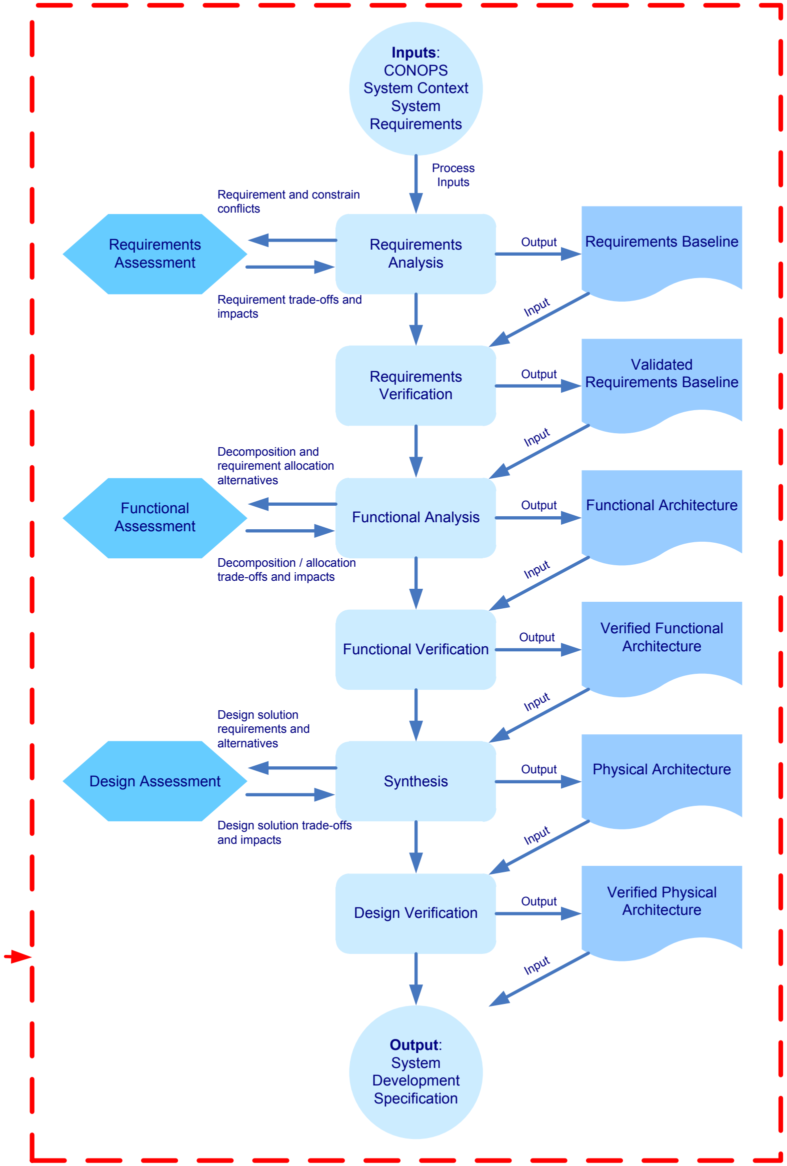
IEEE std 1220: System Engineering Process (SEP)

- IEEE 1220 defined System Engineering Process (SEP) within System Life Cycle (SLC)

IEEE 1220: System Life Cycle (SLC)



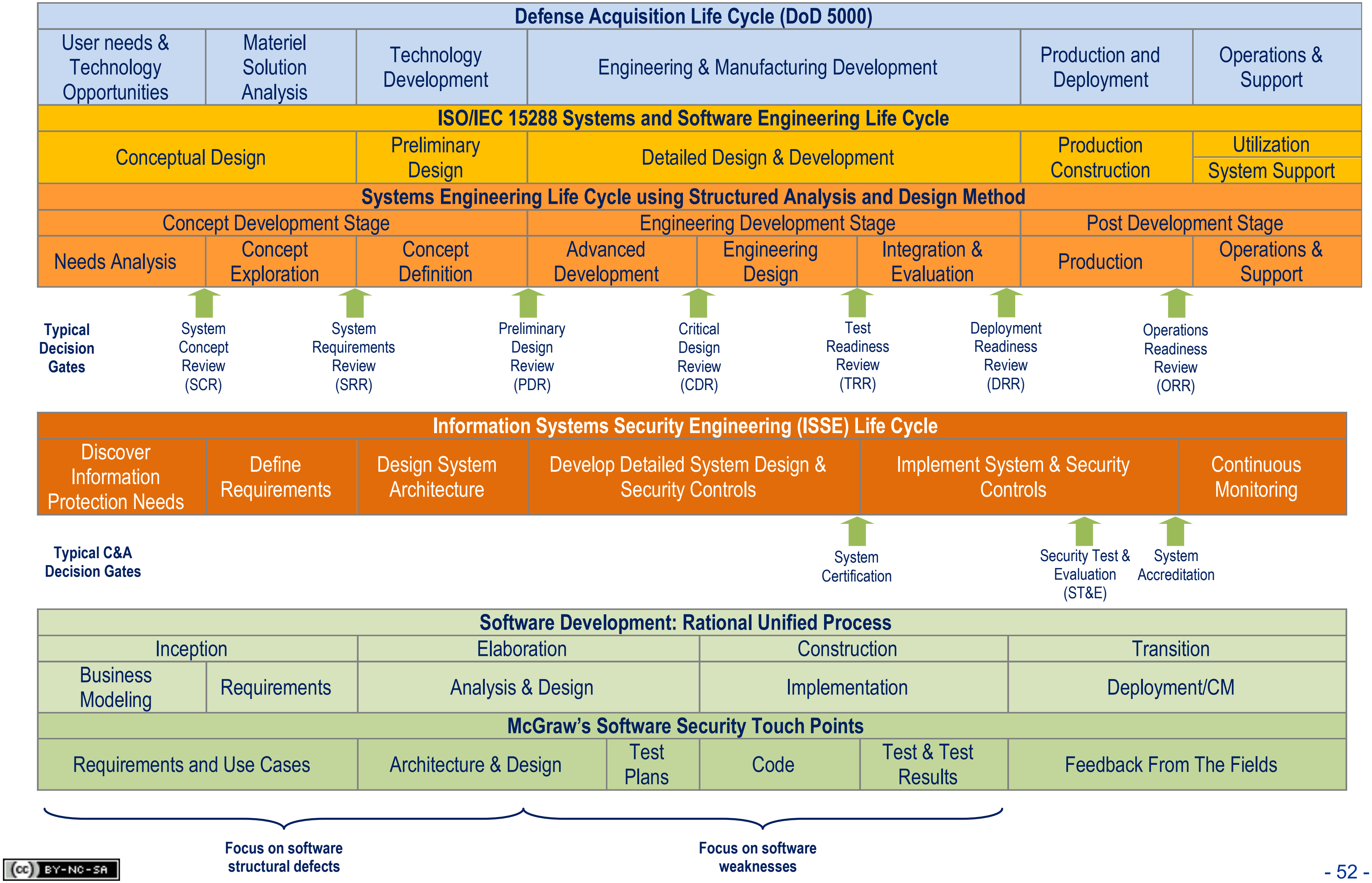
System Engineering Process (SEP)


Inputs: CONOPS System Context System Requirements
Process Inputs
Requirement and constrain conflicts
Requirements Assessment
Requirements Analysis
Requirement trade-offs and impacts
Output
Requirements Baseline
Input
Requirements Verification
Output
Validated Requirements Baseline
Input
Decomposition and requirement allocation alternatives
Functional Assessment
Functional Analysis
Decomposition / allocation trade-offs and impacts
Output
Functional Architecture
Input
Functional Verification
Output
Verified Functional Architecture
Input
Design solution requirements and alternatives
Design Assessment
Synthesis
Design solution trade-offs and impacts
Output
Physical Architecture
Input
Design Verification
Output
Verified Physical Architecture
Input
Output: System Development Specification

Reference: IEEE STD 1220: *Standard for Application and Management of the Systems Engineering Process*

System/Software Development Life Cycle (SDLC)

Introducing Security into SDLC



Defense Acquisition Life Cycle (DoD 5000)

User needs & Technology Opportunities	Materiel Solution Analysis	Technology Development	Engineering & Manufacturing Development	Production and Deployment	Operations & Support

ISO/IEC 15288 Systems and Software Engineering Life Cycle

Conceptual Design	Preliminary Design	Detailed Design & Development	Production Construction	Utilization System Support

Systems Engineering Life Cycle using Structured Analysis and Design Method

Concept Development Stage			Engineering Development Stage			Post Development Stage	
Needs Analysis	Concept Exploration	Concept Definition	Advanced Development	Engineering Design	Integration & Evaluation	Production	Operations & Support

Typical Decision Gates

- System Concept Review (SCR)
- System Requirements Review (SRR)
- Preliminary Design Review (PDR)
- Critical Design Review (CDR)
- Test Readiness Review (TRR)
- Deployment Readiness Review (DRR)
- Operations Readiness Review (ORR)

Information Systems Security Engineering (ISSE) Life Cycle

Discover Information Protection Needs	Define Requirements	Design System Architecture	Develop Detailed System Design & Security Controls	Implement System & Security Controls	Continuous Monitoring

Typical C&A Decision Gates

- System Certification
- Security Test & Evaluation (ST&E)
- System Accreditation

Software Development: Rational Unified Process

Inception		Elaboration	Construction	Transition
Business Modeling	Requirements	Analysis & Design	Implementation	Deployment/CM

McGraw's Software Security Touch Points

Requirements and Use Cases	Architecture & Design	Test Plans	Code	Test & Test Results	Feedback From The Fields

Focus on software structural defects

Focus on software weaknesses

Security Considerations in SDLC

1. Initiation Phase (IEEE 1220: Concept Stage)
   - Survey & understand the policies, standards, and guidelines.
   - Identify information assets (tangible & intangible).
   - Define information classification & protection level (security categorization).
   - Define rules of behavior & security CONOPs.
   - Conduct preliminary risk assessment.
2. Acquisition / Development Phase (IEEE 1220: Development Stage)
   - Conduct risk assessment.
   - Define security requirements and select security controls (categories & types).
   - Perform cost/benefit analysis (CBA).
   - Security planning (based on risks & CBA).
   - Practice Information Systems Security Engineering (ISSE) Process to develop security controls.
   - Develop security test & evaluation (ST&E) plan for verification & validation of security controls.

Reference: NIST SP 800-64 *Security Considerations in the Information System Development Life Cycle.*

Security Considerations in SDLC

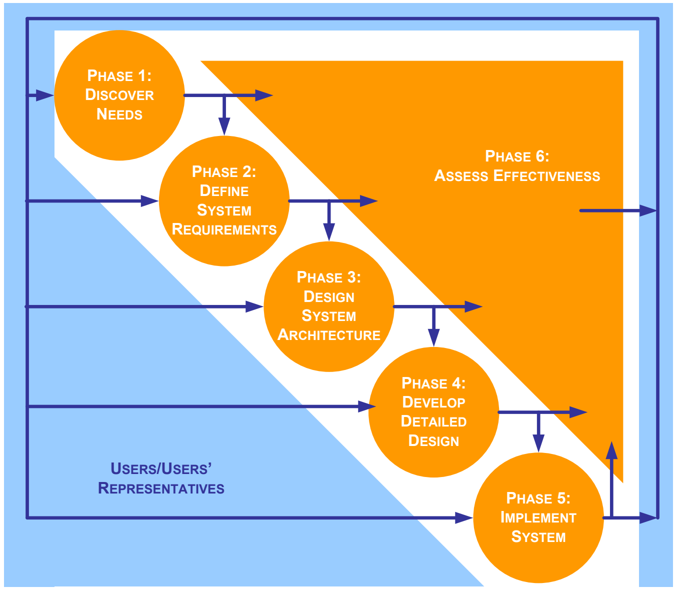
3. Implementation Phase (IEEE 1220: Production Stage)
   - Implement security controls in accordance with system security plan (SSP).
   - Perform Security Certification & Accreditation of target system.
4. Operations / Maintenance Phase (IEEE 1220: Support Stage)
   - Configuration management & perform change control.
   - Continuous monitoring – Perform periodic security assessment.
5. Disposition Phase (IEEE 1220: Disposal Stage)
   - Preserve information. archive and store electronic information
   - Sanitize media. Ensure the electronic data stored in the disposed media are deleted, erased, and over-written
   - Dispose hardware. Ensure all electronic data resident in hardware are deleted, erased, and over-written (i.e. EPROM, BIOS, etc.)

Reference: NIST SP 800-64 *Security Considerations in the Information System Development Life Cycle*.

System/Software Development Life Cycle (SDLC)

Information Systems Security Engineering (ISSE) Process

- Phase 1: Discover Information Protection Needs
  - Ascertain the system purpose.
  - Identify information asset needs protection.
- Phase 2: Define System Security Requirements
  - Define requirements based on the protection needs.
- Phase 3: Design System Security Architecture
  - Design system architecture to meet on security requirements.
- Phase 4: Develop Detailed Security Design
  - Based on security architecture, design security functions and features for the system.
- Phase 5: Implement System Security
  - Implement designed security functions and features into the system.
- Phase 6: Assess Security Effectiveness
  - Assess effectiveness of ISSE activities.

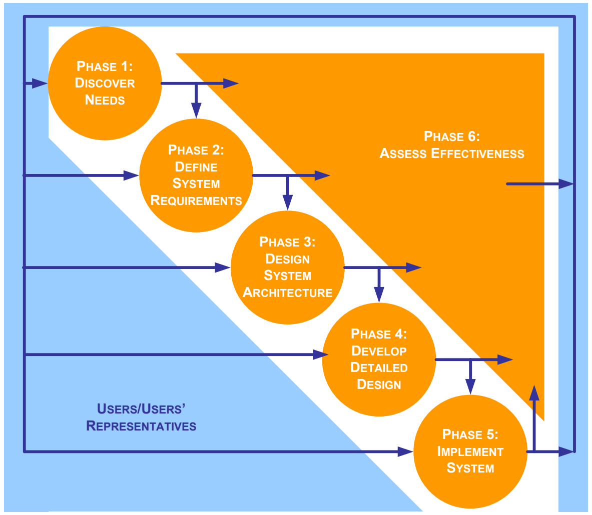

PHASE 1: DISCOVER NEEDS
PHASE 2: DEFINE SYSTEM REQUIREMENTS
PHASE 3: DESIGN SYSTEM ARCHITECTURE
PHASE 4: DEVELOP DETAILED DESIGN
PHASE 5: IMPLEMENT SYSTEM
PHASE 6: ASSESS EFFECTIVENESS
USERS/USERS' REPRESENTATIVES

Reference: *Information Assurance Technical Framework* (IATF) Rel. 3.1

System/Software Development Life Cycle (SDLC)

Security starts at the beginning…

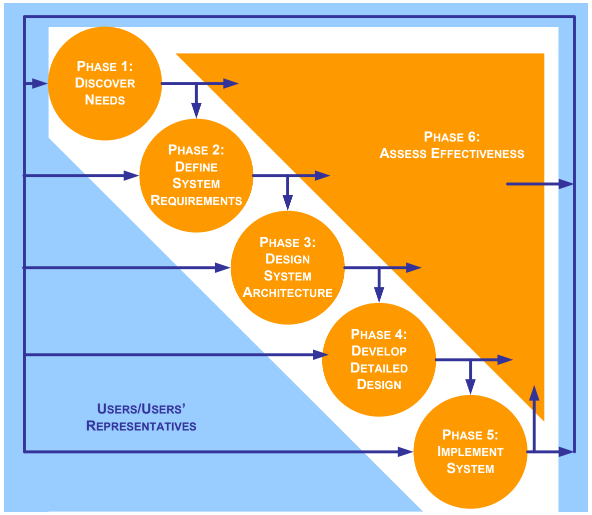
IEEE 1220	DoD Acquisition SDLC	Key System Engineering Tasks	Key Security Engineering Tasks*
Concept Stage	User Needs & Technology Opportunities	**Task 1: Discover Mission/Business Needs**	**Task 1: Discover Information Protection Needs**
		• Understand customer's mission/business goals (i.e., initial capability, project risk assessment)	• Understand customer's information protection needs (i.e., infosec. risk assessment)
	Concept Refinement	• Understand system concept of operations (CONOPS)	• Understand operating environment (i.e., sensitivity of information assets, mode of operations)
		• Create high-level entity-data relations model (i.e., system context diagram)	• Create information management model (IMM)
		• Define engineering project strategy and integrate into the overall project strategy	• Define information protection policy (IPP) and integrate into the project strategy
		• Create system engineering management plan (SEMP)	• Create system security plan (SSP) and integrate into SEMP
	Milestone A	**Task 6: Assess project performance in meeting mission/business needs**	


Phase 1: Discover Needs
Phase 2: Define System Requirements
Phase 3: Design System Architecture
Phase 4: Develop Detailed Design
Phase 5: Implement System
Phase 6: Assess Effectiveness
Users/Users' Representatives

* **Reference**: *Information Assurance Technical Framework* (IATF), Release 3.1

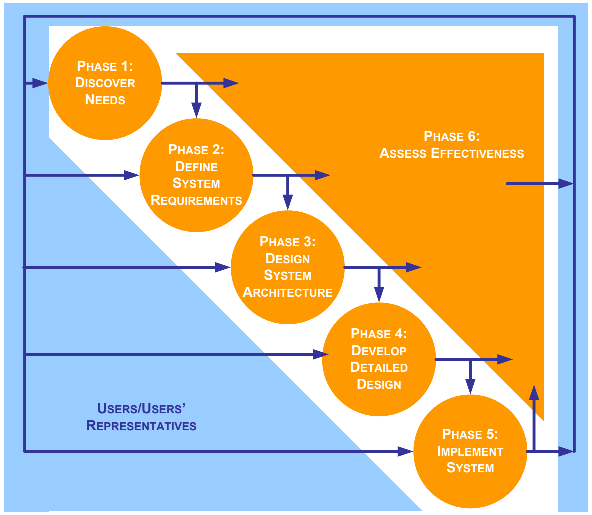
- Key Deliverables
  - Mission Needs Statement / Project Goal(s) and Objectives
  - System Capabilities
  - Preliminary CONOPS
  - Preliminary System Context Descriptions
  - Project Risk Assessment
  - Draft System Engineering Management Plan (SEMP)

IEEE 1220	DoD Acquisition SDLC	Key System Engineering Tasks	Key Security Engineering Tasks
Development Stage	Technology Development	**Task 2: Define System Requirements**	**Task 2: Define Security Requirements**
		• Refine system context (e.g., functional components)	
		• Define system requirements (e.g., functional, performance, operational, support, etc.)	• Select assurance requirements and define security functional requirements
		• Refine CONOPS	• Refine IMM and SSP
		• Baseline system requirements	
	Milestone B	**Task 6: Assess project performance in meeting mission/business needs**	
	System Development & Demonstration	**Task 3: Design System Architecture**	**Task 3: Design System Security Architecture**
		• Determine & select architecture framework	
		• Design system architecture and allocate system requirements to subsystems and components (i.e., RTM)	• Allocate system security requirements to subsystems and service components (i.e., RTM)
		• Analyze gaps (i.e., risk assessment)	
		Task 4: Develop Detailed System Design (Logical & Physical)	**Task 4: Develop Detailed System Security Design (Logical & Physical)**
		• Refine entity-data relations model (i.e., UML diagrams, data-flow, network, etc.)	• Refine IMM, embed security controls into system design products (i.e., UML, data-flow, network, etc.)
		• Perform system synthesis analysis to assure system integration (i.e., system design, system architecture, system requirements, and project mission/business needs)	
	Milestone C	**Task 6: Assess project performance in meeting mission/business needs**	


Phase 1: Discover Needs
Phase 2: Define System Requirements
Phase 3: Design System Architecture
Phase 4: Develop Detailed Design
Phase 5: Implement System
Phase 6: Assess Effectiveness
Users/Users' Representatives

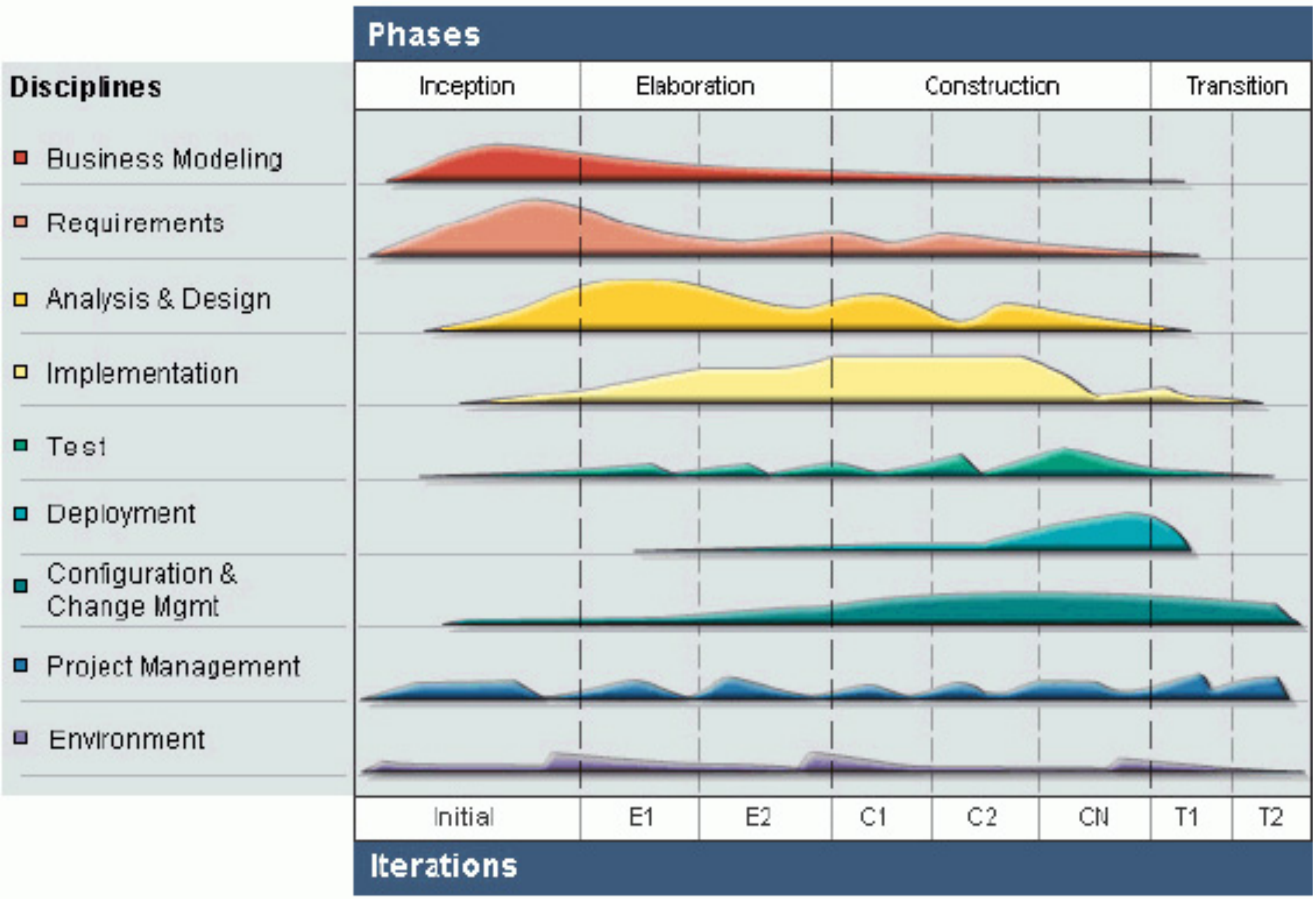
- Key Deliverables
  - System Requirements
  - Functional Definitions (+ allocation of system requirements)
  - System Architecture (Contextual + Logical)
  - Detailed System Design (Logical + Physical)
  - Requirements Traceability Matrix (RTM)

IEEE 1220	DoD Acquisition SDLC	Key System Engineering Tasks	Key Security Engineering Tasks
Production Stage	Production and Deployment	**Task 5: Implement System Design**	**Task 5: Implement Security Controls**
		• Procure system components / construct system	
		• Code/ customize/ configure system functional components	
		• Conduct code inspection/ walk-through/ unit test	
		• Perform system integration	
		• Conduct system test	• Conduct security test & evaluation (ST&E)
		Task 6: Assess project performance in meeting mission/business needs	
		• Generate system operations procedure (SOP) and users guide/ manual	• Generate SOP (a.k.a. trusted facility manual (TFM)), Incident response plan, business continuity plan (BCP)
		• Conduct system readiness review	• Obtain system certification
		• Deploy system	
		• Conduct system acceptance test	• Assess security effectiveness
		• Obtain approval to operate (ATO)	


Phase 1: Discover Needs
Phase 2: Define System Requirements
Phase 3: Design System Architecture
Phase 4: Develop Detailed Design
Phase 5: Implement System
Phase 6: Assess Effectiveness
Users/Users' Representatives

- Key Deliverables
  - Implement detailed system design
  - Perform test & evaluations (unit, system, security tests)
  - Test reports
  - Standard Operating Procedure (SOP) + User Manuals
  - Deploy system
  - Conduct acceptance tests

Rational Unified Process (RUP)



Reference: http://www.ibm.com/developerworks/webservices/library/ws-soa-term2/

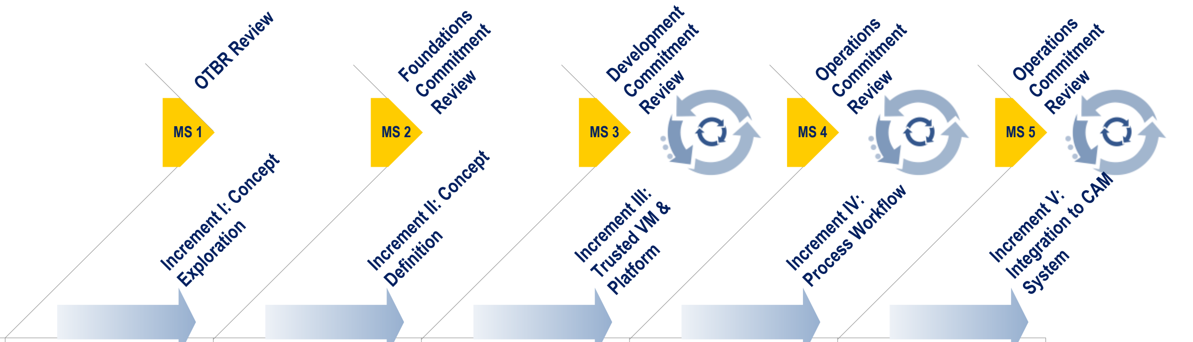
Rational Unified Process (RUP)

Software Development: Rational Unified Process						
Inception		Elaboration		Construction		Transition
Business Modeling	Requirements	Analysis & Design		Implementation		Deployment/CM
McGraw's Software Security Touch Points						
Requirements and Use Cases		Architecture & Design	Test Plans	Code	Test & Test Results	Feedback From The Fields

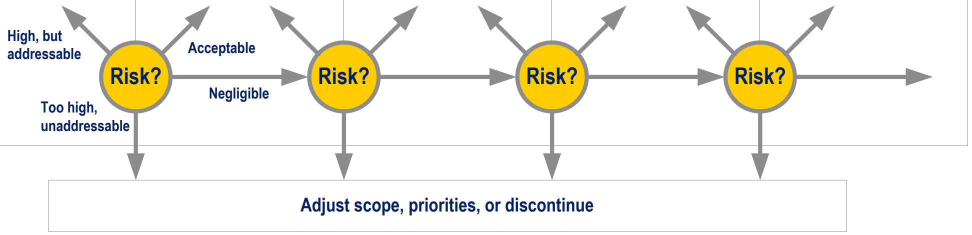
- Use cases drives requirements (Business Needs/Concept Exploration)
  - System, software, and security engineers create operational use cases (e.g., operational, functions, threat, risks models)
  - Use cases drives operational requirements
- System design drives design specifications (Concept Definition/Detailed Design)
  - Operational requirements are decomposed into system functions and functional requirements
  - Architecture organizes system functions allocation of functional requirements
  - Architecture is further decomposed into detailed system design
  - Detailed system design is explained in design specifications
- Design specifications drives programming of software codes (Implementation/Coding/Integration/Testing)
  - Software components integrated into functional components/subsystems (Unit Testing)
  - Functional subsystems integrated into system (/systems) (System Testing)
  - System perform functions that meets the operational needs (Acceptance Testing)
- Deployment/transition into operations

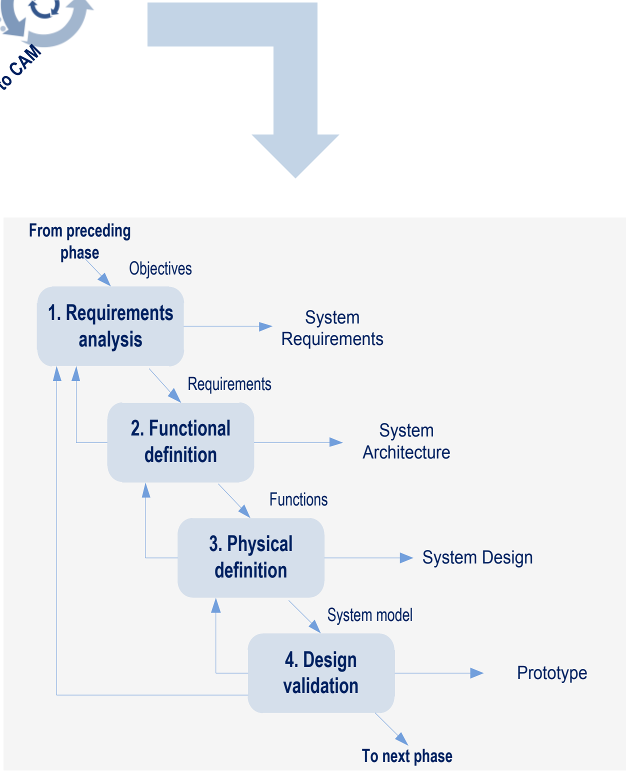
System/Software Development Life Cycle (SDLC)

Integrated System/Security Engineering in RAD


OTBR Review
MS 1
Foundations Commitment Review
MS 2
Development Commitment Review
MS 3
Operations Commitment Review
MS 4
Operations Commitment Review
MS 5

Major Activities	Increment I: Concept Exploration	Increment II: Concept Definition	Increment III: Trusted VM & Platform	Increment IV: Process Workflow	Increment V: Integration to CAM System
Concurrent risk and opportunity-driven growth of system understanding and definition	· Initial scoping · Concept definition · Investment analysis	· System life cycle architecture and CONOPS · Build to increment plans and specifications	· Develop Increment III prototype · Exercise Increment III prototype · Rebaseline system features & capabilities	· Develop Increment IV prototype · Exercise Increment IV prototype · Rebaseline system features & capabilities (if necessary)	· Develop Increment V prototype · Exercise Increment V prototype · Transition into operations · Plan for future release (if necessary)
Evaluation of evidence of feasibility to proceed	· OTBR Package · Draft PIP	· Updated PIP · CONOPS · Conceptual Architecture · System Reqs. & Functional Specs.	· Prototype · Updated PIP · Updated CONOPS · System Architecture · Updated System Reqs. & Functional Specs.	· Prototype · Updated PIP · System Design · Updated System Reqs. & Functional Specs.	· Prototype · Operational Transition Plan · Updated System Design Baseline · User features requests
Stakeholder review & commitment					





Questions:

- What are the relationships between SDLC models and SSE-CMM models?
  - SDLC describes… to a system acquisition project
  - SSE-CMM describes…
- What are the relationships between security controls models (NIST SP800-53, DoDI 8500.2, ISO/IEC 27001, etc.) and CMM/SSE-CMM models?
  - Security assurance requirements provide measurement of…
  - CMM utilizes the measurement metrics from security control models to measure…

Answers:

- What are the relationships between SDLC models and SSE-CMM models?
  - SDLC describes the key engineering process to a system acquisition project
  - SSE-CMM describes the key security and management processes to a security engineering practice

- What are the relationships between security controls models (NIST SP800-53, DoDI 8500.2, ISO/IEC 27001, etc.) and CMM/SSE-CMM models?
  - Security assurance requirements provide measurements of management, operational, and technical controls
  - CMM utilizes the measurement metrics from security control models to measure practice maturity

Topics

Software Development Security Domain

- Governance & Management
- System Life Cycle and Security
- ➡ Software Environment and Security Controls
- Programming Languages
- Database and DB Warehousing Vulnerabilities, Threats, and Protections
- Software Vulnerabilities and Threats

Review of Computer Operations Architecture Model

- Reference monitor is a conceptual abstraction of a "machine", system, or software that mediates access of objects by subjects.
- Trusted computing base is a system of security controls that meets the confidentiality and integrity security objectives.
- Secure kernel is a part of the trusted computing base that implements reference monitor concept.

Reference: *Secrets & Lies – Digital Security in a Networked World*, Bruce Schneier, Wiley Publishing, 2000

Reference Monitor

- Reference monitor is performed by a reference validation mechanism.
- Reference validation mechanism is a system composed of hardware and software.
- Operating condition principles:
  - The reference validation mechanism must be tamper proof.
  - The reference validation mechanism must always be invoked.
  - The reference validation mechanism must be small enough to be subject to analysis and tests to assure that it is correct.
- OS shall be evaluated at TCSEC B2 (i.e. structured protection) and above.

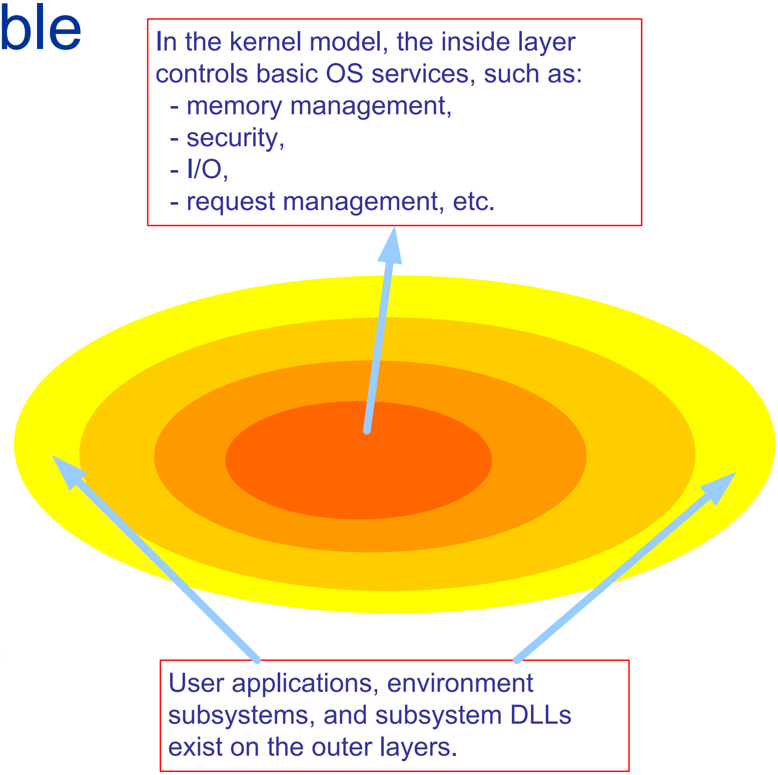
Trusted Computing Base (TCB)

- The Trusted Computing Base is the totality of protection mechanisms within a computing system – hardware, firmware, software, processes, transports
- The TCB maintains the confidentiality and integrity of each domain and monitors four basic functions:
  - Process activation
  - Execution domain switching
  - Memory protection
  - Input/output operation

Reference: DoD 5200.28-STD, *Department of Defense Trusted Computer System Evaluation Criteria* (TCSEC), August 15, 1983

Secure Kernel

- Secure kernel is an implementation of a reference monitoring mechanism responsible for enforcing security policy.
- It meets the following three (3) conditions:
  - Completeness. All accesses to information must go through the kernel.
  - Isolation. The kernel itself must be protected from any type of unauthorized access.
  - Verifiability. The kernel must be proven to meet design specifications.



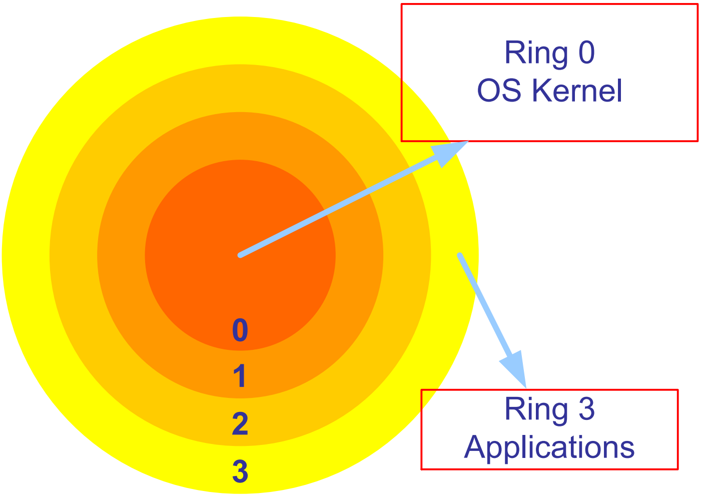
In the kernel model, the inside layer controls basic OS services, such as:
- memory management,
- security,
- I/O,
- request management, etc.

User applications, environment subsystems, and subsystem DLLs exist on the outer layers.

Processor Privilege States

- Processor privilege states protect the processor and the activities that it performs.
- Privileged levels are called rings.

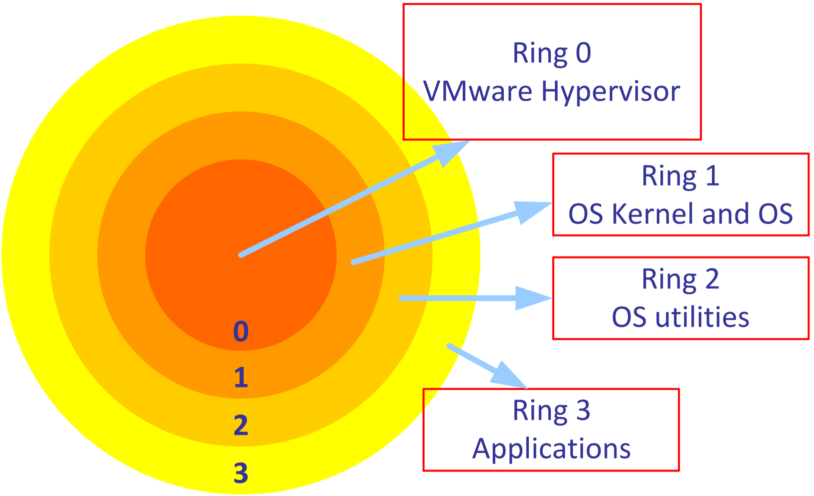
- For example: Intel x86 has 4 privilege ring levels
  - Ring 0 contains kernel functions of the OS.
  - Ring 1 contains the OS.
  - Ring 2 contains the OS utilities.
  - Ring 3 contains the applications.

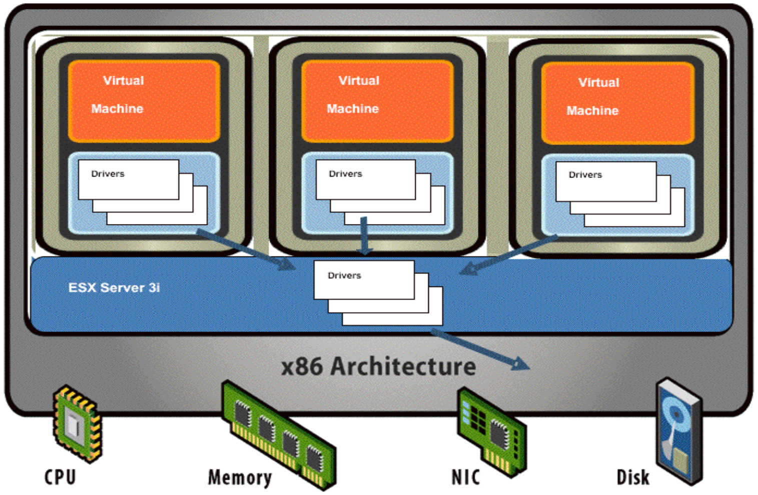


Example of Processor Privilege States

VMware ESX

- Hypervisor operates at Ring 0
- Guest OS kernel and OS now moved to Ring 1
- OS utilities in Ring 2
- Application in Ring 3

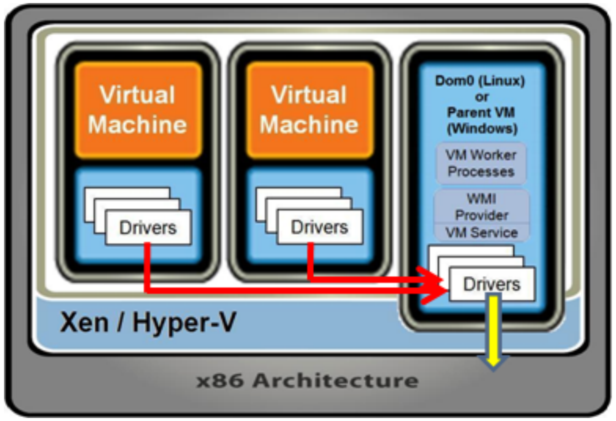




Reference: *VMware ESX I/O Driver Model* (http://blogs.vmware.com/performance/2007/11/ten-reasons-why.html)

Same principles, but different technology thus different attacks

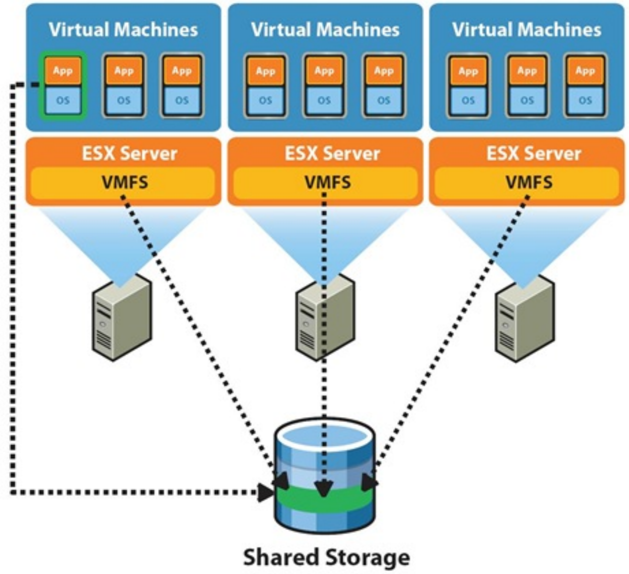
- Reference monitoring principles is consistent even with virtualization: Violation of privilege
  - Hypervisor vulnerabilities. Attack of kernel (Ring 0)
  - Hypervisor escape vulnerabilities. Violation of isolation of guest VMs (Ring 0)
  - Administrative VM vulnerabilities
    - Management server vulnerabilities. Exploitation of virtualized system configuration. (Ring 0)
    - Management console vulnerabilities. Attacks of privileged state (Entire TCB)
  - Guest VM vulnerabilities. Exploitation of OS vulnerabilities, but can potentially provide an attack vector to administrative VM, hypervisor, then other guest VMs (Ring 3/Ring 2 → Ring 1 → Ring 0)

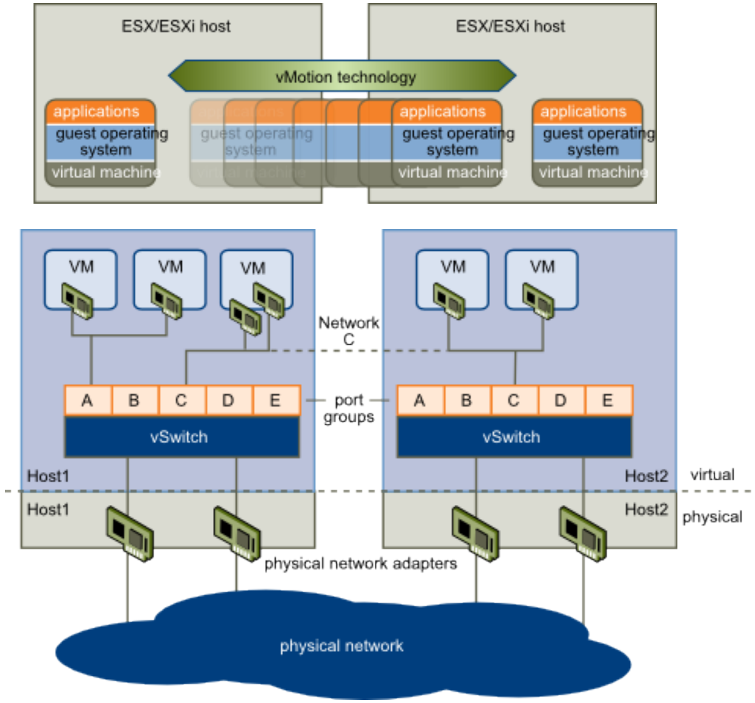


Reference: *Virtual Reality* (http://blogs.vmware.com/virtualreality/2008/06/)

Example of Processor Privilege States – Many-to-Many

- VMware vSphere further abstracts the hardware layer
  - Virtual Machine File System (VMFS) for abstraction of data storage
  - vNetwork Distributed Switch (vDS) for abstraction of network layer
  - vMotion for distribution of processing power and high availability





Reference: http://pubs.vmware.com/vsphere-4-esx-vcenter/index.jsp?topic=/com.vmware.vsphere.intro.doc_41/c_vmware_infrastructure_introduction.html

More complexity, more attack surfaces - Examples

- Hypervisor vulnerability:
  - CVE-2010-2070: Xen IA-64 architecture, allows local user to modify processor status register that can cause DoS. (CVSS: 4.9 [Medium])
- Hypervisor escape vulnerability:
  - CVE-2009-1244: VM display function in VMware allows guest OS user to execute arbitrary code in hypervisor. (CVSS: 6.8 [Medium])
- Administrative VM vulnerabilities:
  - CVE-2008-2097: Buffer overflow in VMware ESX management service that allows remote authenticated users to gain root privileges. (CVSS: 9.0 [High])
  - CVE-2008-4281: Directory traversal in VMware ESXi that allows VM administrators to gain elevated privileges. (CVSS: 9.3 [High])
  - CVE-2009-2277: Cross-site scripting (XSS) vulnerability in WebAccess in VMware VirtualCenter that allows remote attacker to inject arbitrary web script to steal "context data" such as authentication credentials (CVSS: 4.3 [Medium])
- Guest VM vulnerabilities:
  - CVE-2011-2145: VMware Host Guest File System (HGFS) allows Solaris or FreeBSD guest OS users to modify guest OS files. (CVSS: 6.3 [Medium])
  - CVE-2011-2217: ActiveX controls in Internet Explorer allows remote attacker to execute arbitrary code or corrupt memory in VMware Infrastructure. (CVSS: 9.3 [High])

Reference:

- T. McNevin, *Introduction to Hypervisor Vulnerabilities (Part 1),* MITRE, 2009
- B. Williams, T. Cross, *Virtualization System Vulnerabilities*, IBM X-Force, 2010
- NVD (http://web.nvd.nist.gov/view/vuln/search)

Security Controls for Software Environment

- For CISSP Exam, countermeasures are also called “security controls”…
  - Security Controls for Buffer Overflows
  - Memory Protection
  - Covert Channel Controls
  - Cryptography
  - Password Protection Techniques
  - Inadequate Granularity of Controls
  - Control and Separation of Environments
  - Time of Check/Time of Use (TOC/TOU)
  - Social Engineering
  - Backup Controls
  - Malicious Code/Malware Controls
  - Virus Protection Controls
  - Mobile Code Controls
  - Sandbox
  - Programming Language Support
  - Access Controls

Reference: *Official (ISC)^2 Guide to The CISSP CBK*, H. Tipton, et. al., (ISC)^2 Press, Auerbach Publications, 2007.

Security Controls for Buffer Overflow

- One of the oldest and most common problems to software.
- A buffer overflow occurs when a program or process tries to store more data in a buffer (temporary data storage area) than it was intended to hold.
- Vulnerability is caused by lack of parameter checking or enforcement for accuracy and consistency by the software application or OS.
- Countermeasure:
  - Practice good SDLC process (code inspection & walkthrough).
  - Programmer implementing parameter checks and enforce data rules.
  - Apply patches for OS & applications.
  - If available, implement hardware states and controls for memory protection.
  - Buffer management for OS.

Memory Protection

- Memory protection is enforcement of access control and privilege level to prevent unauthorized access to OS memory.
- Countermeasures:
  - Ensure all system-wide data structures and memory pools used by kernel-mode system components can only be accessed while in kernel mode.
  - Separate software processes, protect private address space from other processes.
  - Hardware-controlled memory protection
  - Use Access Control List (ACL) to protect shared memory objects.

Covert Channel Controls*

- Covert channel is an un-controlled information flow (or unauthorized information transfer) through hidden communication path(s).
  - Storage channel
  - Timing channel
- Countermeasure steps:
  - Identify potential covert channel(s)
  - Verify and validate existence of covert channel(s)
  - Close the covert channel by install patch or packet-filtering security mechanism.

* **Note**: While the definition of covert channel may be old, it is considered as “fundamental” in CISSP CBK.

Reference: NCSC-TG-30, *A Guide To Understanding Covert Channel Analysis of Trusted System*

Covert Channel Controls*

- Countermeasure for covert channel:
  - Information Flow Model is a variation of access control matrix
  - Information Flow Model is based on Object Security Levels.
  - Object-to-object information flow is constrained in accordance with object's security attributes.

Object	A	B	C	D	E	F	G
A	N/A			X			
B		N/A				X	
C	X		N/A				
D				N/A	X		
E		X			N/A		
F						N/A	X
G			X				N/A

Cryptography

- Cryptography provides confidentiality, integrity, authentication, and non-repudiation in information operations.
  - Asymmetric Key Cryptography
    - Because of slow cipher operation speed, it is mostly used for key management function.
  - Symmetric Key Cryptography
    - Because of speed, symmetric-key cryptosystems are used for crypto. operations. E.g. SSL/TLS at Transport-level (communication path), e-mail & SOAP messages at message-level.
  - Hash Function
    - Message Digest
    - Message Authentication Code (MAC)
    - Key-hashed MAC (HMAC)
  - Digital Signature

Security Controls: Password Protection Techniques

- Password Structure
  - Password length
  - Password complexity: a mix of upper/lowercase letters, numbers, special characters
  - Not using common words found in dictionary
- Password Maintenance

  Set password lifetime limits & policy…
  - Password change in <90> days
  - Password can not be reused within <10> password changes
  - <One> change to <every 24 hr.>
  - Password file must be encrypted and access controlled.

Granularity of Controls

- Separation of duties means that a process is designed so that separate steps must be performed by different people (i.e. force collusion)
  - Define elements of a process or work function.
  - Divide elements among different functions
- Least privilege is a policy that limits both the system's user and processes to access only those resources necessary to perform assigned functions.
  - Limit users and system processes to access only resources necessary to perform assigned functions.
- Separation of system environments.
  - Development environment.
  - QA/test environment.
  - Production or operational environment.

Other Security Controls

- Social Engineering
  - Countermeasure: User security awareness training.
- Backup, Malicious Code/Malware, Virus Protection Controls
  - Countermeasures:
    - Install & use anti-virus system, H-IDS.
    - Enable access control to critical system files.
    - Tape backups, access control of media.
    - Encrypt sensitive information for confidentiality & integrity.
- Mobile Code Controls
  - Install Sandbox for access control of mobile codes.
  - Example: Java "containers" or Java Virtual Machine (JVM).
    - Java applets running in Web browser.
    - Applications using Java Remote Method Invocation (RMI) to run Java Beans.

Security Controls – Access Controls

- Discretionary access control (DAC)
  - Information owner determines who has access & what privileges they have.
- Mandatory access control (MAC)
  - Information classification & system determine access.
  - Access decision based on privilege (clearance) of subject & sensitivity (classification) of object (file).
  - Requires labeling (or data tag)
- Access Control/Capability Matrix
  - Implement through the use of ACL.
- View-based Access Control
  - Authorization of specific views by tables, columns, and key sets.

Questions:

- What are the three operating condition principles for a reference monitor?
  - –
  - –
  - –

- What are the three operating conditions for a secure kernel?
  - –
  - –
  - –

Answers:

- What are the three operating condition principles for a reference monitor?
  - must be tamper proof
  - must always be invoked
  - subject to analysis and tests

- What are the three operating conditions for a secure kernel?
  - Completeness (must always be invoked)
  - Isolation (must be tamper proof)
  - Verifiability (each operations shall be subject to analysis and tests)

Questions:

- What causes buffer overflow?
  -

- Why a good information flow model is a good tool for supporting the identification of covert channel?
  -

Answers:

- What causes buffer overflow?
  - When a program or process that lacks parameter enforcement control tries to store more data in a buffer than it was intended to hold

- Why a good information flow model is a good tool for supporting the identification of covert channel?
  - Information flow model is the system design baseline that illustrates the directional vectors of information flow between objects (e.g., programs or processes)

Questions:

- Program that allows the information owner to determine who has what type of access and privilege is an implementation of what type of access control?
  -

- For mandatory access control (MAC), an access decision is based on privilege of ___ & sensitivity of ___?
  -
  -

Answers:

- Program that allows the information owner to determine who has what type of access and privilege is an implementation of what type of access control?
  - Discretionary access control (DAC)

- For mandatory access control (MAC), an access decision is based on privilege of ___ & sensitivity of ___?
  - Subject
  - Object

Topics

Software Development Security Domain

- Governance & Management
- System Life Cycle and Security
- Software Environment and Security Controls

➡ Programming Languages

- Database and DB Warehousing Vulnerabilities, Threats, and Protections
- Software Vulnerabilities and Threats

Programming Languages

- A set of instructions and rules that tell the computer what operations to perform.
- Languages have evolved in “generations”
  - 1st Generation: Machine language
  - 2nd Generation: Assembly language
  - 3rd Generation: High-level language
    - Ada, COBOL, BASIC, FORTRAN, Pascal, C, C+, C++, C#, Java
  - 4th Generation: Very high-level language
    - SQL, JavaScript, Perl, SGML (Standard General Markup Language): HTML, XML, SAML, XACML.
  - 5th Generation: Natural language
    - BPEL (Business Process Execution Language), BQEL (Business Query Language)

Programming Languages

- Assembler – program that translates an assembly language program into machine language.
  - Assembly Language → Machine Language.

- Compiler – translates a high-level language into machine language.
  - High-level Language (3rd Gen.) → Machine Language.

- Interpreter – instead of compiling a program at once, the interpreter translates it instruction-by-instruction. It has a fetch and execute cycle.
  - Very high-level Language (4th Gen.) → Interpreter instruction → Machine Language.

Object-Oriented Programming (OOP)

- OOP method that creates an object.
  - The object is a block of pre-assembled code that is a self-contained module.
  - Once written, object can be reused.
  - Objects are encapsulated, thus providing some security.
  - Objects have methods (code with programming interfaces) and attributes (data) encapsulated together.

Object-Oriented Programming (OOP) – Characteristics

- Object is an instance of the class.
- Class tell the system how to make objects.
- Encapsulation is the technique of keeping together data structures and methods (procedures) which act on them.
- Method is a procedure or routine associated with one or more classes.
- Message: objects perform work by sending messages to other objects.
- Inheritance is the ability to derive new classes from existing classes. A derived class (or subclass) inherits the instance variables and methods of the “base-class” (or superclass), and may add new instance variables and methods.

Object-Oriented Programming (OOP) – Characteristics

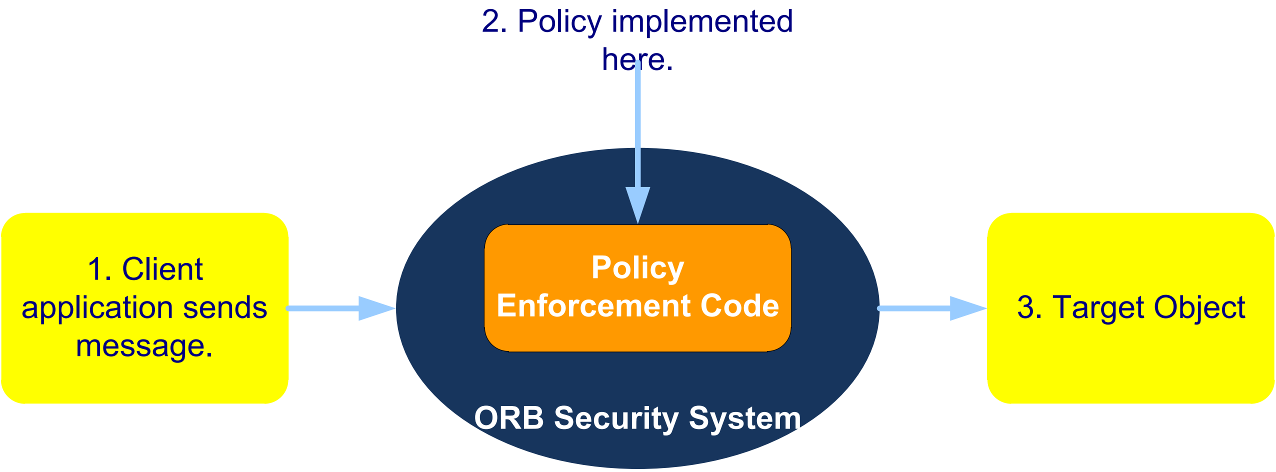
- Polymorphism describes the process of using an object in different ways for different set of inputs.
- Polyinstantiation is creating a new version of an object by replacing variables with other values (or variables).
  - Also used to prevent inference attacks against databases because it allows different versions of the same information to exist at different classification levels.
- Cohesion is the ability of a module to execute one function with little interaction from other modules.
- Coupling is a measure of the interconnection among modules in an application.

Distributed Object-Oriented Systems

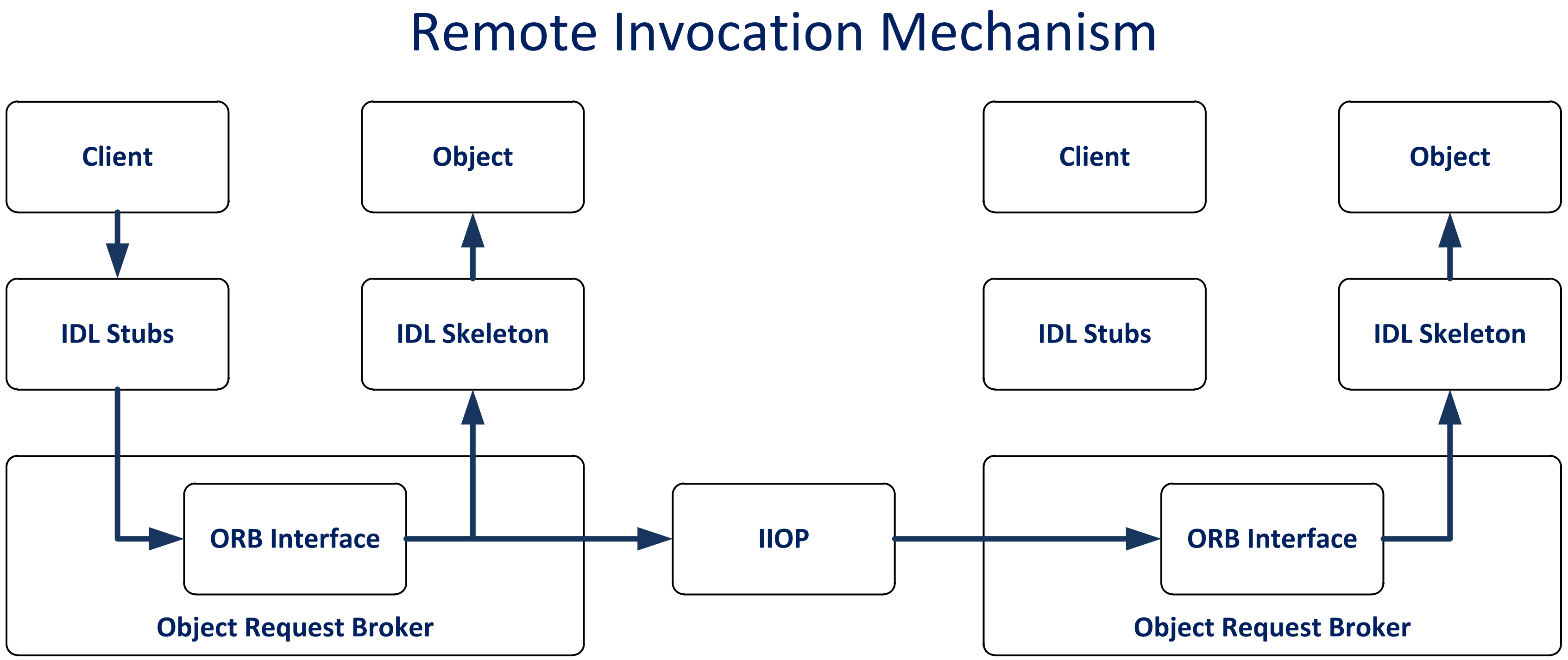
- Common Object Request Broker Architecture (CORBA)
  - A standard that “wrap” data objects. The object request broker (ORB) component enables heterogeneous applications and computing environment to interoperate.
- Component Object Model (COM) & Distributed Component Object Model (DCOM)
  - COM and DCOM are Microsoft object-oriented system standards for interoperate in a heterogeneous applications within a homogeneous (Microsoft) computing environment. It uses Object Linking & Embedding (OLE) and ActiveX.
- Java
  - Java Platform Standard Edition (Java SE)
  - Java Platform Enterprise Edition (Java EE)

Common Object Request Broker Architecture (CORBA)

- A set of standards that address the need for interoperability between hardware and software.
  - Allows applications to communicate with one another regardless of their location.
  - The Object Request Broker (ORB) establishes a client/ server relationship between objects.
  - The ORB enforces the system's security policy.


2. Policy implemented here.
1. Client application sends message.
Policy Enforcement Code
ORB Security System
3. Target Object

How CORBA Works



- CORBA uses Interface Definition Language (IDL) to describe interface requirements.
- CORBA uses Internet Inter-ORB Protocol (IIOP) to communicate between Object Request Brokers (ORBs).

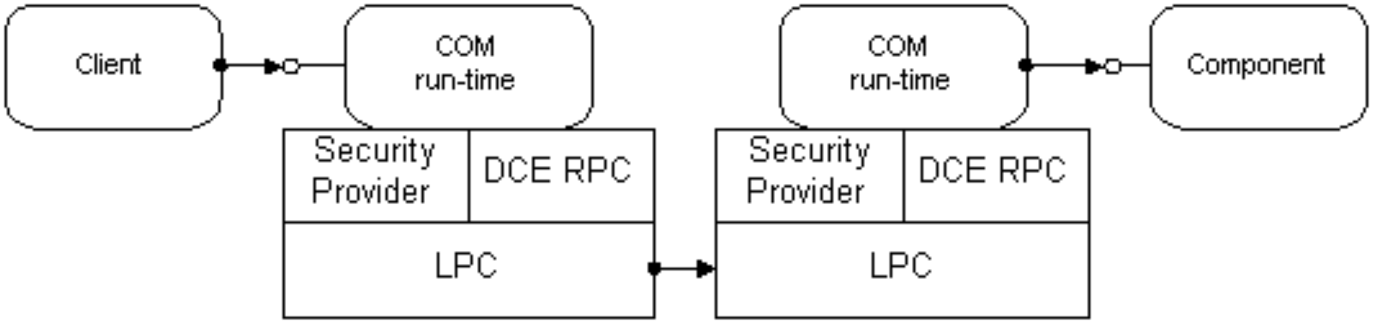
Source: http://www.omg.org/gettingstarted/corbafaq.htm

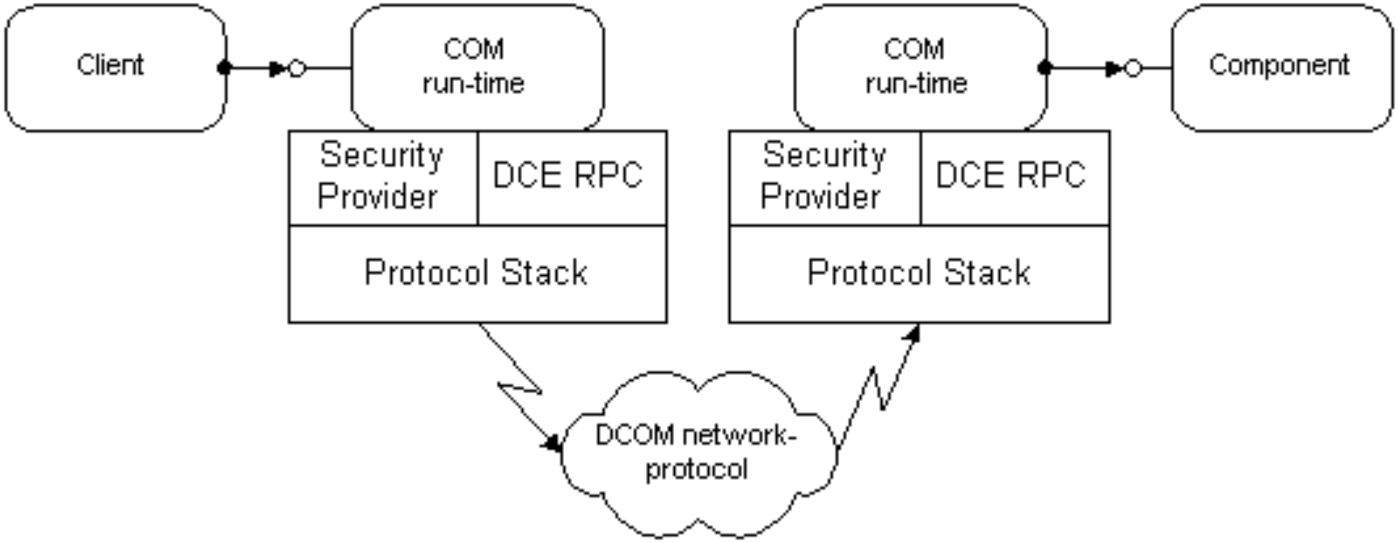
Component Object Models

- Component Object Model (COM) architecture
  - An open software architecture from DEC and Microsoft, allowing interoperation between ObjectBroker and OLE. Microsoft evolved COM into DCOM .

- Distributed Component Object Model (DCOM) architecture
  - An extension of COM to support objects distributed across a network.

Component Object Models





Source: http://technet.microsoft.com/en-us/library/cc722925.aspx

Object Linking & Embedding (OLE)

- OLE allows applications to share functionality by live data exchange and embedded data.
  - Embedding – places data in a foreign program.

    For example: Embedding of a Visio diagram inside of a PowerPoint slide.

  - Linking – capability to call a program.

    For example: Double click on the embedded Visio diagram in a PowerPoint slide and invoke Visio application to edit the diagram.

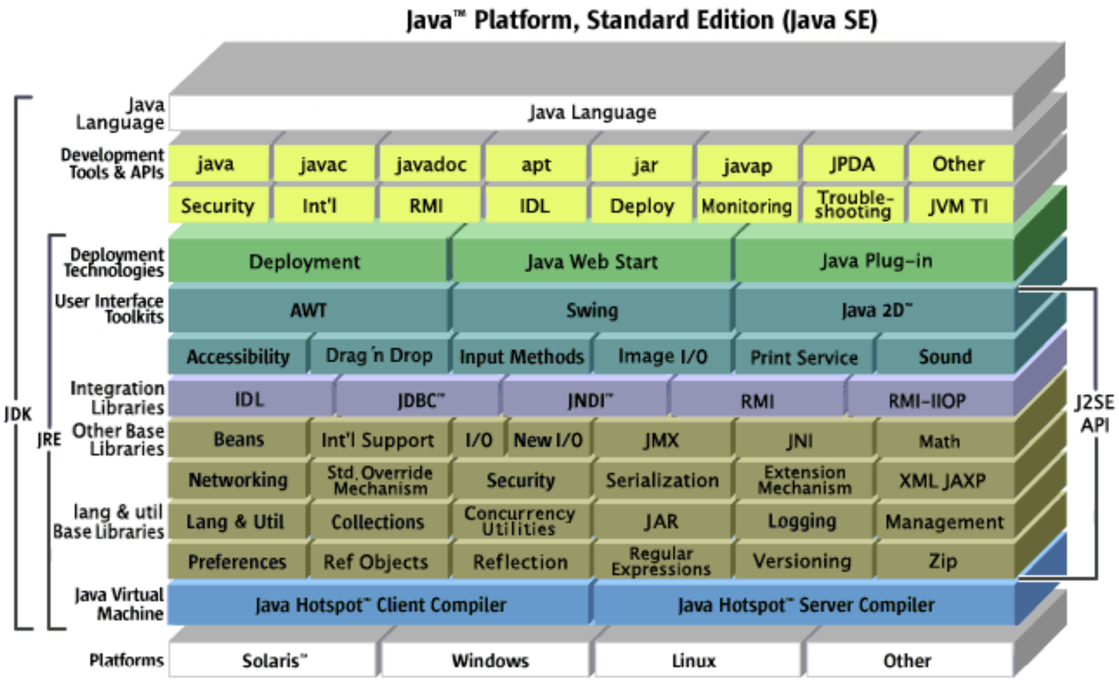
ActiveX

- A loosely defined set of technologies developed by Microsoft. ActiveX is a set of technologies that enables interactive contents for web.
- Elements of ActiveX technologies:
  - ActiveX Controls: interactive objects in a web page that provides user interaction functions.
  - ActiveX Documents: enable user to view non-HTML documents (e.g. Word, Excel, or PPT)
  - ActiveX Scripting Controls: integrated controls for ActiveX controls and/or Java Applets from web browser or server.
  - Java Virtual Machine (JVM): enables web browser (IE) to run Java applets and integrate with ActiveX controls.
  - ActiveX Server Framework: provide web server functions to support the above functions plus objects for database access and online transactions.

Java Platforms

Source: http://java.sun.com/javase/javasemap-lg.html

- Java is designed as a standard application "platform" for computing in a networked heterogeneous environment (developed by Sun Microsystems.)
- Java is a high-level programming language. Java source code are compiled into bytecode, which can then be executed by a Java interpreter.
- Java has three platforms:
  - Java SE (Standard Edition)
  - Java EE (Enterprise Edition)
  - Java ME (Micro Edition)



Java™ Platform, Standard Edition (Java SE)

JDK

JRE

J2SE API

Java Language: Java Language

Development Tools & APIs: java | javac | javadoc | apt | jar | javap | JPDA | Other

Security | Int'l | RMI | IDL | Deploy | Monitoring | Trouble-shooting | JVM TI

Deployment Technologies: Deployment | Java Web Start | Java Plug-in

User Interface Toolkits: AWT | Swing | Java 2D™

Accessibility | Drag 'n Drop | Input Methods | Image I/O | Print Service | Sound

Integration Libraries: IDL | JDBC™ | JNDI™ | RMI | RMI-IIOP

Other Base Libraries: Beans | Int'l Support | I/O | New I/O | JMX | JNI | Math

Networking | Std. Override Mechanism | Security | Serialization | Extension Mechanism | XML JAXP

lang & util Base Libraries: Lang & Util | Collections | Concurrency Utilities | JAR | Logging | Management

Preferences | Ref Objects | Reflection | Regular Expressions | Versioning | Zip

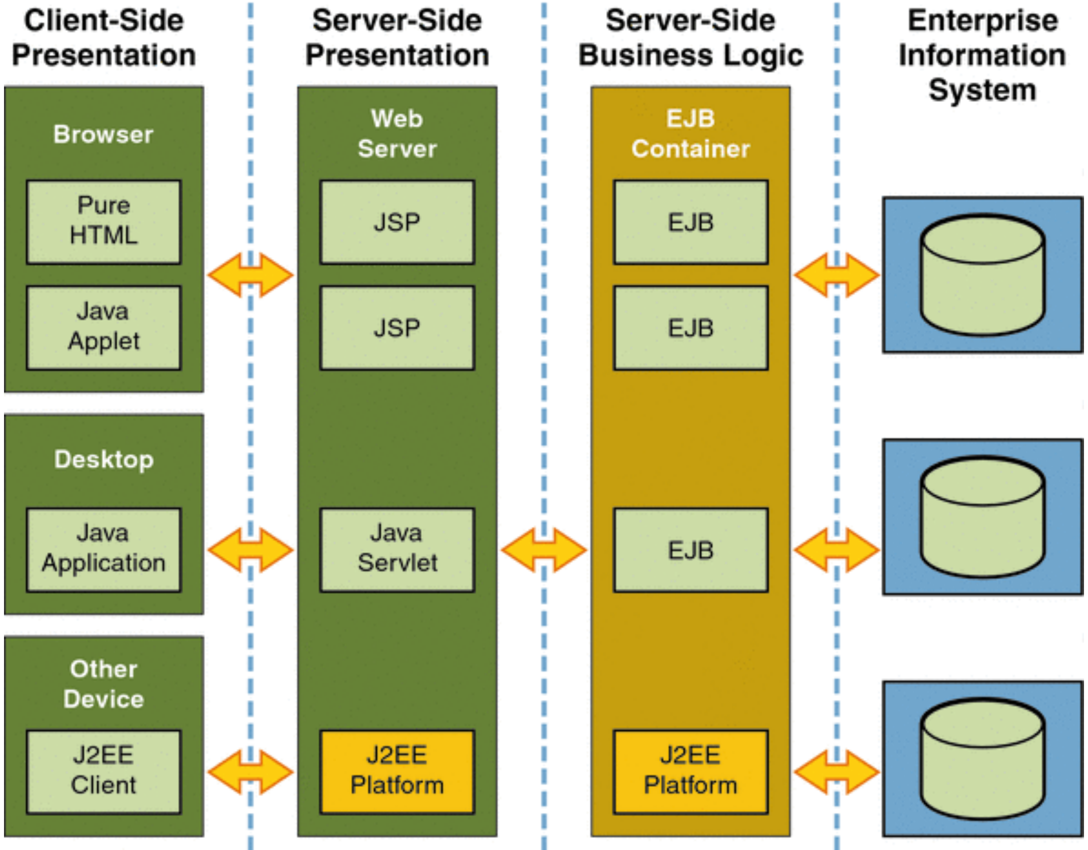
Java Virtual Machine: Java Hotspot™ Client Compiler | Java Hotspot™ Server Compiler

Platforms: Solaris™ | Windows | Linux | Other

Java Platform Enterprise Edition

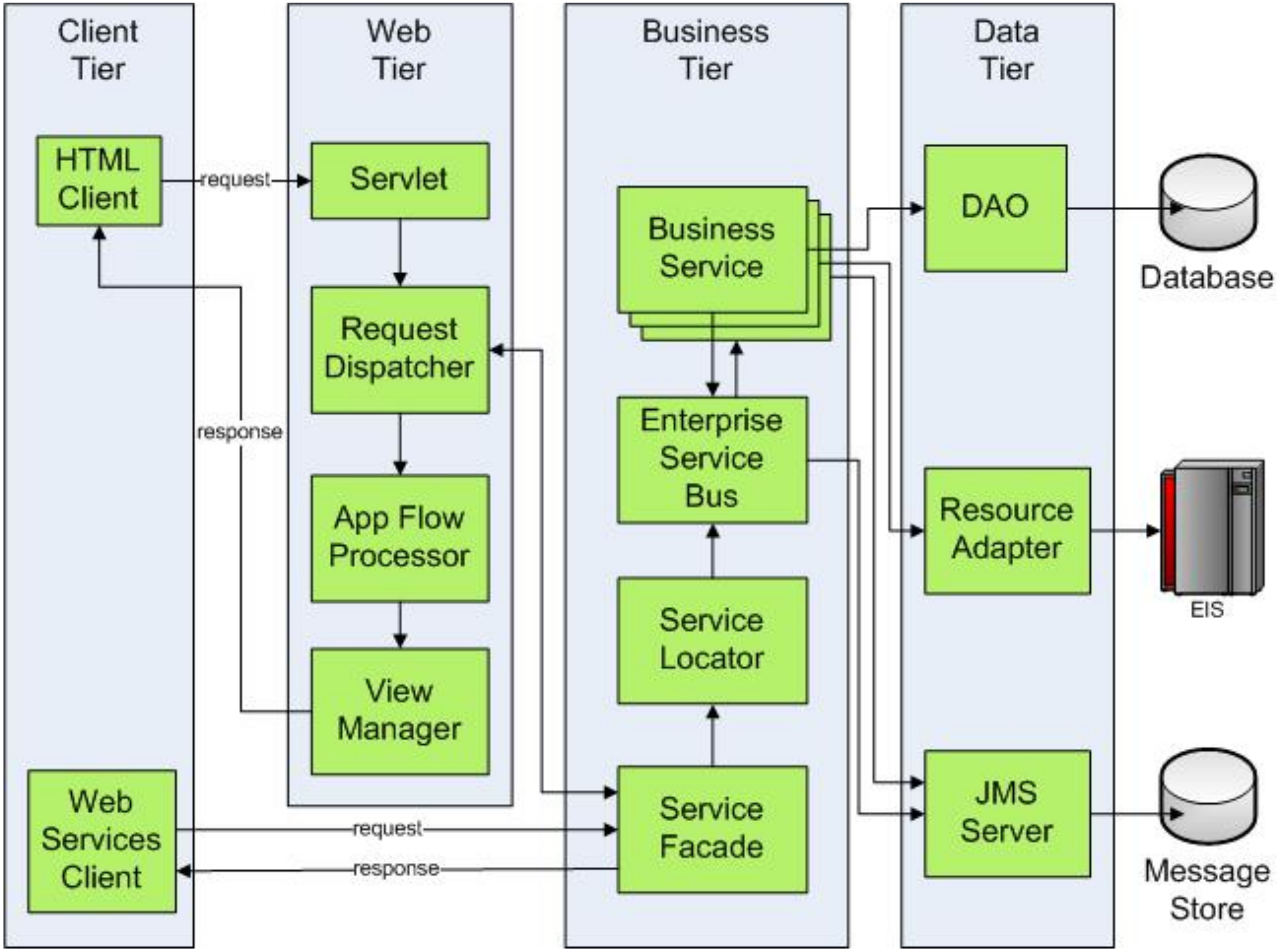
- Java Enterprise Edition (Java EE) uses Java SE as a foundation

- There are Containers are the runtime components for Java EE.
  - Applet
  - Application Client
  - Web
  - EJB



Source: http://docs.oracle.com/cd/E19879-01/820-4343/abeat/index.html

Java Application Server Architecture



Source: http://www.javaworld.com/javaworld/jw-01-2008/images/tomcat6_1.jpg

Questions:

- COBOL, FORTRAN, C, C+, C++, C# are what generation programming languages?
  -

- JavaScript, Perl, SQL, SGML are what generation programming languages?
  -

- What mechanism translates a high-level language (3^{rd} Generation) into machine language?
  -

Answers:

- COBOL, FORTRAN, C, C+, C++, C# are what generation programming languages?
  - 3rd Generation

- JavaScript, Perl, SQL, SGML are what generation programming languages?
  - 4th Generation

- What mechanism translates a high-level language (3rd Generation) into machine language?
  - Compiler

Questions:

- In object-oriented programming (OOP), what tells the system how to make object(s)?
  - –

- In OOP, what is the technique that keeps the data structures and methods (procedures) together?
  - –

- In OOP, what is the term that describes the process of using an object in different ways for different set of inputs?
  - –

Questions:

- In object-oriented programming (OOP), what tells the system how to make object(s)?
  - Class

- In OOP, what is the technique that keeps the data structures and methods (procedures) together?
  - Encapsulation

- In OOP, what is the term that describes the process of using an object in different ways for different set of inputs?
  - Polymorphism

Topics

Software Development Security Domain

- Governance & Management
- System Life Cycle and Security
- Software Environment and Security Controls
- Programming Languages
- ➡ Database and DB Warehousing Vulnerabilities, Threats, and Protections
- Software Vulnerabilities and Threats

Database Management System (DBMS)

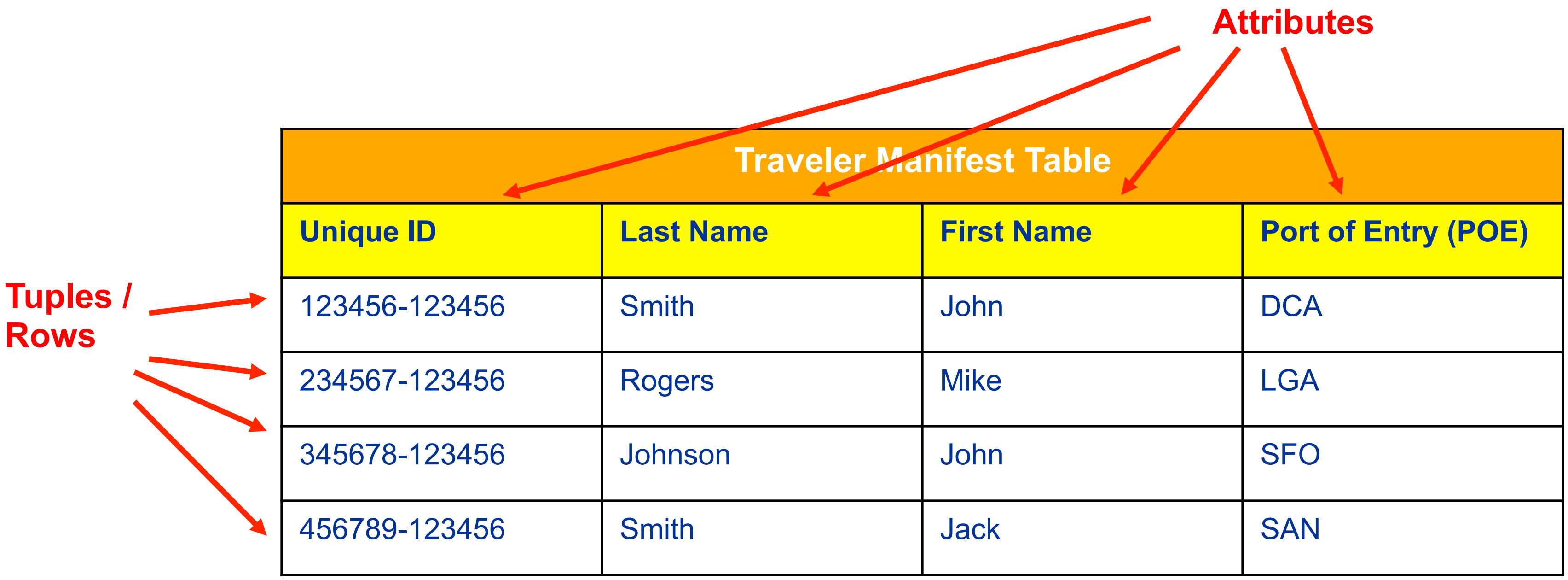
- Databases are developed to manage information from many sources in one location.
  - Eliminate the need for duplication of information in the system (thus preserves storage space).
  - Prevent inconsistency in data by making changes in one central location.
- DBMS consists of: hardware, software, and databases used to manage large sets of structured data (or information asset).
  - Enables Multiple Users and Applications to access, view, and modify data as Needed.
  - Can enforce control restrictions.
  - Provides data integrity and redundancy.
  - Established procedures for data manipulation.

DBMS Models

- Hierarchical DBMS
  - Stores information records (data) in a single table
  - Uses parent/child relationships
  - Limited to a single tree, no links between branches
- Network DBMS
  - Relationship of information records are of same type
  - All associations are direct connects, which forms a network
- Relational DBMS
  - Information records are structured in tables
  - Columns are the “attributes”, Rows are the “records”
- Object-oriented DBMS & object relational DBMS
  - Information records are objects
  - Relationships of objects are dynamic. The association can be made hierarchical, network, or relational

Relational DBMS (RDBMS)

- Information records (data) are structured in database tables.
  - Columns (attributes) represent the variables
  - Rows (records) contain the specific instance of information records
- Atomic relation = Every row/column position has always exactly one data value and never a set of values.



Attributes

Traveler Manifest Table

Unique ID	Last Name	First Name	Port of Entry (POE)
123456-123456	Smith	John	DCA
234567-123456	Rogers	Mike	LGA
345678-123456	Johnson	John	SFO
456789-123456	Smith	Jack	SAN

Tuples / Rows

Relational DBMS (RDBMS) – Primary & Foreign Keys

Data within the RDBMS…

- Unique ID is the "primary key". It identifies each row (record or tuple)
- Tuple cannot have a null value in the primary key.
- The primary key value guarantees that the tuple is unique
- "Foreign key" is an attribute or combination of attributes in another database table that matches the value of "primary key" in the first database table
  - Referential integrity rule
    - For any foreign key value, the reference relation to another table must have a tuple with the same value of the other table's primary key
    - A null value in the foreign key field prevents a join

Relational DBMS (RDBMS) – Primary & Foreign Keys

Traveler Manifest Table

Unique ID	Last Name	First Name	Port of Entry (POE)
123456-123456	Smith	John	DCA
234567-123456	Rogers	Mike	LGA
345678-123456	Johnson	John	SFO
456789-123456	Smith	Jack	SAN

Primary Key

Foreign Key

Baggage Manifest Table

Unique Tag ID	Airline	Flight Number	Unique ID
DCA456-123456	AA	AA-456	123456-123456
LGA567-123456	JetBlue	JB-567	234567-123456
SFO678-123456	United	UA-678	345678-123456
SAN89-123456	NW	NW-89	456789-123456

Relational DBMS (RDBMS) – View & Schema

- Data dictionary – Central repository of data elements and their relationships.
- Schema – Holds data that describes a database.
- View – Virtual relation defined by the database to keep subjects from viewing certain data.

Relational DBMS (RDBMS) – Security Issues

- Ensure integrity of input data (check input values, prevent buffer overflow).
- Access control ensuring only authorized user are performing authorized activities (“need-to-know”, “least privilege”).
- Preventing deadlock (stalemate when 2 or more processes are each waiting for the other to do something before they can proceed).

OODBMS &ORDBMS

- Class is a set of objects which shares a common structure and behavior. The relationship between classes can be hierarchical. (i.e. super-class, and subclass.)

- Object is a unique instance of a data structure defined according to the template provided by its class. Each object has its own values for the variables belonging to its class and can respond to the messages (methods) defined by its class.

- Method is a procedure or routine associated with one or more classes.

OODBMS &ORDBMS

- Object-oriented database (OODB) represents a “paradigm-shift” in the traditional database models (hierarchical, network, and relational).
  - Example of OODBMS: Versant.
- Object relations are build dynamically based on “business needs” instead of a series of fixed “business processes”.
  - Currently, the foundational DBMS engine for most of ORDBMS are still RDBMS. Object relations are build:
    - Presentation Layer: User/client level.
    - Business Logic Layer: Accepts commands from the presentation layer and send instructions to the data layer.
    - Data Layer: The database.
  - Example of ORDBMS: Oracle (8i, 9i, 10g), IBM DB2.

Data Warehousing and Mining

- Data Warehousing
  - Combines data from multiple databases or data sources into a large database called "data warehouse".
  - Requires more stringent security because all data is in a central facility.
- Data Mining
  - A.k.a. Knowledge-discovery in databases (KDD).
  - Practice of automatically searching large stores of data for patterns.
  - Data mining tools are used to find associations and correlations to product Metadata and can show previously unseen relationships.

Database Controls

- Granularity - The degree to which access to objects can be restricted.
- Content dependant access control
  - Permissions by View combining specific tables, columns, and key sets.
    - Authorizations for specific views having specific attributes, and for actions to perform within those views.
    - DAC, by specific grant to user or group by owner.
    - MAC, by classification level.
  - Cell Suppression
    - A technique used to hide or not show specific cells that contain information that could be used in an inference attack.

Database Controls

- Partitioning – Involved dividing a database into different parts which makes it harder for an individual to find connecting pieces
- Noise and perturbation – A technique of inserting bogus information aimed at misdirecting or confusing an attacker
- Concurrency – allowing multiple users to access the data contained within a database at the same time.
  - Making sure the most up to date information is available
  - If concurrent access is not managed by the Database Management System (DBMS) so that simultaneous operations don't interfere with one another problems can occur when various transactions interleave, resulting in an inconsistent database.

Database Controls – Types of Integrity Service

- Semantic integrity – Ensures that structural and semantic rules are enforced. Types of rules include data types, logical values, uniqueness constraints, and operations that could adversely affect the database.

- Entity integrity – Ensures that tuples are uniquely identified by primary key values.

- Referential integrity – Ensures that all foreign keys reference valid (and existing) primary keys. The other word, if a record does not include a primary key it cannot be referenced.

Database Controls – Configurable Controls for Integrity

- Rollback – is a statement that ends a current transaction and cancels all other changes
  - Occurs when some type of “glitch” is encountered during transaction
- Commit – terminates a transaction and executes all changes that were just made by a user.
  - If a user attempts a “commit” and it cannot be completed correctly…a “rollback” is executed to ensure integrity
- Savepoint(s) – are used to ensure that if a system failure occurs, or an error is detected, the database can return to a known good state prior to the problem
- Checkpoint(s) – (similar to Savepoints) when the database S/W fills to a certain amount of memory, a checkpoint is initiated, which saves the data from the memory segment to a temporary file.

Database Security Controls

- Polyinstantiation
  - Allows a relation to contain multiple rows with the same primary key
  - The multiple instances of Primary Keys are distinguished by their security levels
  - Used to prevent inference attacks by inserting “bogus” data at lower security levels

- Granularity – The degree to which access to objects can be restricted.
  - Granularity can be applied to both the actions allowable on objects, as well as to the users allowed to perform those actions on the object

Database Security Issues

- Online Transaction Processing (OLTP)
  - Usually used when multiple databases are clustered to provide fault tolerance and higher performance.
  - Transaction logs are used for synchronization of databases
  - OLTP transactions occur in real time which usually updates more than one database…which introduces integrity threats. To counteract this ACID test should be implemented.
    - Atomicity – Divides transactions into units of work and ensures all modifications take effect or none do
    - Consistency – A transaction must follow integrity policy for that specific database and ensure that all data is consistent in the different databases
    - Isolation – Transactions execute in isolation until completed, without interacting with other transactions
    - Durability – Once the transaction is verified as accurate on all systems, it is committed and the databases cannot be rolled back

Database Threats

- Aggregation
  - The act of combining information from separate sources.
  - The combined information has a sensitivity level greater that any of the individual parts.
- Inference
  - A user deduces (infers or figures out) the full story from pieces learned through aggregation and other sources.
  - Differs from aggregation in that data not explicitly available is used during the act of deduction (inference or plain figuring it out).
- Deadlocking
  - Two processes have locks on separate objects and each process is trying to acquire a lock on the object the other process has.

Questions:

- What are the four types of database management system (DBMS) models?
  - –
  - –
  - –
  - –
- In RDBMS, what is the definition for atomic relation?
  - –
- In RDBMS, what is a primary key?
  - –

Answers:

- What are the four types of database management system (DBMS) models?
  - Hierarchical
  - Network
  - Relational
  - Object-oriented

- In RDBMS, what is the definition for atomic relation?
  - Every row/column position always contains exactly one data value

- In RDBMS, what is a primary key?
  - The attribute that uniquely identifies each record

Questions:

- For RDBMS, how is the relationship between database tables created?

  –

- In an object-oriented relational database (ORDBMS), what are the three layers where the object relations are build?

  –

  –

  –

Answers:

- For RDBMS, how is the relationship between database tables created?
  - When an attribute of a database table is also an attribute of another database table

- In an object-oriented relational database (ORDBMS), what are the three layers where the object relations are build?
  - Presentation Layer: User/client level
  - Business Logic Layer: Accepts commands from the presentation layer and send instructions to the data layer
  - Data Layer: The database

Questions:

- For granularity access control, what are the two content dependent access control implementations for a DBMS?
  –
  –

- For DBMS, what is the term used that describes multiple users accessing data contained within a database at the same time?
  –

- What is the act of combining information from different sources?
  –

Answers:

- For granularity access control, what are the two content dependent access control implementations for a DBMS?
  - Permissions by view
  - Cell suppression
- For DBMS, what is the term used that describes multiple users accessing data contained within a database at the same time?
  - Concurrency
- What is the act of combining information from different sources?
  - Aggregation

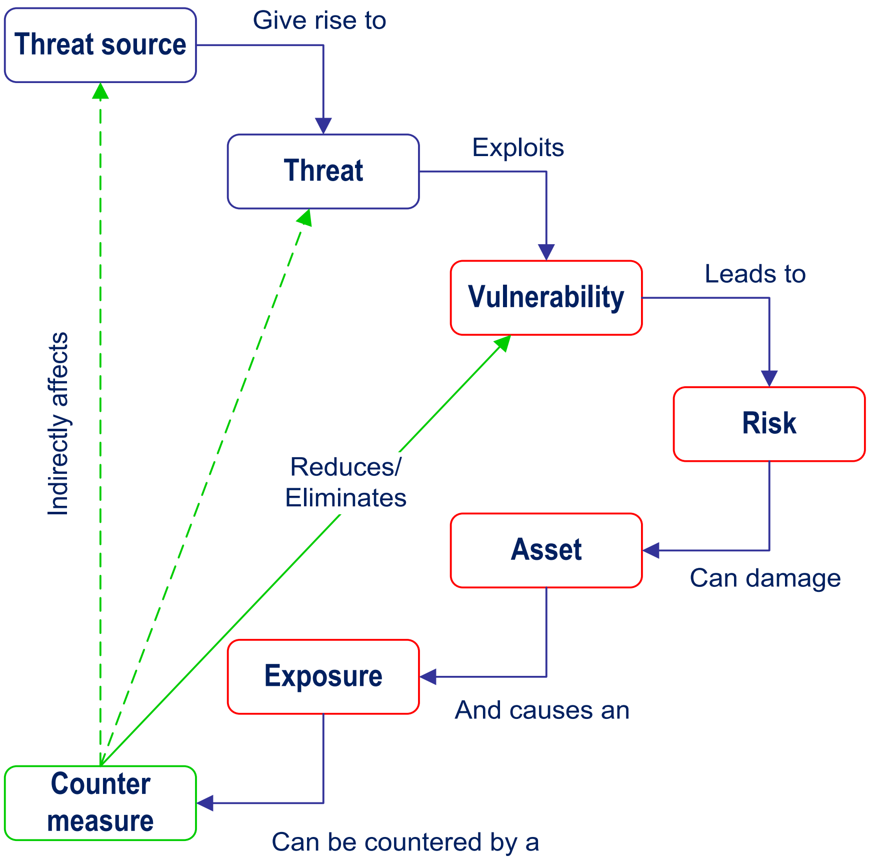
Topics

Software Development Security Domain

- Governance & Management
- System Life Cycle and Security
- Software Environment and Security Controls
- Programming Languages
- Database and DB Warehousing Vulnerabilities, Threats, and Protections
- Software Vulnerabilities and Threats

Relationship between Threat, Risk, and Countermeasure

- Threat source. Entity that may acts on a vulnerability.
- Threat. Any potential danger to information life cycle.
- Vulnerability. A system has weakness or flaw that may provide an opportunity to a threat source.
- Risk. The likelihood of a threat source take advantage of a vulnerability.
- Exposure. An instance of being compromised by Threat Source.
- Countermeasure / safeguard. An administrative, operational, or logical mitigation against potential risk(s).



Structural Defects, Weaknesses, Bugs, and Vulnerabilities

- Vulnerabilities are weaknesses that allow attackers to compromise the security objectives of information and/or information systems.
- Defects can be design flaws and/or implementation weaknesses.
- Bugs are implementation-level weaknesses.

<table>
<tr><th colspan="6">Information Systems Security Engineering (ISSE) Life Cycle</th></tr>
<tr><td>Discover Information Protection Needs</td><td>Define Requirements</td><td>Design System Architecture</td><td>Develop Detailed System Design & Security Controls</td><td>Implement System & Security Controls</td><td>Continuous Monitoring</td></tr>
</table>

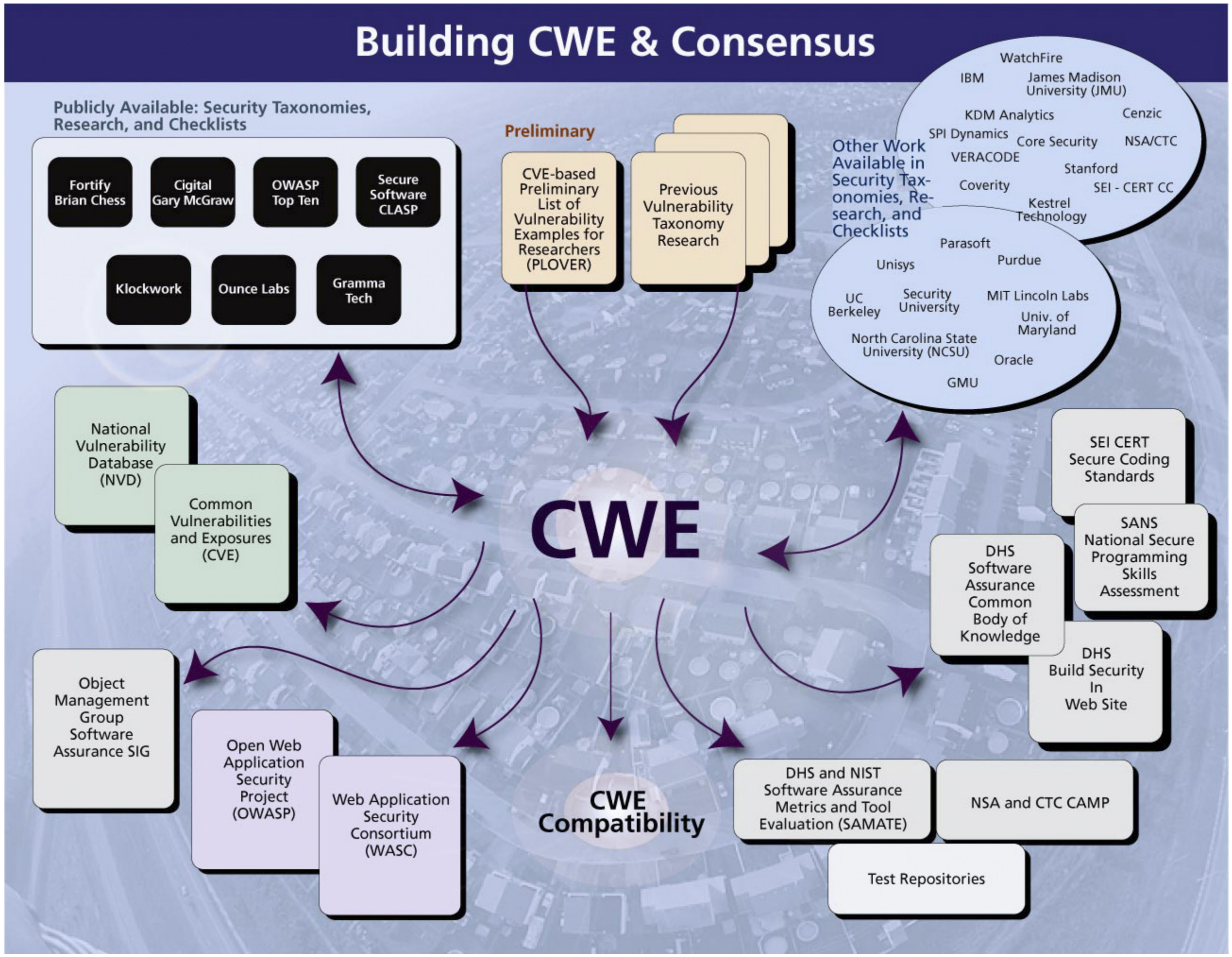
<table>
<tr><th colspan="7">Software Development: Rational Unified Process</th></tr>
<tr><td colspan="2">Inception</td><td colspan="2">Elaboration</td><td colspan="2">Construction</td><td>Transition</td></tr>
<tr><td>Business Modeling</td><td>Requirements</td><td colspan="2">Analysis & Design</td><td colspan="2">Implementation</td><td>Deployment/CM</td></tr>
<tr><th colspan="7">McGraw’s Software Security Touch Points</th></tr>
<tr><td colspan="2">Requirements and Use Cases</td><td>Architecture & Design</td><td>Test Plans</td><td>Code</td><td>Test & Test Results</td><td>Feedback From The Fields</td></tr>
</table>

Focus on software structural defects (flaws)

Focus on software weaknesses (bugs)

Common Weakness Enumeration (CWE)

- CWE is an online dictionary of software weaknesses.


Building CWE & Consensus
Publicly Available: Security Taxonomies, Research, and Checklists
Fortify Brian Chess
Cigital Gary McGraw
OWASP Top Ten
Secure Software CLASP
Klockwork
Ounce Labs
Gramma Tech
Preliminary
CVE-based Preliminary List of Vulnerability Examples for Researchers (PLOVER)
Previous Vulnerability Taxonomy Research
Other Work Available in Security Taxonomies, Research, and Checklists
WatchFire
IBM
James Madison University (JMU)
KDM Analytics
Cenzic
SPI Dynamics
Core Security
NSA/CTC
VERACODE
Stanford
Coverity
SEI - CERT CC
Kestrel Technology
Parasoft
Unisys
Purdue
UC Berkeley
Security University
MIT Lincoln Labs
Univ. of Maryland
North Carolina State University (NCSU)
Oracle
GMU
National Vulnerability Database (NVD)
Common Vulnerabilities and Exposures (CVE)
CWE
SEI CERT Secure Coding Standards
SANS National Secure Programming Skills Assessment
DHS Software Assurance Common Body of Knowledge
DHS Build Security In Web Site
Object Management Group Software Assurance SIG
Open Web Application Security Project (OWASP)
Web Application Security Consortium (WASC)
CWE Compatibility
DHS and NIST Software Assurance Metrics and Tool Evaluation (SAMATE)
NSA and CTC CAMP
Test Repositories

Reference: Common Weakness Enumeration (CWE) (http://cwe.mitre.org/)

2011 CWE/SANS Top 25 Most Dangerous Programming Errors

Rank	Score	ID	Name
[1]	93.8	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
[2]	83.3	CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
[3]	79.0	CWE-120	Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')
[4]	77.7	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
[5]	76.9	CWE-306	Missing Authentication for Critical Function
[6]	76.8	CWE-862	Missing Authorization
[7]	75.0	CWE-798	Use of Hard-coded Credentials
[8]	75.0	CWE-311	Missing Encryption of Sensitive Data
[9]	74.0	CWE-434	Unrestricted Upload of File with Dangerous Type
[10]	73.8	CWE-807	Reliance on Untrusted Inputs in a Security Decision
[11]	73.1	CWE-250	Execution with Unnecessary Privileges
[12]	70.1	CWE-352	Cross-Site Request Forgery (CSRF)
[13]	69.3	CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')
[14]	68.5	CWE-494	Download of Code Without Integrity Check
[15]	67.8	CWE-863	Incorrect Authorization
[16]	66.0	CWE-829	Inclusion of Functionality from Untrusted Control Sphere
[17]	65.5	CWE-732	Incorrect Permission Assignment for Critical Resource
[18]	64.6	CWE-676	Use of Potentially Dangerous Function
[19]	64.1	CWE-327	Use of a Broken or Risky Cryptographic Algorithm
[20]	62.4	CWE-131	Incorrect Calculation of Buffer Size
[21]	61.5	CWE-307	Improper Restriction of Excessive Authentication Attempts
[22]	61.1	CWE-601	URL Redirection to Untrusted Site ('Open Redirect')
[23]	61.0	CWE-134	Uncontrolled Format String
[24]	60.3	CWE-190	Integer Overflow or Wraparound
[25]	59.9	CWE-759	Use of a One-Way Hash without a Salt

Reference: http://cwe.mitre.org/top25/

Categories of Software Weaknesses

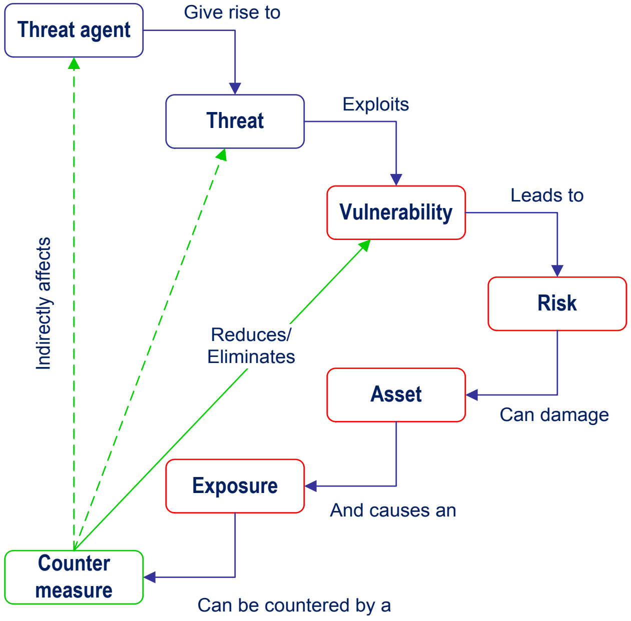
- Insecure interaction between components
  - "Weaknesses related to insecure ways in which data is sent and received between separate components, modules, programs, processes, threats, or systems."
- Risky resource management
  - "Weaknesses related to ways in which software does not properly manage the creation, usage, transfer, or destruction of important system resources."
- Porous defenses
  - "Weaknesses related to defensive techniques that are often misused, abused, or just plain ignored."

Reference: http://cwe.mitre.org/top25/index.html

Reduce / Eliminate Software Vulnerabilities

- Addressing structural/design flaws
  - Understand the information protection needs
  - Develop use/abuse cases
  - Define system security requirements
  - Design system architecture
  - Develop detailed system design & security controls

- Addressing software bugs (weaknesses)
  - Develop detailed software design & specifications
  - Implement code reviews
    - Static code analyzers
  - Perform tests
    - Unit, subsystems, system, acceptance tests
    - Vulnerability scanners



Information Systems Security Engineering (ISSE) Life Cycle					
Discover Information Protection Needs	Define Requirements	Design System Architecture	Develop Detailed System Design & Security Controls	Implement System & Security Controls	Continuous Monitoring

McGraw's Software Security Touch Points					
Requirements and Use Cases	Architecture & Design	Test Plans	Code	Test & Test Results	Feedback From The Fields

Focus on software structural defects (flaws)

Focus on software weaknesses (bugs)

Threats to Software – Buffer Overflow …(1/2)

- One of the oldest and most common problems to software.
  - Wagner et. al. estimated over 50% of all vulnerabilities are due to buffer overflow.*
- No. 3 in 2011 CWE/SANS Top 25.
- A buffer overflow occurs when a program or process tries to store more data in a buffer (temporary data storage area) than it was intended to hold.
- In buffer overflow attacks, the extra data may contain codes designed to trigger specific actions, in effect sending new instructions to the attacked computer that could, for example, damage the user's files, change data, or disclose confidential information.

Reference:

- ** A First Step Towards Automated Detection of Buffer Over-run Vulnerabilities*, D. Wagner, et. al., 2000.
- *2011 CWE/SANS Top 25 Most Dangerous Programming Errors*, MITRE, September 2011.

Threats to Software – Buffer Overflow …(2/2)

Recommended countermeasure to prevent buffer overflow attacks:

- Patch, patch, and patch
- Always check for inputs. Enforce controls at the interfaces
- Ensure applications are not exposed to faulty components
- Use language and frameworks that are relatively “immune” to buffer overflows:

Language/ Environment	Compiled / Interpreted	Strongly Typed	Direct Memory Access	Safe/ Unsafe
Java, JVM	Both	Yes	No	Safe
.NET	Both	Yes	No	Safe
Perl	Both	Yes	No	Safe
Python	Interpreted	Yes	No	Safe
Ruby	Interpreted	Yes	No	Safe
C/C++	Compiled	No	Yes	Unsafe
Assembly	Compiled	No	Yes	Unsafe
COBOL	Compiled	Yes	No	Safe

Reference: *Buffer Overflows – OWASP* (https://www.owasp.org/index.php/Buffer_Overflows) (5/14/12)

Threats to Software – Cross-site Scripting (XSS) …(1/2)

- XSS is one of the most prevalent web application (web app) security flaw.
- No. 4 in 2011 CWE/SANS Top 25 and OWASP Top 10.
  - XSS occurs when a web app in web browser accepts "untrusted data" and sends it to a web app server without proper validation. Attackers can then execute scripts in a victim's web browser to hijack user sessions, deface web sites, insert malicious content, redirect users, etc.
  - These "untrusted data" could be JavaScript, or other browser-executable RIA contents such as Active X, Flash, Silverlight, etc.

Reference: *CWE-79: Improper Neutralization of Input During Web Page Generation (Cross-site Scripting)* (http://cwe.mitre.org/data/definitions/79.html) (6/2/2013).

Threats to Software – Cross-site Scripting (XSS) …(2/2)

- Recommended countermeasures to prevent XSS attacks:
  - Never insert untrusted data except in allowed locations.
  - Use “escaping” (a.k.a. output encoding) technique.
  - Use an HTML policy engine to validate or clean user-driven HTML in an outbound way.
  - Prevent DOM-based XSS.
  - Use “HTTPOnly” cookie flag

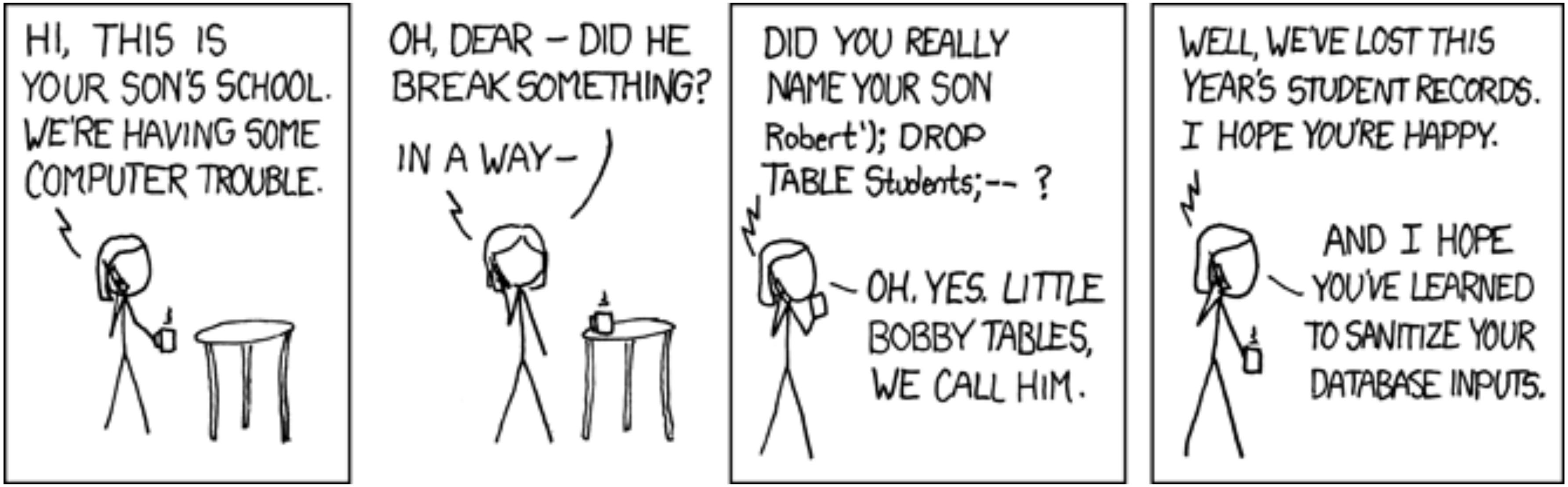
Reference:
- CWE-7: Improper Neutralization of Input During Web Page Generation (Cross-site Scripting) (http://cwe.mitre.org/data/definitions/79.html) (6/2/2013)
- *XSS (Cross Site Scripting Prevention Cheat Sheet – OWASP* (https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet) (5/14/12)

Threats to Software – SQL Injection …(1/3)

- In 2011, SQL Injection is No.1 in both CWE/SANS Top 25 and OWASP Top 10.
- SQL injection occurs when an application sends “untrusted data” to an interpreter as a part of command or query.
  - These “untrusted data” can be in SQL queries, LDAP queries, Xpath queries, etc.
- Attackers can:
  - Alter the logic of SQL queries to bypass security (e.g., authentication, authorization, etc.) to gain unauthorized access to data (e.g., steal, corrupt, or change data.)
  - Trick the interpreter to execute unintended commands

Reference: *CWE-89: Improper Neutralization of Special Elements used in an SQL Command (SQL Injection)* (http://cwe.mitre.org/top25/#CWE-89) (6/2/2013)

Threats to Software – SQL Injection …(2/3)


HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY–
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH. YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

Reference: *Exploits of a Mom* http://xkcd.com/327/) (5/14/12)

Threats to Software – SQL Injection …(3/3)

- Recommended countermeasures to prevent SQL injection attacks:
  - Use "prepared statements" (/ parameterized queries) such as:
    - Java EE – use PreparedStatement() with bind variables
    - .NET – use parameterized queries like SqlCommand() or OleDbCommand() with binding variables
    - PHP – use PDO with strongly typed parameterized queries (use binParam())
  - Use stored procedures
  - Escaping all "user supplied" inputs. Treat user inputs as untrusted data, don't insert them directly as a part of a SQL query

Reference: *SQL Injection Prevention Cheat Sheet – OWASP* (https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet) (5/14/12)

Threats to Software – OS Command Injection

- OS Command Injection (a.k.a. Shall Injection) is #2 in 2011 Top 25 CWE.
  - Attacker injects and execute unwanted system commands through vulnerable applications that lacks proper input data validation (e.g., forms, cookies, HTTP headers etc.)
  - As with SQL Injection, it is a variant of Code Injection attack, except it utilizes applications running as "system" instead of "user".
- Recommended countermeasure:
  - Validate inputs
  - Use application provided API
  - Run automated code analysis tools

Reference: *CWE-78: Improper Neutralization of Special Elements used in an OS Command* (http://cwe.mitre.org/data/definitions/78.html) (6/2/2013)

Use of Automated Analysis Tools

- For detection of structural flaws (“defects”)
  - Tool integration frameworks (a.k.a. IDEs)
    - Software engineering management, architecture/ design modeling (MBSE), requirements traceability, design patterns
  - Code quality review tools

- For detection of software weakness (“bugs”)
  - Static code analysis tools
    - Source code security analyzers, byte code scanners, binary code scanners
  - Dynamic analysis tools
    - Web application vulnerability scanners, database vulnerability scanners
  - Network vulnerability scanners
  - SCAP-compatible security configuration scanners

<table>
<tr><th colspan="6">Information Systems Security Engineering (ISSE) Life Cycle</th></tr>
<tr><td>Discover Information Protection Needs</td><td>Define Requirements</td><td>Design System Architecture</td><td>Develop Detailed System Design & Security Controls</td><td>Implement System & Security Controls</td><td>Continuous Monitoring</td></tr>
</table>

<table>
<tr><th colspan="7">Software Development: Rational Unified Process</th></tr>
<tr><td colspan="2">Inception</td><td colspan="2">Elaboration</td><td colspan="2">Construction</td><td>Transition</td></tr>
<tr><td>Business Modeling</td><td>Requirements</td><td colspan="2">Analysis & Design</td><td colspan="2">Implementation</td><td>Deployment/CM</td></tr>
<tr><th colspan="7">McGraw’s Software Security Touch Points</th></tr>
<tr><td colspan="2">Requirements and Use Cases</td><td>Architecture & Design</td><td>Test Plans</td><td>Code</td><td>Test & Test Results</td><td>Feedback From The Fields</td></tr>
</table>

Focus on software structural defects (flaws)

Focus on software weaknesses (bugs)

Malicious Code / Malware

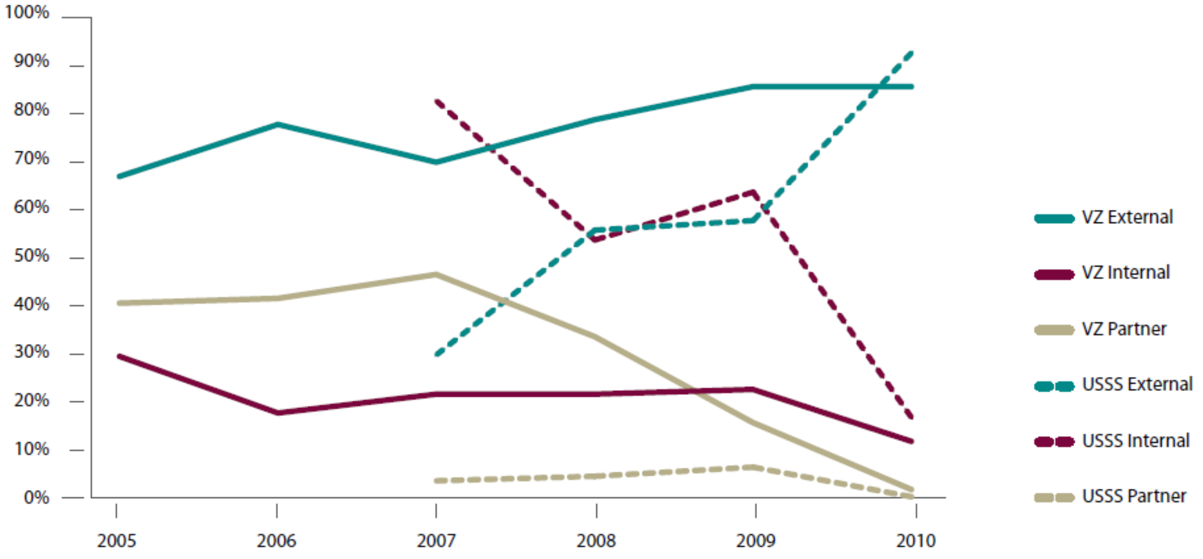
- Malicious code / malware (MALicious softWARE)

- For CISSP, there are many types of “malware”:
  - Viruses
  - Worms
  - Trojan horses
  - Rootkits
  - Spyware
  - Some cookies…

Reference: http://youtu.be/cf3zxHuSM2Y

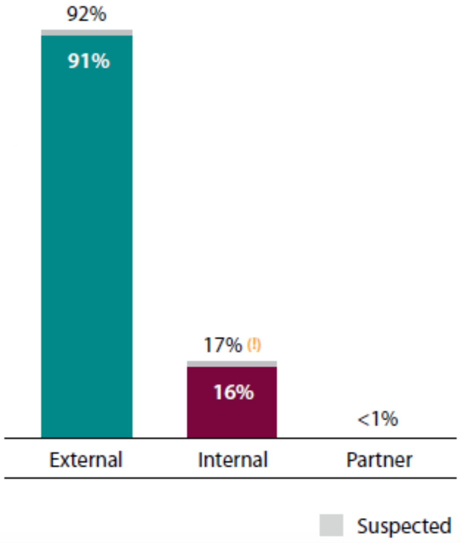
Malware as a Threat to Information Operations …(1/3)

- Operations are getting better at addressing insider threats…



  - » VZ (Verizon)
  - » USSS (United States Secret Service)

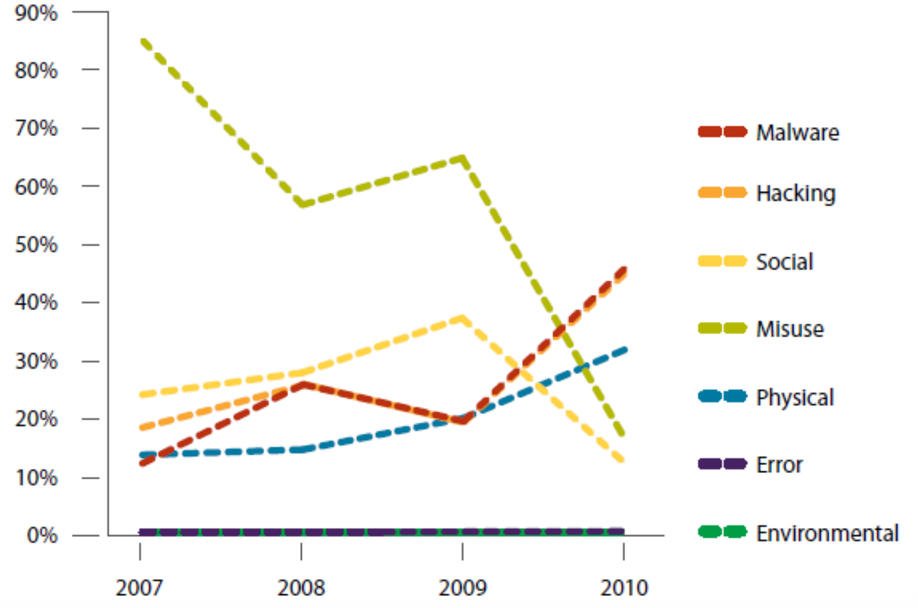
- The fact is that most of threats are still from external threat agents.



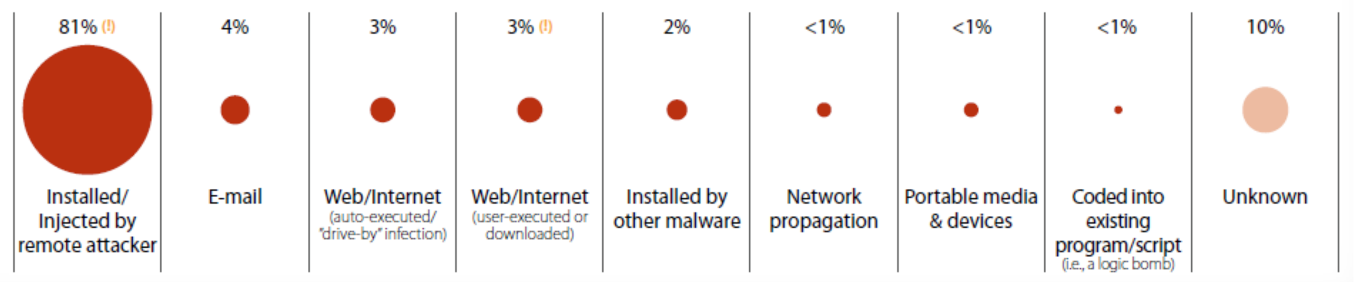
Reference: *2011 Data Breach Investigations Report*, Verizon, January 2012 (http://www.verizonbu
resources/reports/rp_data-breach-investigations-report-2011_en_xg.pdf)

Malware as a Threat to Information Operations …(2/3)

- Most of data breaches are from hacking and malware...



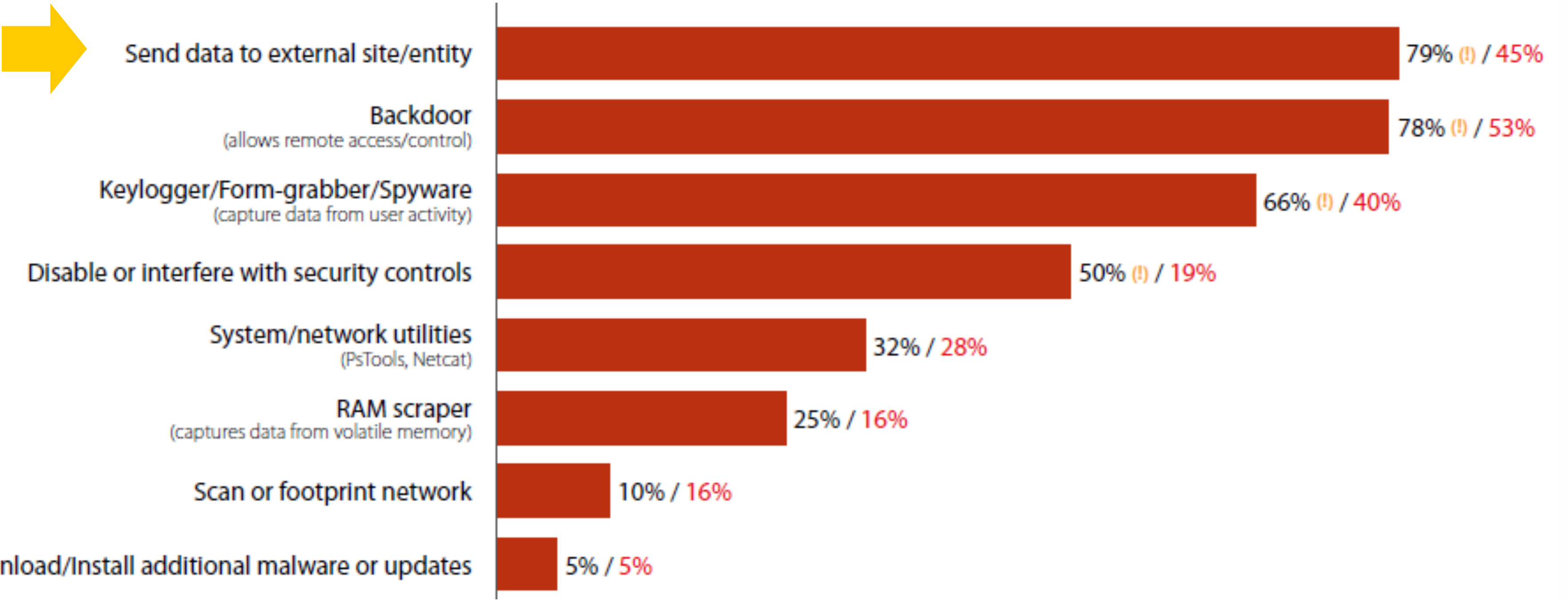
- Majority of malware are installed remotely...



Reference: *2011 Data Breach Investigations Report*, Verizon, January 2012 (http://www.verizonbusiness.com/resources/reports/rp_data-breach-investigations-report-2011_en_xg.pdf)

Malware as a Threat to Information Operations …(3/3)

- Advanced Persistent Threat (APT) is very real
  - Malware is now a tool for hackers
  - They are stealing data...



Reference: *2011 Data Breach Investigations Report*, Verizon, January 2012 (http://www.verizonbusiness.com/resources/reports/rp_data-breach-investigations-report-2011_en_xg.pdf)

Threats to Software – Malicious Code / Malware

Malicious code / malware (MALicious softWARE)

- Virus – A program or piece of code that is loaded onto your computer without your knowledge and runs against your wishes. Viruses can also replicate themselves. A simple virus that can make a copy of itself over and over again is relatively easy to produce.
- Polymorphic virus – A virus that changes its virus signature (i.e., its binary pattern) every time it replicates and infects a new file in order to keep from being detected by an antivirus program.

Threats to Software – Malicious Code / Malware

- Boot sector virus – A boot sector virus is a common type of virus that replaces the boot sector with its own code. Since the boot sector executes every time a computer is started, this type of virus is extremely dangerous.
- Macro virus – A type of computer virus that is encoded as a macro embedded in a document. Many applications, such as Microsoft Word and Excel, support powerful macro languages. These applications allow you to embed a macro in a document, and have the macro execute each time the document is opened.
  - According to some estimates, 75% of all viruses today are macro viruses. Once a macro virus gets onto your machine, it can embed itself in all future documents you create with the application.

Threats to Software – Malicious Code / Malware

- Worm – A program or algorithm that replicates itself over a computer network and usually performs malicious actions. Differ from viruses in that they are self contained and do not need a host application to reproduce.
- Logic bomb – Also called *slag code*, programming code (typically malicious) added to the software of an application or operating system that lies dormant until a predetermined period of time or event occurs, triggering the code into action.

Threats to Software – Malicious Code / Malware

- Trojan horse – A destructive program that masquerades as a benign application. Unlike viruses, Trojan horses do not replicate themselves but they can be just as destructive. One of the most insidious types of Trojan horse is a program that claims to rid your computer of viruses but instead introduces viruses onto your computer.
- Data diddler – refers to the payload in a Trojan or virus that deliberately corrupts data, generally by small increments over time.
- Hoax – usually warnings about viruses that do not exist, generally carry a directive to the user to forward the warning to all addresses available to them.
- Trapdoor/backdoor – can also be called a maintenance hook; it’s a hidden mechanism that bypasses access control measures.

Validation Time… ☺

1. Classroom Exercise

2. Review Answers

Classroom Exercise: Constructing a Security Engineering Project... (1/5)

Systems Engineering (SE) Activities	Security Engineering (ISSE) Activities
Discover Mission/Business Needs The SE helps the customer understand and document the information management needs that support the business or mission. Statements about information needs may be captured in an information management model (IMM).	**Discover Information Protection Needs** The ISSE facilitates the system owners, architects, and engineers in assessing the information protection needs by performing risk assessment, capturing the information management model (IMM), defining the information protection policy (IPP) and compile them into a comprehensive information management plan (IMP).
Define System Requirements The SE allocates identified needs to systems. A system context is developed to identify the system environment and to show the allocation of system functions to that environment. A preliminary system Concept of Operations (CONOPS) is written to describe operational aspects of the candidate system (or systems). Baseline requirements are established.	**Define System Security Requirements** The ISSE allocates the information protection needs in accordance with the information management plan (IMP) that aligns with a preliminary system security concept of operations (CONOPS) and generates a set of baseline security requirements in accordance with FIPS 200.
Design System Architecture The SE performs functional analysis and allocate by analyzing candidate architectures, allocating requirements, and selecting mechanisms. The system engineer identifies components or elements, allocates functions to those elements, and describes the relationships between the elements.	**Design System Security Architecture** The ISSE works in conjunction with system architect and engineers in defining a system architecture using the designated system architecture framework to explain the system architecture at the conceptual and logic levels in meeting the defined baseline security requirements.

Classroom Exercise: Constructing a Security Engineering Project... (2/5)

Systems Engineering (SE) Activities	Security Engineering (ISSE) Activities
Develop Detailed System Design The SE analyzes design constrains, analyzes trade-offs, does detailed system design, and considers life-cycle support. The systems engineer traces all of the system requirements to the elements until all are addressed. The final detailed design results in component and interface specifications that provide sufficient information for acquisition where the system is implemented.	**Develop Detailed Security Design** The ISSE analyzes the design constrains, trade-offs from the system architecture and begin to work with system architect and engineers to define detailed system design.
Implement System The SE moves the system from specifications to the tangible. The main activities are acquisition, integration, configuration, testing, documentation, and training. Components are tested and evaluated to ensure that they must meet the specifications. After successful testing, the individual components – hardware, software, and firmware – are integrated, properly configured, and tested as a system.	**Implement System Security** The ISSE works with SE in implementing the baseline detailed system design. The information systems security engineer provide inputs to the certification and accreditation (C&A) process and verify the implemented system design meets the defined baseline security requirements against the identified threats .
Assess System Effectiveness The results of each activity are evaluated to ensure that the system will meet the user's needs by performing the required functions to the required quality standard in the intended environment. The systems engineer examines how well the system meets the needs of the mission.	**Assess System Security Effectiveness** The ISSE focuses on the effectiveness of the implemented security controls and countermeasures, and validates them against the defined information management plan (IMP).

Classroom Exercise: Constructing a Security Engineering Project... (3/5)

1. Discovering the Information Protection Needs
   - 1.1 Collect & analyze system information: Business/ Mission Needs, high-level concept of information operations, data sensitivity, mode of operations, etc.
   - 1.2 Perform Risk Assessment of the “to-be” information system
   - 1.3 Generate Information Management Model (IMM)
   - 1.4 Generate Information Protection Policy (IPP)
   - 1.5 Assemble Information Management Plan
2. Defining the System Security Requirements
   - 2.1 Define security context description (i.e. scope)
   - 2.2 Generate system security requirements: functional & assurance

Classroom Exercise: Constructing a Security Engineering Project… (4/5)

3. Designing the System Security Architecture

    3.1 Describe the Conceptual Security Architecture

    3.2 Describe the Logical Security Architecture

    3.3 Describe the Physical Security Architecture

4. Developing the Detailed System Security Design

    4.1 Describe the Security Architecture at the components level

    4.1.1 Defending the Network & Infrastructure

    4.1.2 Defending the Enclave Boundary

    4.1.3 Defending the Computing Environment

    4.1.4 Supporting the IT Infrastructure

Classroom Exercise: Constructing a Security Engineering Project... (5/5)

5. Implementing the System Security

   5.1 Implement system design for defending the network infrastructure

   5.2 Implement system design for defending the enclave boundary

   5.3 Implement system design for defending the computing environment

   5.4 Implement system design for supporting the IT Infrastructure

6. Assessing the Security Effectiveness

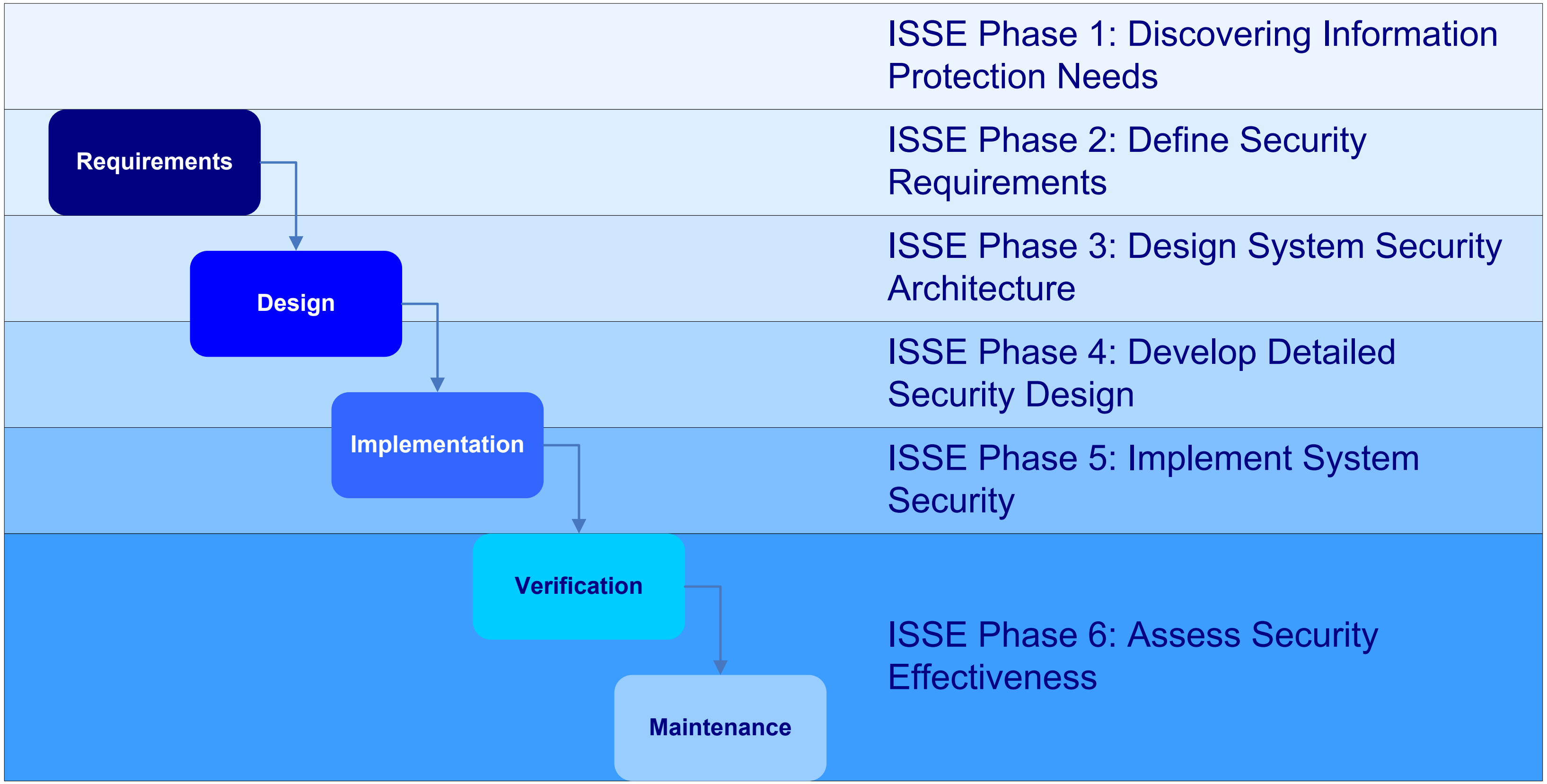
   6.1 Perform analysis on Security Requirements Traceability matrix (S-RTM)

   6.2 Verify conformance of system design to S-RTM

   6.3 Validate security implementation to S-RTM

   6.4 Support Security Certification & Accreditation (C&A) Team

Group 1: Waterfall SDLC Model



ISSE Phase 1: Discovering Information Protection Needs

Requirements

ISSE Phase 2: Define Security Requirements

Design

ISSE Phase 3: Design System Security Architecture

Implementation

ISSE Phase 4: Develop Detailed Security Design

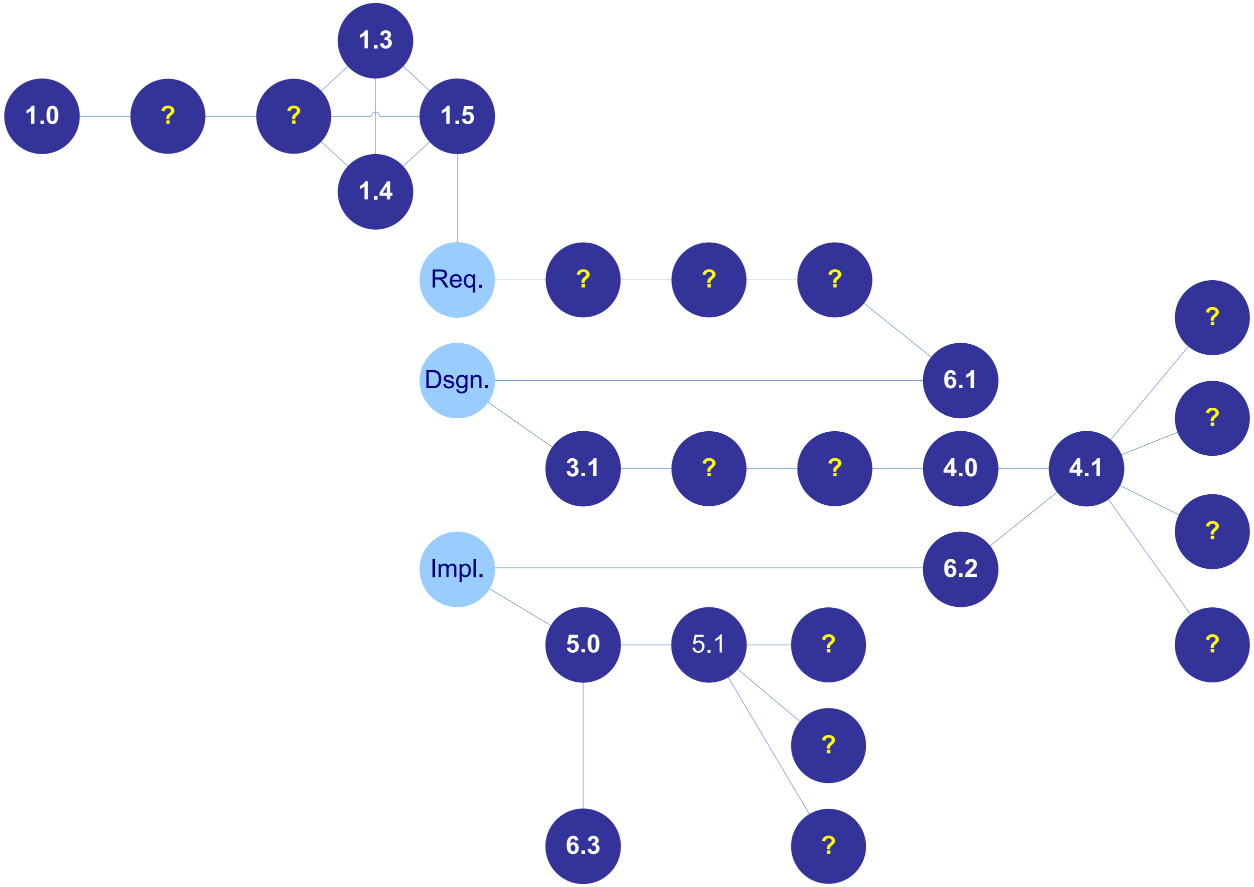
ISSE Phase 5: Implement System Security

Verification

Maintenance

ISSE Phase 6: Assess Security Effectiveness

Group 1: Waterfall SDLC Model



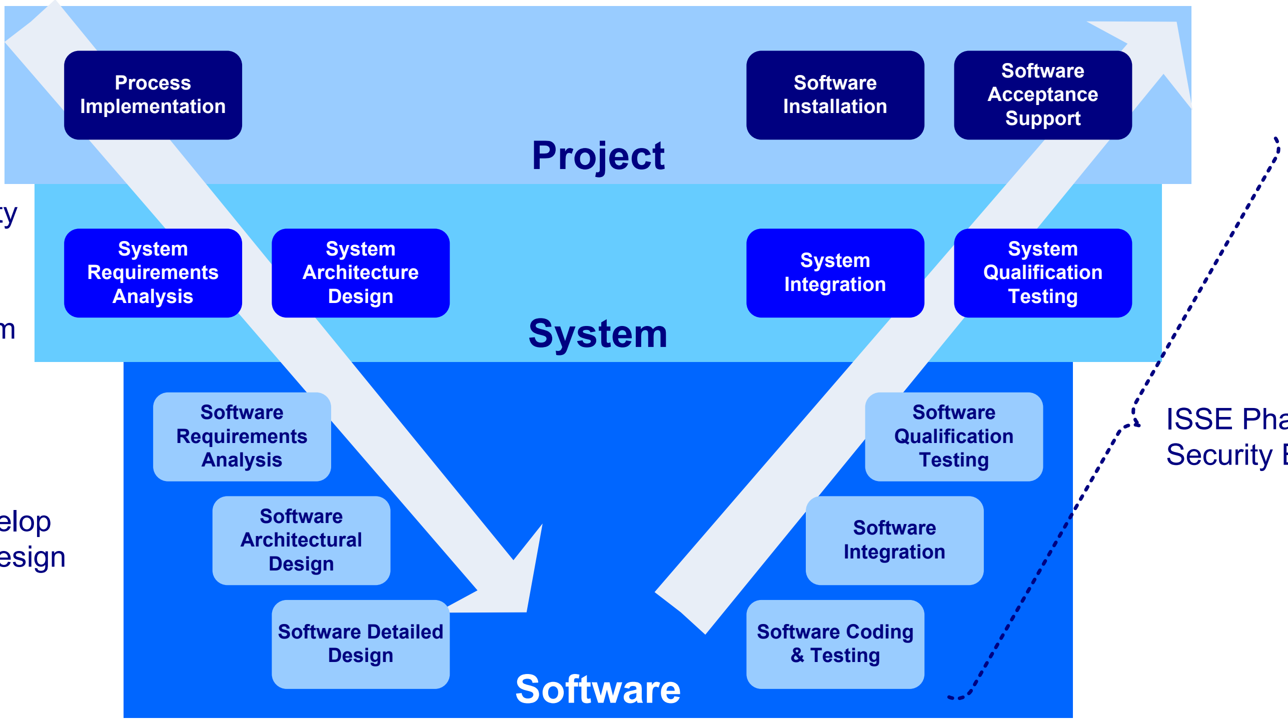
Group 2: DoD-STD-2167A (V-Model)

ISSE Phase 1: Discovering Information Protection Needs

ISSE Phase 2: Define Security Requirements

ISSE Phase 3: Design System Security Architecture

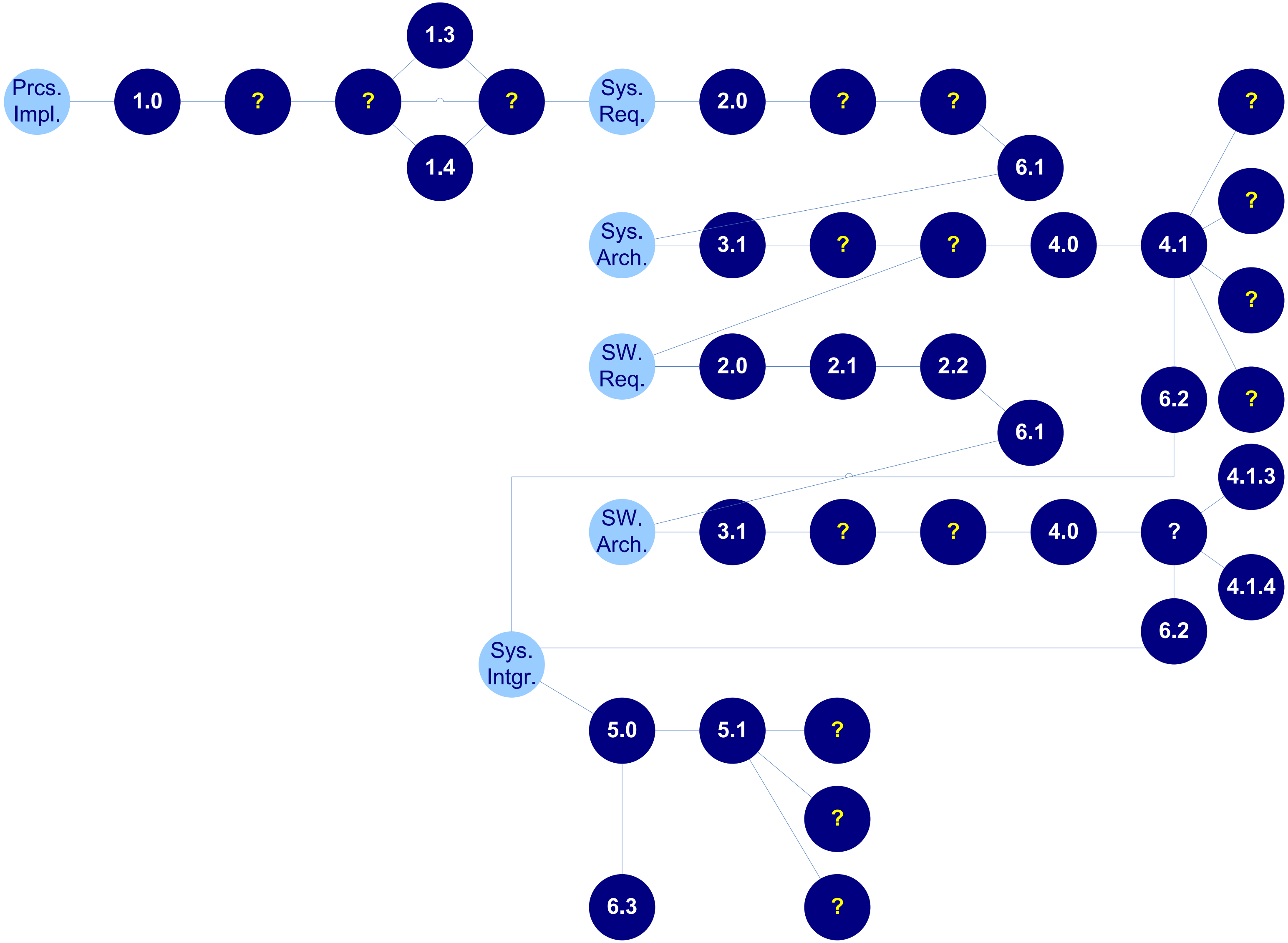
ISSE Phase 4: Develop Detailed Security Design


Process Implementation
Software Installation
Software Acceptance Support
Project
System Requirements Analysis
System Architecture Design
System Integration
System Qualification Testing
System
Software Requirements Analysis
Software Architectural Design
Software Detailed Design
Software Qualification Testing
Software Integration
Software Coding & Testing
Software

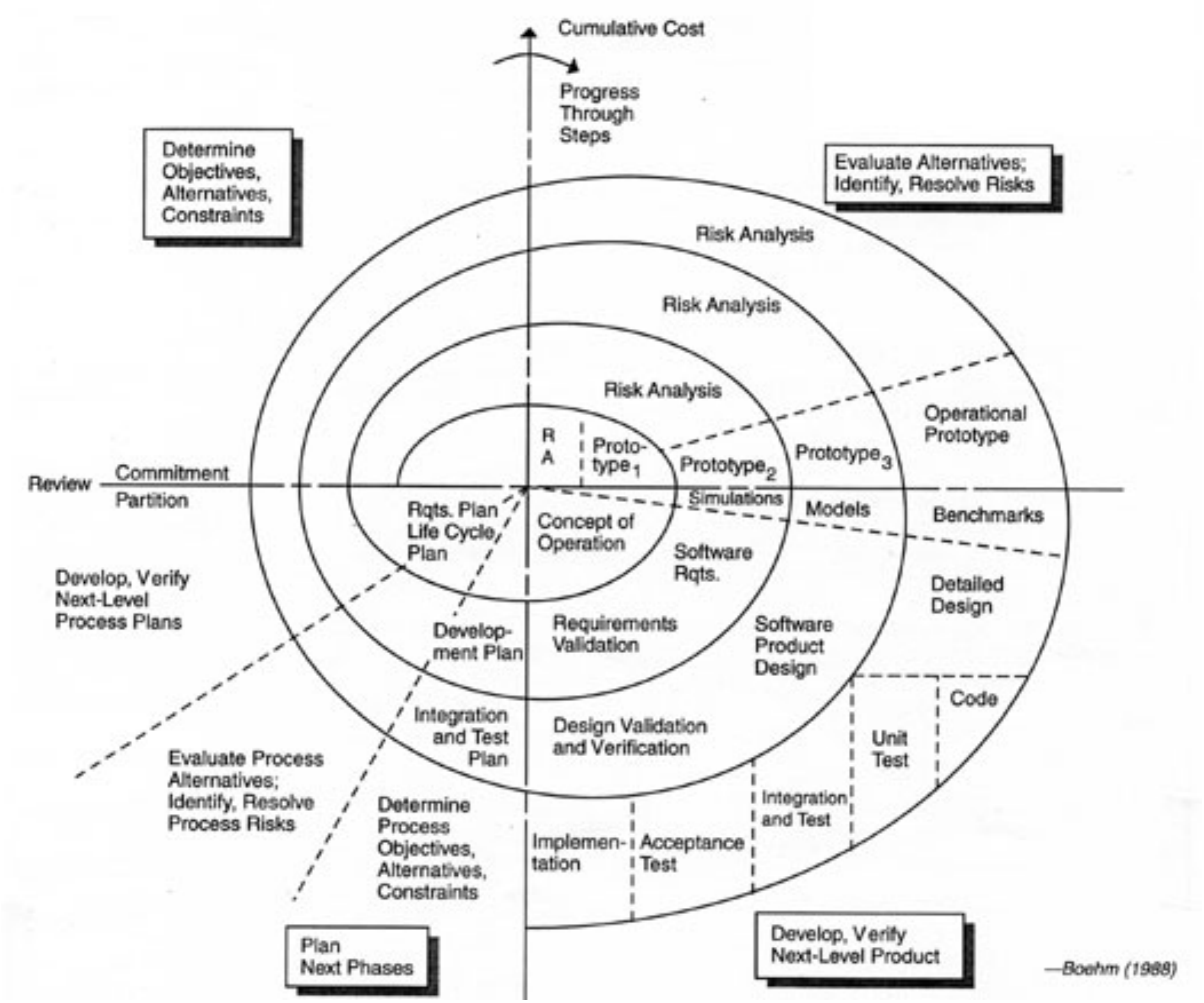
ISSE Phase 6: Assess Security Effectiveness

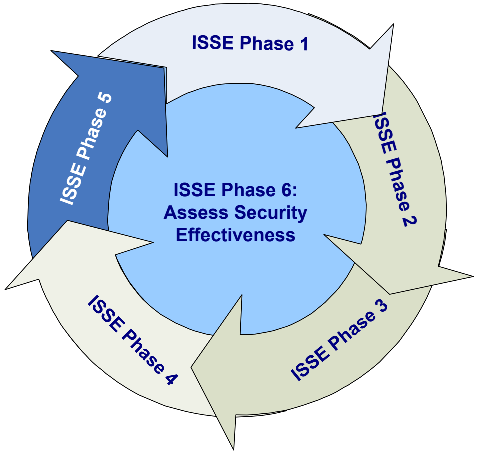
ISSE Phase 5: Implement System Security

Group 2: DoD-STD-2167A (V-Model)

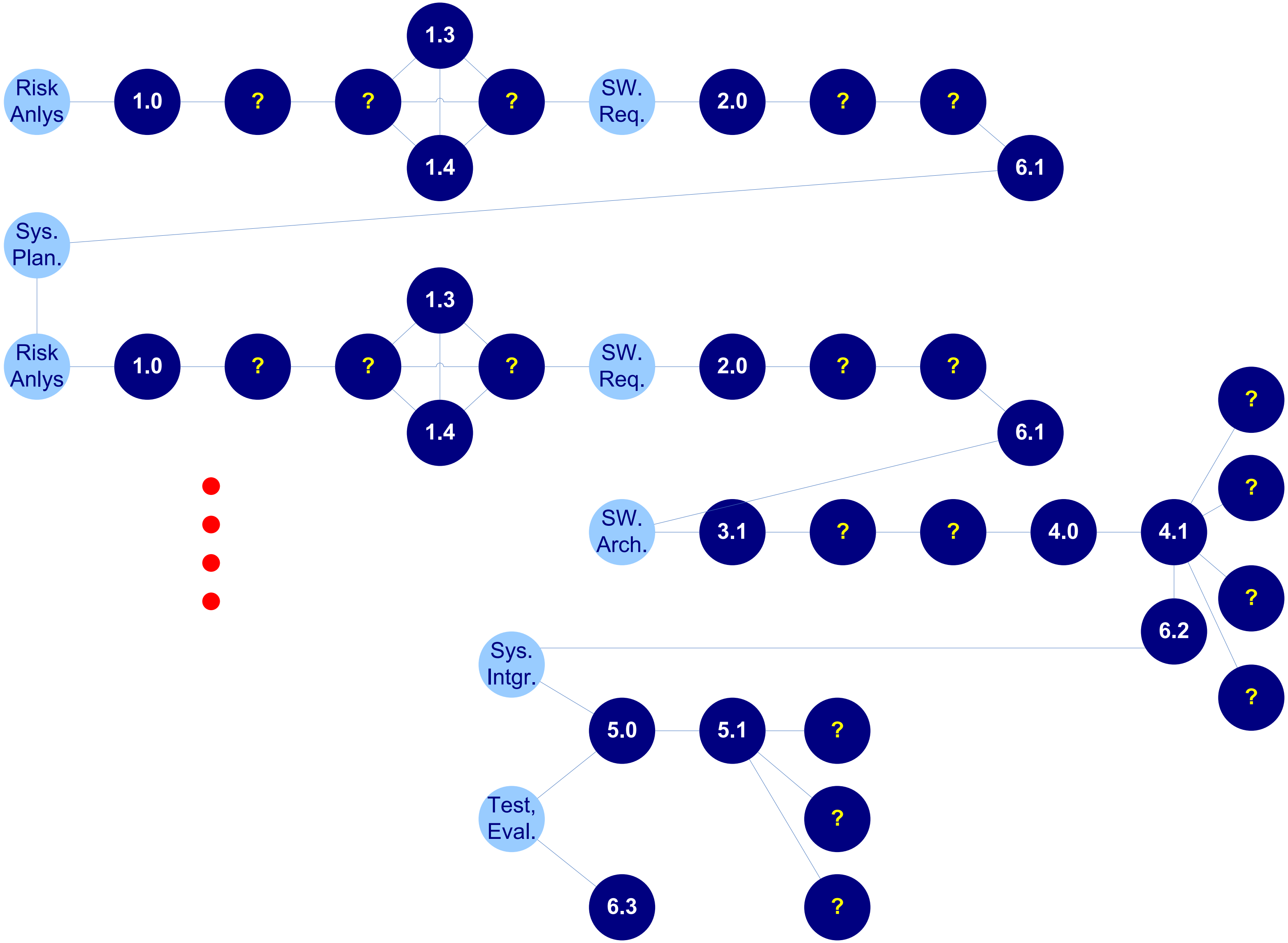


Group 3: Boehm's Spiral SDLC Model





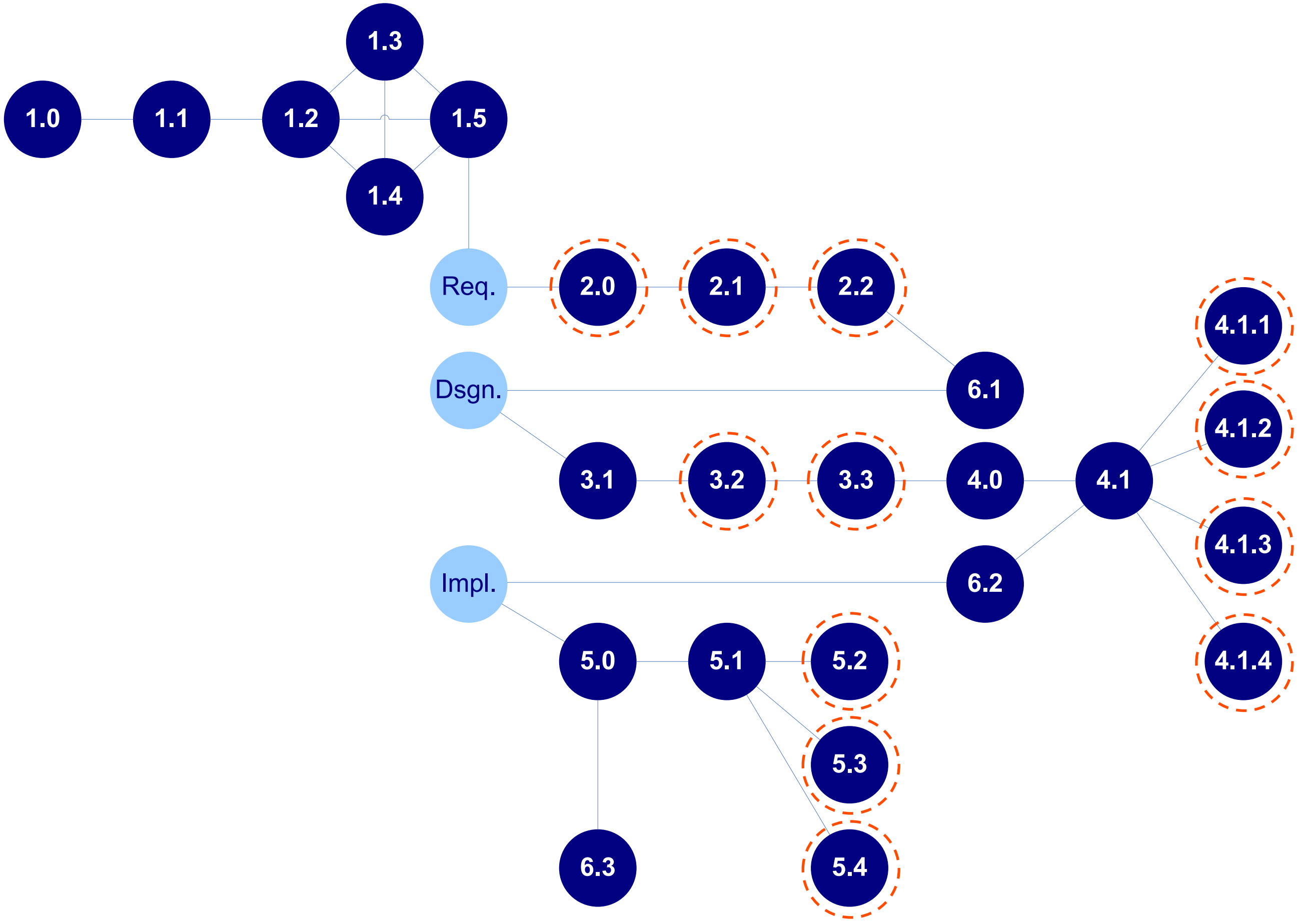
Group 3: Boehm's Spiral SDLC Model



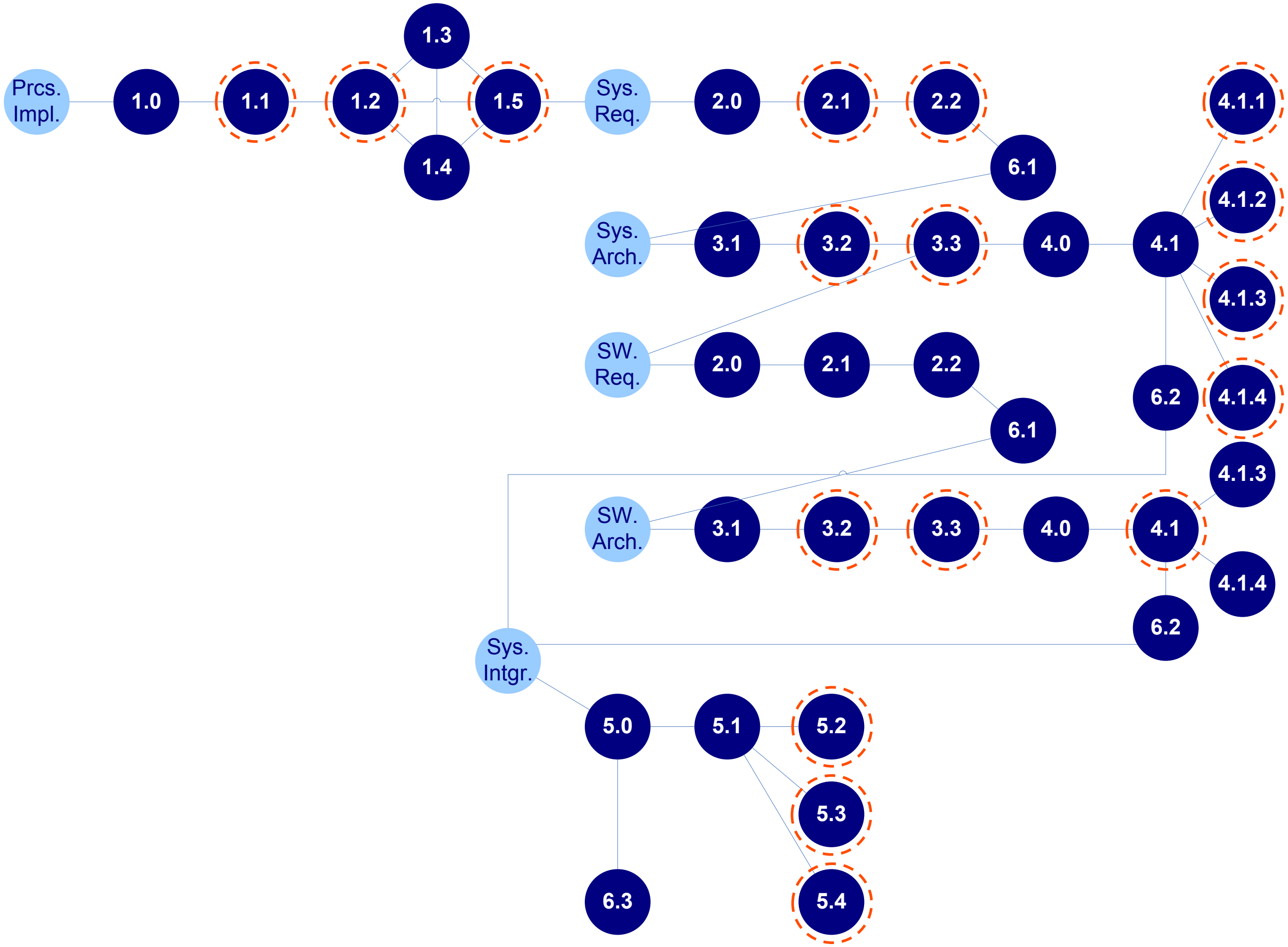
Reference

ANSWERS

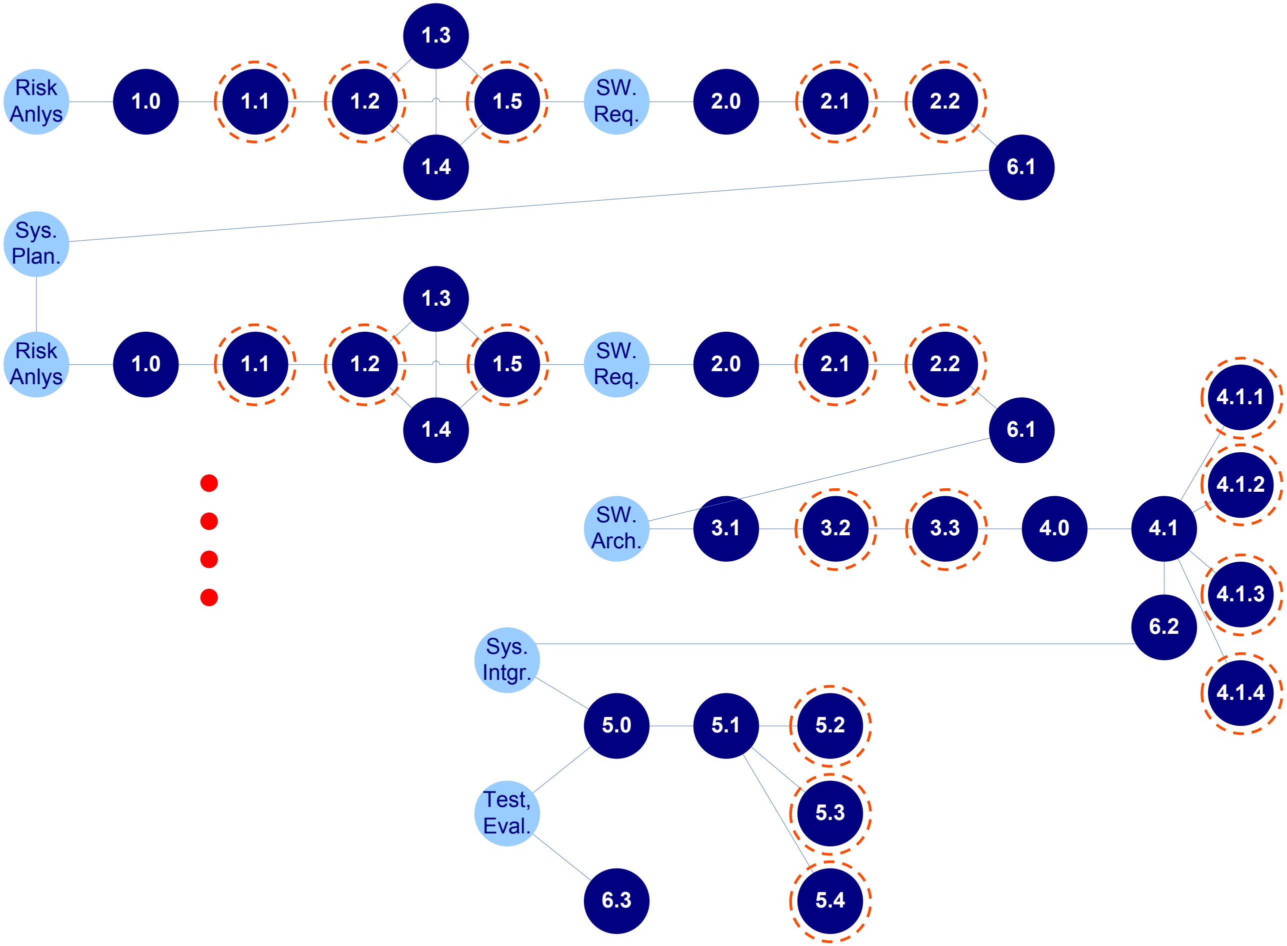
Group 1: Waterfall SDLC Model



Group 2: DoD-STD-2167A (V-Model)



Group 3: Boehm’s Spiral SDLC


Risk Anlys
1.0
1.1
1.2
1.3
1.4
1.5
SW. Req.
2.0
2.1
2.2
6.1
Sys. Plan.
Risk Anlys
1.0
1.1
1.2
1.3
1.4
1.5
SW. Req.
2.0
2.1
2.2
6.1
4.1.1
4.1.2
SW. Arch.
3.1
3.2
3.3
4.0
4.1
4.1.3
6.2
4.1.4
Sys. Intgr.
5.0
5.1
5.2
5.3
Test, Eval.
5.4
6.3