www.pwc.com

Secure Development LifeCycles (SDLC)

Bart De Win

March 2013

SecAppDev 2013

pwc

Bart De Win ?

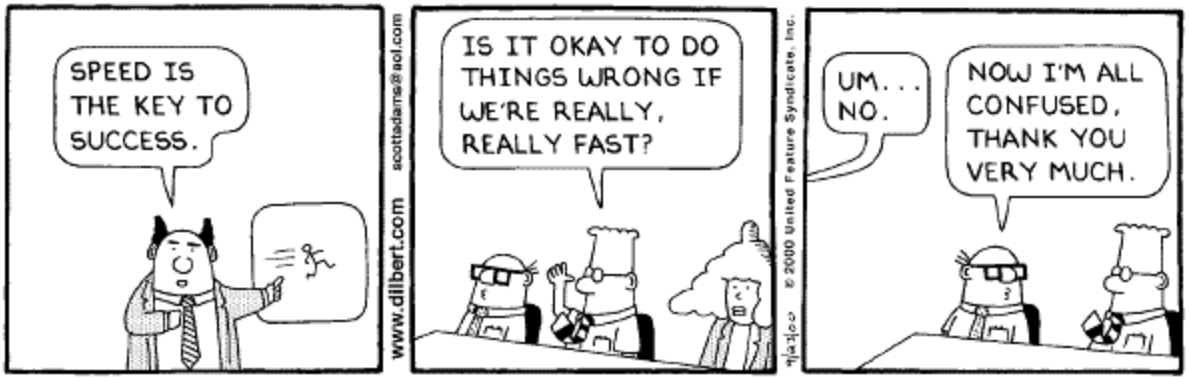
- 15+ years of Information Security Experience
  - Ph.D. in Computer Science - Application Security
- Author of >60 scientific publications
- ISC² CSSLP certified
- Senior Manager @ PwC Belgium:
  - Expertise Center Leader Secure Software
  - (Web) Application tester (pentesting, arch. review, code review, …)
  - Trainer for several courses related to secure software
  - Specialized in Secure Software Development Lifecycle (SDLC)

- Contact me at bart.de.win@pwc.be

Agenda

1. **Motivation**
2. Process Models
3. Maturity Models
4. Implementation: Tips & Challenges
5. Discussion

Application Security Problem



Adaptability

Technology stacks

Software complexity

Training

Better

Growing connectivity

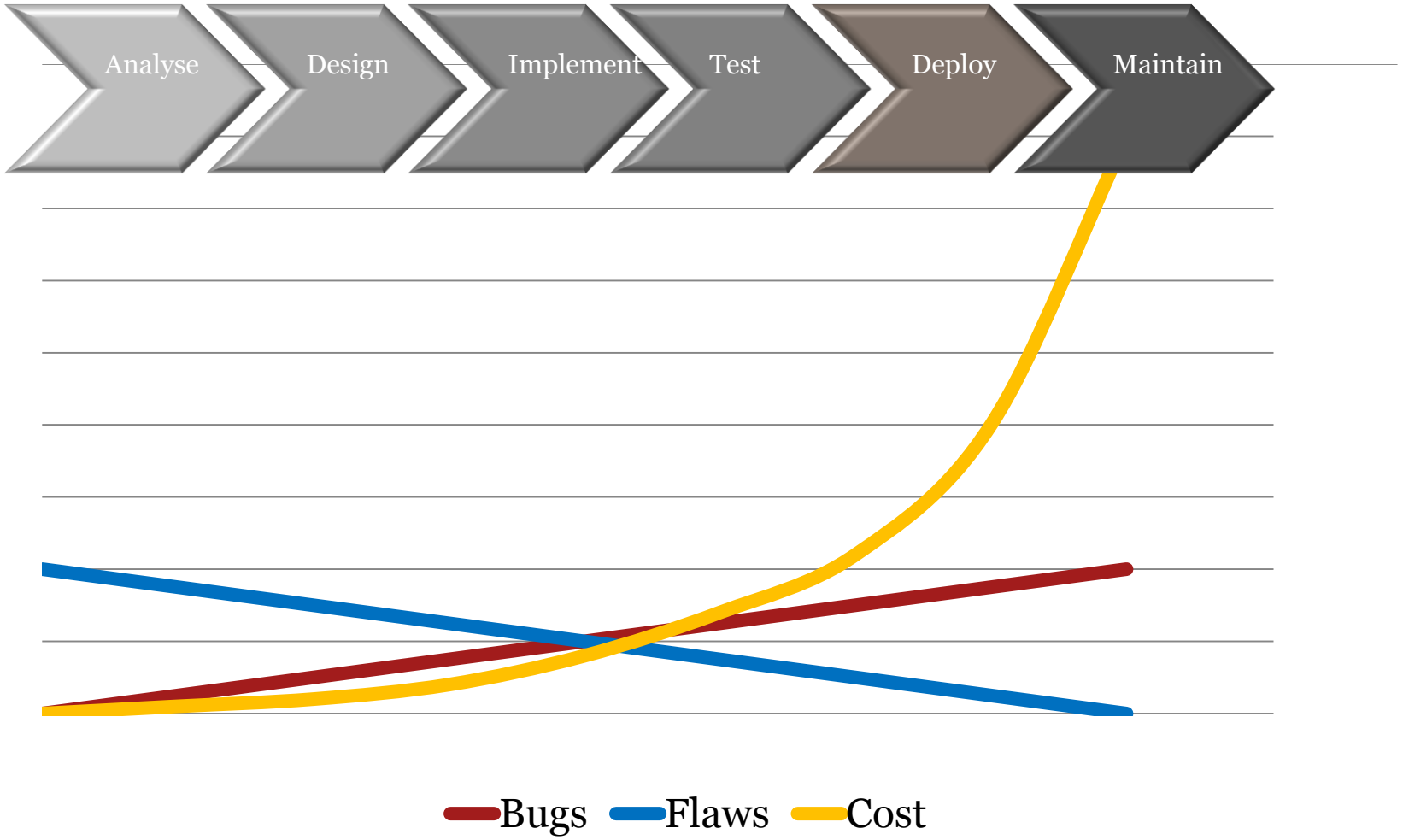
Mobile

Faster

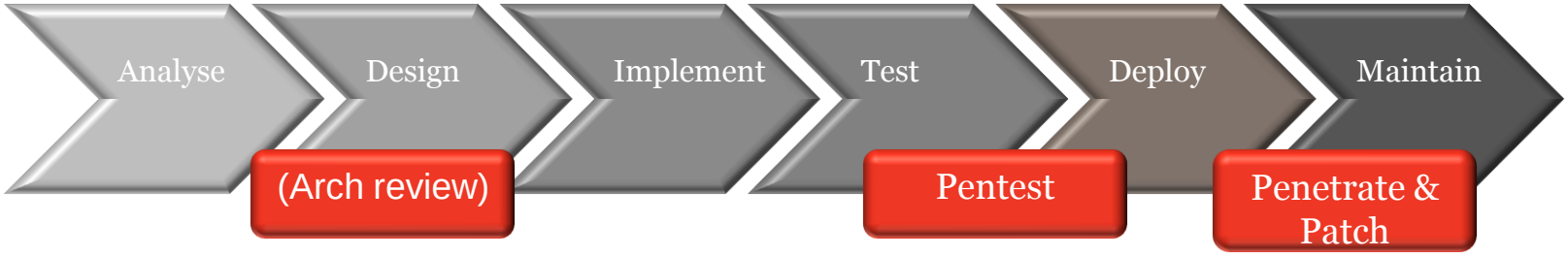
Cloud

75% of vulnerabilities are application related

The Nature of Application Security



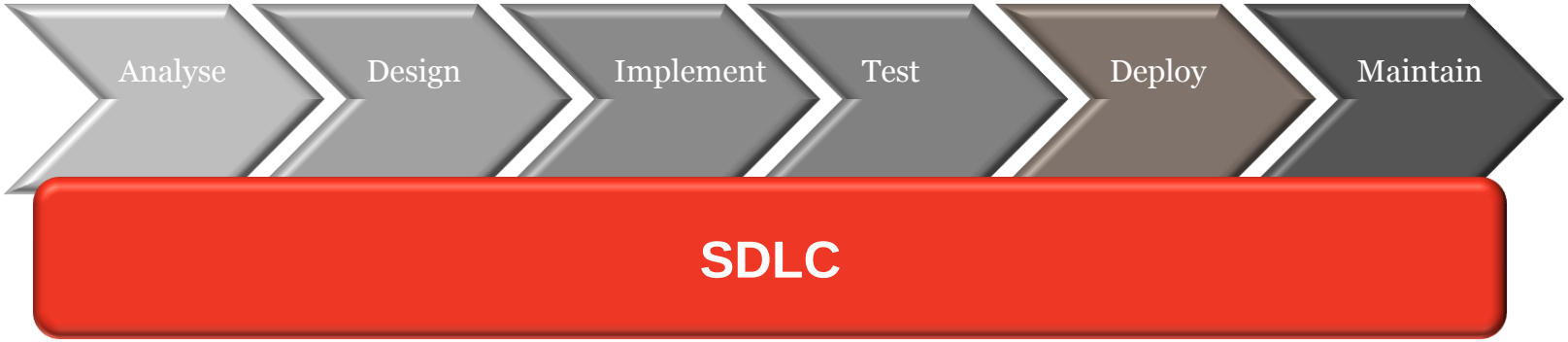
The State-of-Practice in Secure Software Development



Problematic, since:

- Focus on bugs, not flaws
- Penetration can cause major harm
- Not cost efficient
- No security assurance
  - All bugs found ?
  - Bug fix fixes all occurences ? (also future ?)
  - Bug fix might introduce new security vulnerabilities

SDLC ?



Enterprise-wide software security improvement program

- Strategic approach to assure software quality
- Goal is to increase systematicity
- Focus on security functionality and security hygiene

SDLC Objectives

To develop (and maintain) software in a **consistent** and **efficient** way with a **demonstrable** & **standards-compliant security quality**, inline with the organizational **risks**.

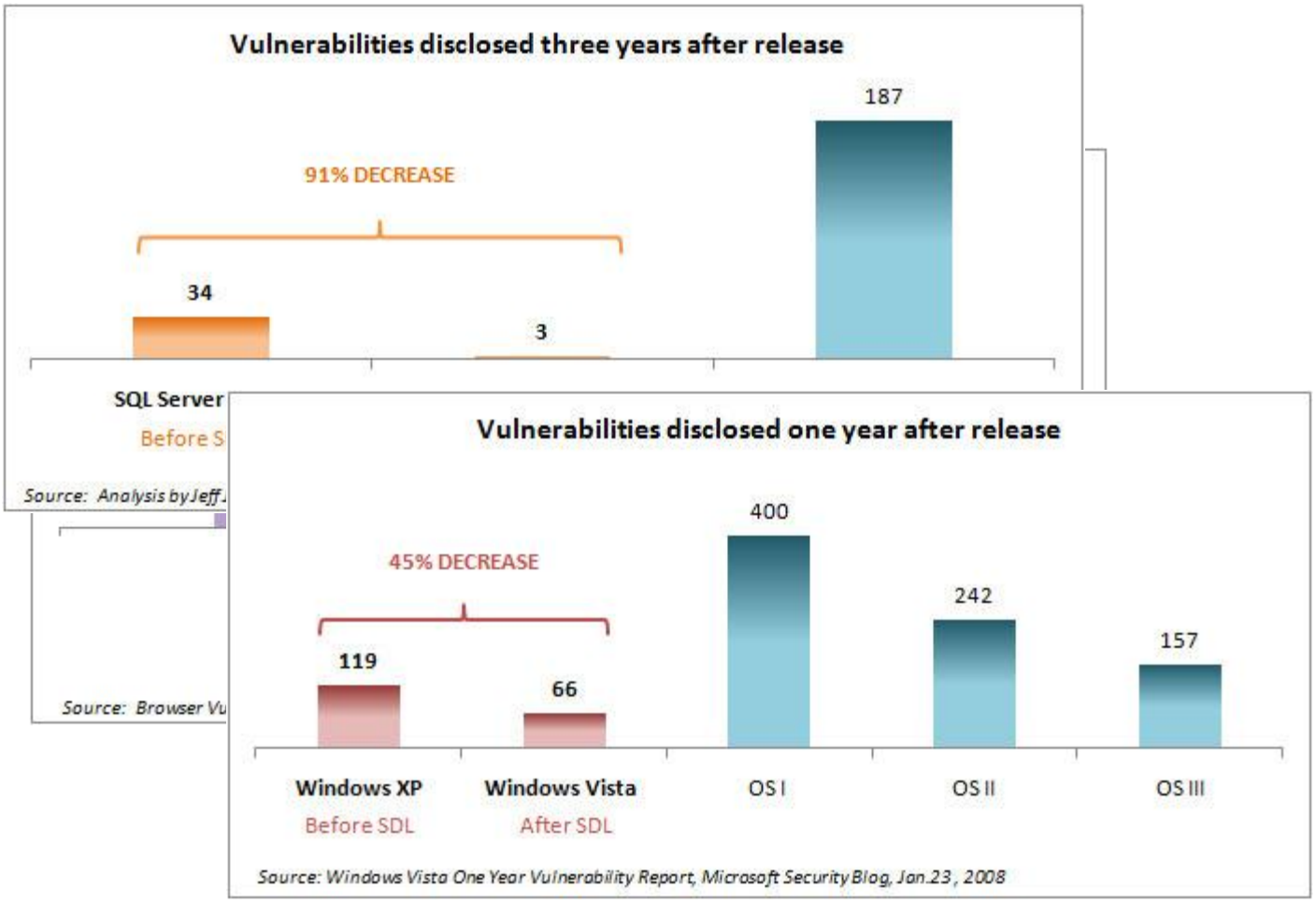
SDLC Cornerstones

Gartner

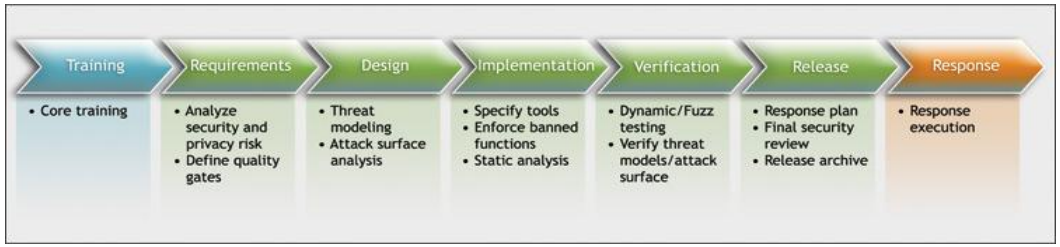
Organizations with a proper SDLC will experience an 80 percent decrease in critical vulnerabilities

Organizations that acquire products and services with just a 50 percent reduction in vulnerabilities will reduce configuration management and incident response costs by 75 percent each.

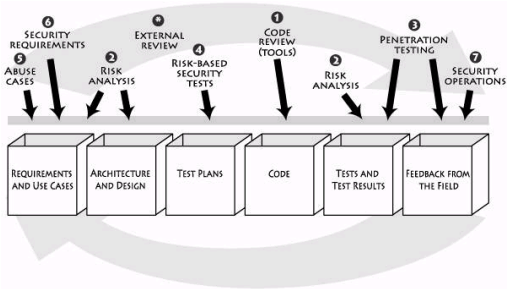
Does it really work ?



SDLC-related initiatives



•Microsoft SDL



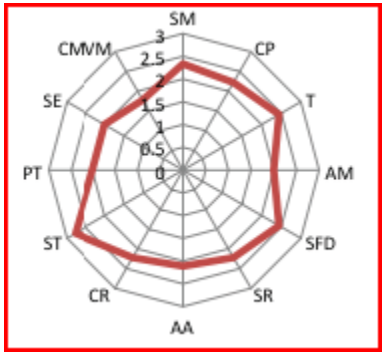
•TouchPoints

•CLASP

NIST
National Institute of Standards and Technology

•SP800-64

SAFECode
Software Assurance Forum for Excellence in Code
Driving Security and Integrity



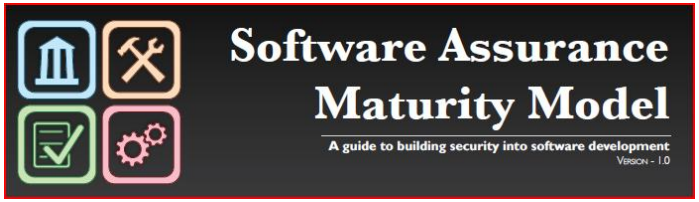
•BSIMM

SSE CMM

•SSE-CMM

Gartner

•TSP-Secure



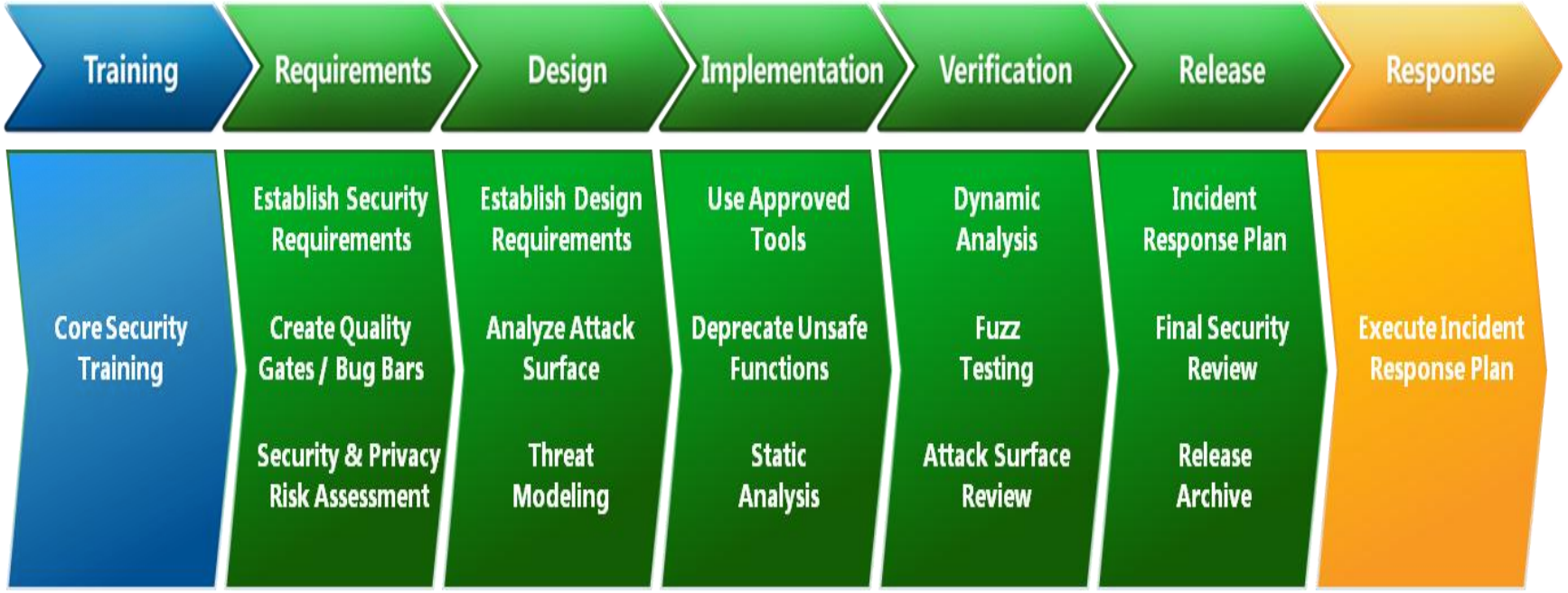
•SAMM

•GASSP

Agenda

1. Motivation
2. **Process Models**
3. Maturity Models
4. Implementation: Tips & Challenges
5. Discussion

Selected Example: Microsoft SDL (SD3+C)



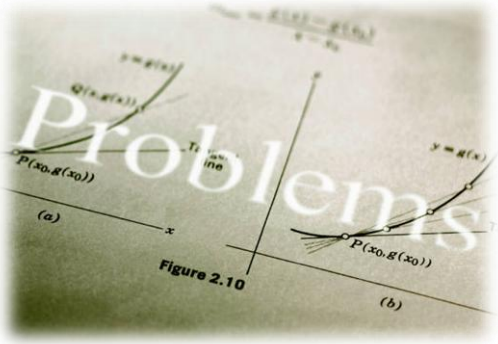
Training

1. **Training**
2. Requirements
3. Design
4. Implementation
5. Verification
6. Release
7. Response

Content

- Secure design
- Threat modeling
- Secure coding
- Security testing
- Privacy

Why?



Requirements

1. Training
2. **Requirements**
3. Design
4. Implementation
5. Verification
6. Release
7. Response

Project inception

When you consider security and privacy at a foundational level

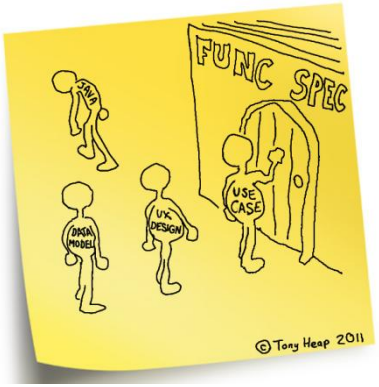
Cost analysis

Determine if development and support costs for improving security and privacy are consistent with business needs

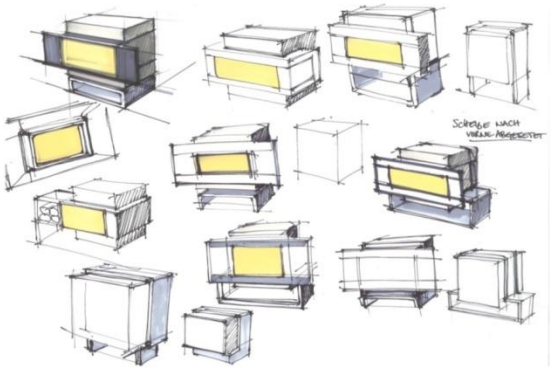
Design

1. Training
2. Requirements
3. **Design**
4. Implementation
5. Verification
6. Release
7. Response

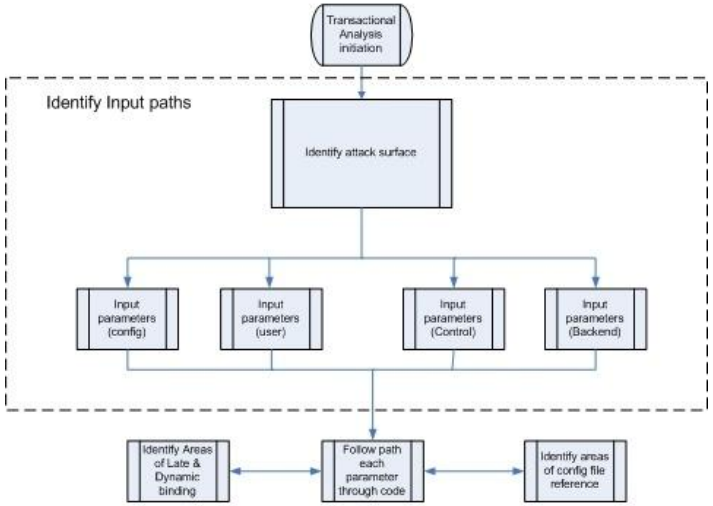
Establish and follow best practices for Design



≠ secure-coding best practices



Risk analysis



Threat modeling

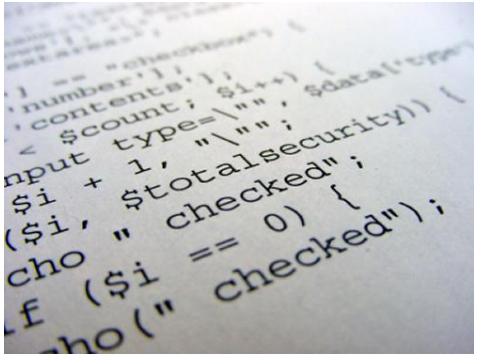
STRIDE

Implementation

1. Training
2. Requirements
3. Design
4. **Implementation**
5. Verification
6. Release
7. Response

Creating documentation and tools for users that address security and privacy

Establish and follow best practices for development

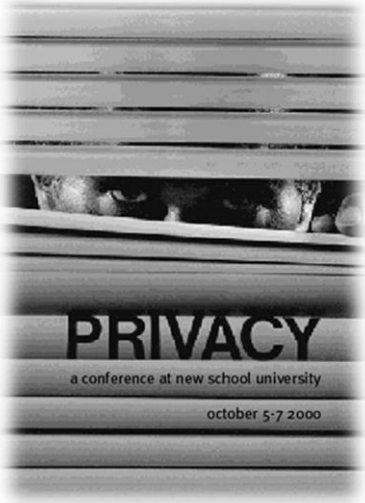


1. Review available information resources
2. Review recommended development tools
3. Define, communicate and document all best practices and policies

Verification

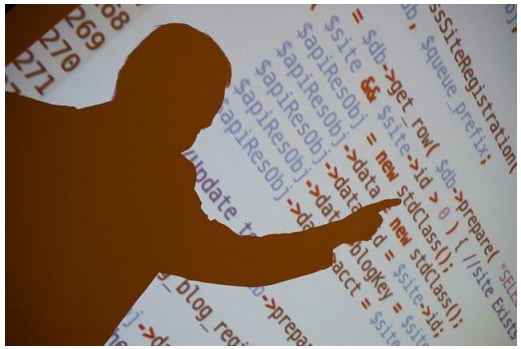
1. Training
2. Requirements
3. Design
4. Implementation
5. **Verification**
6. Release
7. Response

Security and privacy testing



1. Confidentiality, integrity and availability of the software and data processed by the software
2. Freedom from issues that could result in security vulnerabilities

Security push

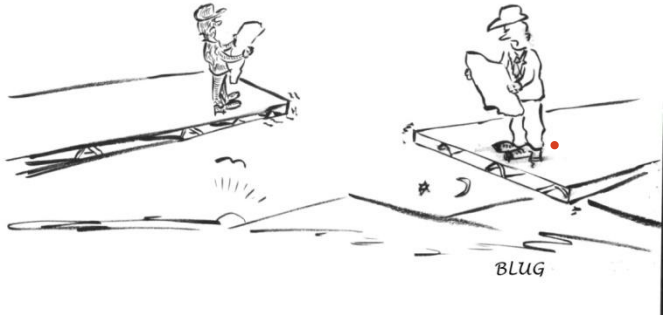


Release

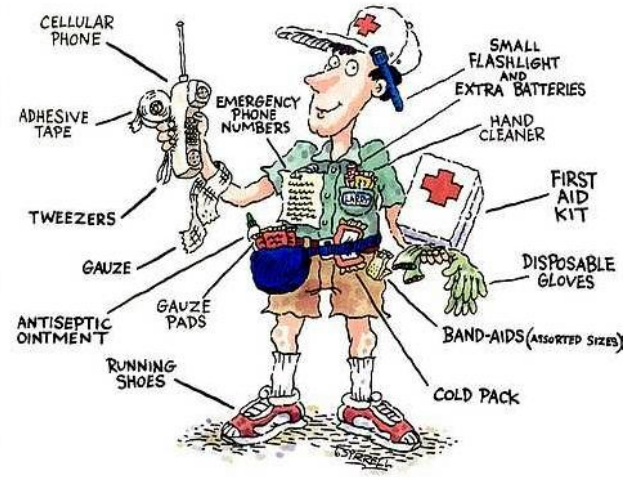
1. Training
2. Requirements
3. Design
4. Implementation
5. Verification
6. **Release**
7. Response

Public pre-release review

1. **Privacy**
2. Security



Planning



Preparation for incident response

Release

1. Training
2. Requirements
3. Design
4. Implementation
5. Verification
6. **Release**
7. Response

Final security and privacy review

Outcomes:

- **Passed FSR**
- **Passed FSR** with exceptions
- **FSR escalation**

Release to manufacturing/release to web

Sign-off process to ensure security, privacy and other policy compliance

Response

1. Training
2. Requirements
3. Design
4. Implementation
5. Verification
6. Release
7. **Response**

Execute Incident Response Plan

=> able to respond appropriately to reports of vulnerabilities in their software products, and to attempted exploitation of those vulnerabilities.

Process Models: wrapup

Microsoft SDL:

- Mature, long-term practical experience
- Heavyweight, ISV flavour
- Several supporting tools and methods

Other process models exist, with their pro's and con's

In general, no process will fit your organization perfectly

Mix-and-Match + adaptation are necessary

Agenda

1. Motivation
2. Process Models
3. **Maturity Models**
4. Implementation: Tips & Challenges
5. Discussion

Why Maturity Models ?

An organization's behavior changes slowly over time.

- Changes must be iterative while working toward long-term goals

There is no single recipe that works for all organizations

- A solution must enable risk-based choices tailor to the organization

Guidance related to security activities must be prescriptive

- A solution must provide enough details for non-security-people

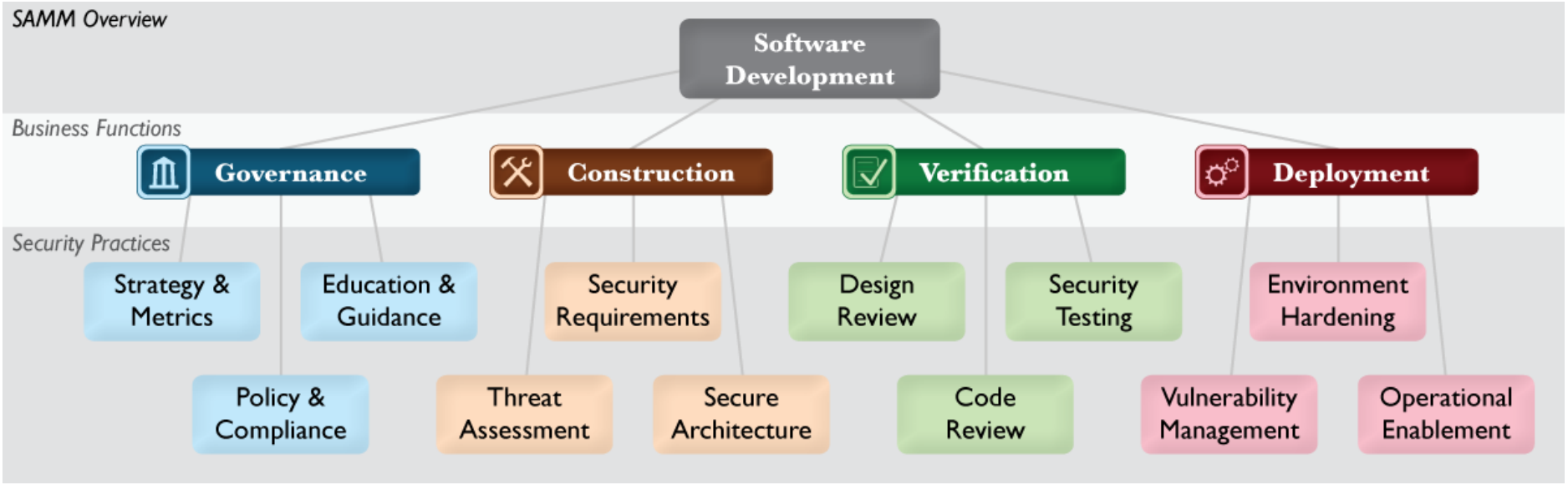
Overall, must be simple, well-defined, and measurable

Selected example: OpenSAMM

http://www.opensamm.org

Version 1.0, 2009

Core Structure



Notion of Maturity

Level	Interpretation
0	Implicit starting point representing the activities in the practice being unfulfilled
1	Initial understanding and ad-hoc provision of the security practice
2	Increase efficiency and/of effectiveness of the security practice
3	Comprehensive mastery of the security practice at scale

An example

	Code Review ...more on page 62		
	CR 1	CR 2	CR 3
OBJECTIVE	**Opportunistically find basic code-level vulnerabilities and other high-risk security issues**	**Make code review during development more accurate and efficient through automation**	**Mandate comprehensive code review process to discover language-level and application-specific risks**
ACTIVITIES	A. Create review checklists from known security requirements B. Perform point-review of high-risk code	A. Utilize automated code analysis tools B. Integrate code analysis into development process	A. Customize code analysis for application-specific concerns B. Establish release gates for code review

OpenSAMM also defines

- Objective
- Activities
- Results
- Success Metrics
- Costs
- Personnel
- Related Levels

Security Testing

ST 3

Require application-specific security testing to ensure baseline security before deployment

Activities

A. Employ application-specific security testing automation

Through either customization of security testing tools, enhancements to generic test case execution tools, or buildout of custom test harnesses, project teams should formally iterate through security requirements and build a set of automated checkers to test the security of the implemented business logic.

Additionally, many automated security testing tools can be greatly improved in accuracy and depth of coverage if they are customized to understand more detail about the specific software interfaces in the project under test. Further, organization-specific concerns from compliance or technical standards can be codified as a reusable, central test battery to make audit data collection and per-project management visibility simpler.

Project teams should focus on buildout of granular security test cases based on the business functionality of their software, and an organization-level team led by a security auditor should focus on specification of automated tests for compliance and internal standards.

B. Establish release gates for security testing

To prevent software from being released with easily found security bugs, a particular point in the software development life-cycle should be identified as a checkpoint where an established set of security test cases must pass in order to make a release from the project. This establishes a baseline for the kinds of security tests all projects are expected to pass.

Since adding too many test cases initially can result in an overhead cost bubble, begin by choosing one or two security issues and include a wide variety of test cases for each with the expectation that no project may pass if any test fails. Over time, this baseline should be improved by selecting additional security issues and adding a variety of corresponding test cases.

Generally, this security testing checkpoint should occur toward the end of the implementation or testing, but must occur before release.

For legacy systems or inactive projects, an exception process should be created to allow those projects to continue operations, but with an explicitly assigned timeframe for mitigation of findings. Exceptions should be limited to no more that 20% of all projects.

Results

- Organization-wide baseline for expected application performance against attacks
- Customized security test suites to improve accuracy of automated analysis
- Project teams aware of objective goals for attack resistance

Add'l Success Metrics

- >50% of projects using security testing customizations
- >75% of projects passing all security tests in past 6 months

Add'l Costs

- Buildout and maintenance of customizations to security testing automation
- Ongoing project overhead from security testing audit process
- Organization overhead from project delays caused by failed security testing audits

Add'l Personnel

- Architects (1 day/yr)
- Developers (1 day/yr)
- Security Auditors (1-2 days/yr)
- QA Testers (1-2 days/yr)
- Business Owners (1 day/yr)
- Managers (1 day/yr)

Related Levels

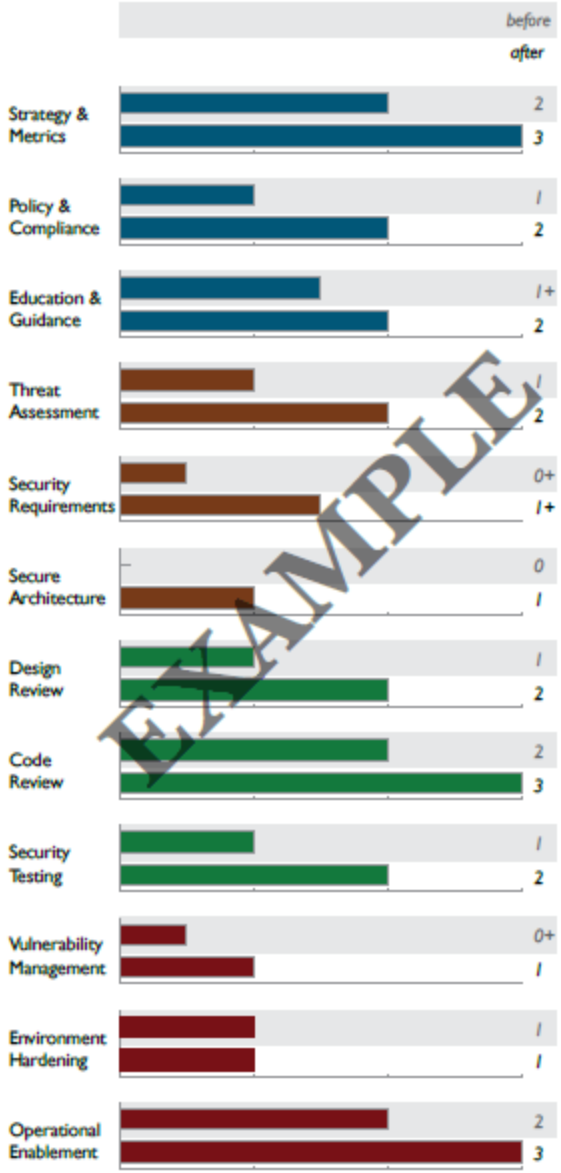
- Policy & Compliance - 2
- Secure Architecture - 3

Practices - v1.0

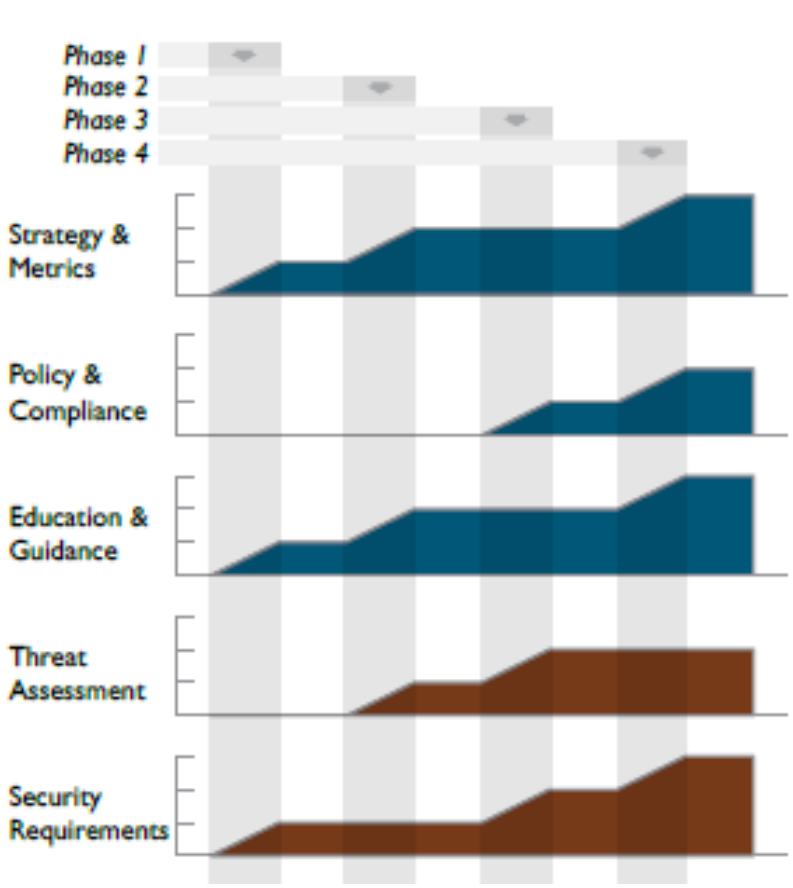
Assessments

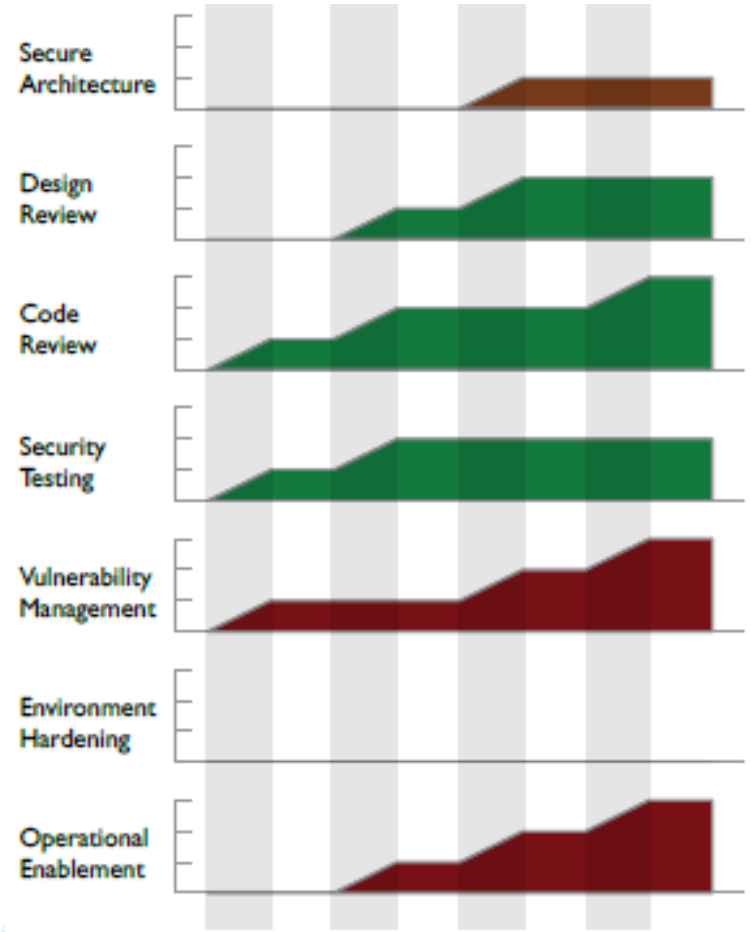
Secure Architecture

	Yes/No
✦Are project teams provided with a list of recommended third-party components?	
✦Are most project teams aware of secure design principles and applying them?	
✦Do you advertise shared security services with guidance for project teams?	
✦Are project teams provided with prescriptive design patterns based on their application architecture?	
✦Are project teams building software from centrally controlled platforms and frameworks?	
✦Are project teams being audited for usage of secure architecture components?	



Roadmap templates per company type (ISV)





BSIMM4 statistics: summary

BSIMM4 statistics: per activity (TODO)

Governance		Intelligence		SSDL Touchpoints		Deployment	
Activity	Observed	Activity	Observed	Activity	Observed	Activity	Observed
[SM1.1]	35	[AM1.1]	15	[AA1.1]	39	[PT1.1]	47
[SM1.2]	30	[AM1.2]	31	[AA1.2]	35	[PT1.2]	40
[SM1.3]	33	[AM1.3]	25	[AA1.3]	27	[PT1.3]	35
[SM1.4]	44	[AM1.4]	13	[AA1.4]	32	[PT2.2]	20
[SM1.6]	35	[AM1.5]	32	[AA2.1]	10	[PT2.3]	24
[SM2.1]	21	[AM1.6]	17	[AA2.2]	7	[PT3.1]	11
[SM2.2]	26	[AM2.1]	12	[AA2.3]	17	[PT3.2]	8
[SM2.3]	26	[AM2.2]	13	[AA3.1]	9		
[SM2.5]	22	[AM3.1]	3	[AA3.2]	4		
[SM3.1]	15	[AM3.2]	5				
[SM3.2]	6						
[CP1.1]	40	[SFD1.1]	44	[CR1.1]	23	[SE1.1]	21
[CP1.2]	45	[SFD1.2]	37	[CR1.2]	20	[SE1.2]	47
[CP1.3]	36	[SFD2.1]	25	[CR1.4]	33	[SE2.2]	21
[CP2.1]	21	[SFD2.2]	19	[CR1.5]	22	[SE2.4]	23
[CP2.2]	20	[SFD2.3]	15	[CR1.6]	21	[SE3.2]	11
[CP2.3]	25	[SFD3.1]	8	[CR2.2]	13	[SE3.3]	7
[CP2.4]	22	[SFD3.2]	9	[CR2.5]	12		
[CP2.5]	31			[CR3.1]	13		
[CP3.1]	7			[CR3.2]	3		
[CP3.2]	12			[CR3.3]	4		
[CP3.3]	8			[CR3.4]	0		
[T1.1]	38	[SR1.1]	38	[ST1.1]	38	[CMVM1.1]	40
[T1.5]	19	[SR1.2]	27	[ST1.3]	37	[CMVM1.2]	44
[T1.6]	21	[SR1.3]	34	[ST2.1]	24	[CMVM2.1]	37
[T1.7]	23	[SR1.4]	21	[ST2.2]	8	[CMVM2.2]	31
[T2.5]	10	[SR2.1]	12	[ST2.4]	12	[CMVM2.3]	23
[T2.6]	12	[SR2.2]	20	[ST3.1]	9	[CMVM3.1]	5
[T2.7]	11	[SR2.3]	18	[ST3.2]	11	[CMVM3.2]	6
[T3.1]	5	[SR2.4]	19	[ST3.3]	5	[CMVM3.3]	0
[T3.2]	5	[SR2.5]	21	[ST3.4]	6		
[T3.3]	8	[SR3.1]	8				
[T3.4]	6						
[T3.5]	6						

Maturity Models wrapup

OpenSAMM

- Comprehensive and rich model, more than just activities
- Supporting tools are available
- Real-world case studies, but few are openly shared

Other models exist with their pro's and con's

Maturity models provide an excellent framework for reasoning on software assurance, on a *strategic* level.

Agenda

1. Motivation
2. Process Models
3. Maturity Models
4. **Implementation: Tips & Challenges**
5. Discussion

Before you begin

Organizational Context

Realistic Goals ?

Scope ?

Constraints (budget, timing, resources)

Affinity with a particular model ?

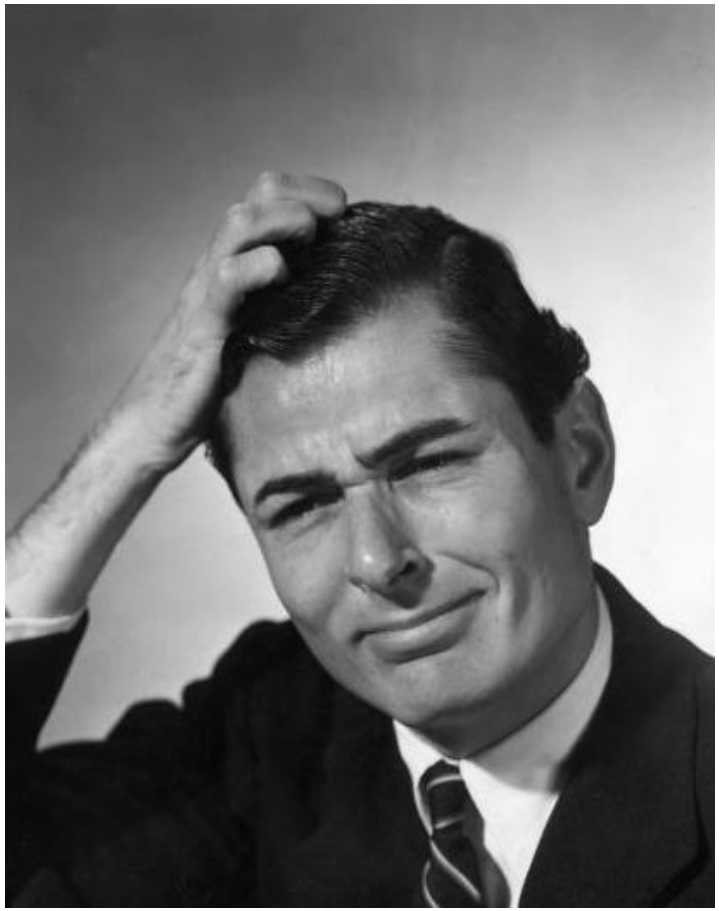
What's your Company Maturity ?

- In terms of IT **strategy** and application **landscape**
- In terms of software **Development** practices
  - Analysis, Design, Implementation, Testing, Release, Maintenance
- In terms of **ITSM** practices
  - Configuration, Change, Release, Vulnerability -Mngt.

Company Maturity ≈ **Feasibility SDLC Program**

Complicating factors, anyone ?

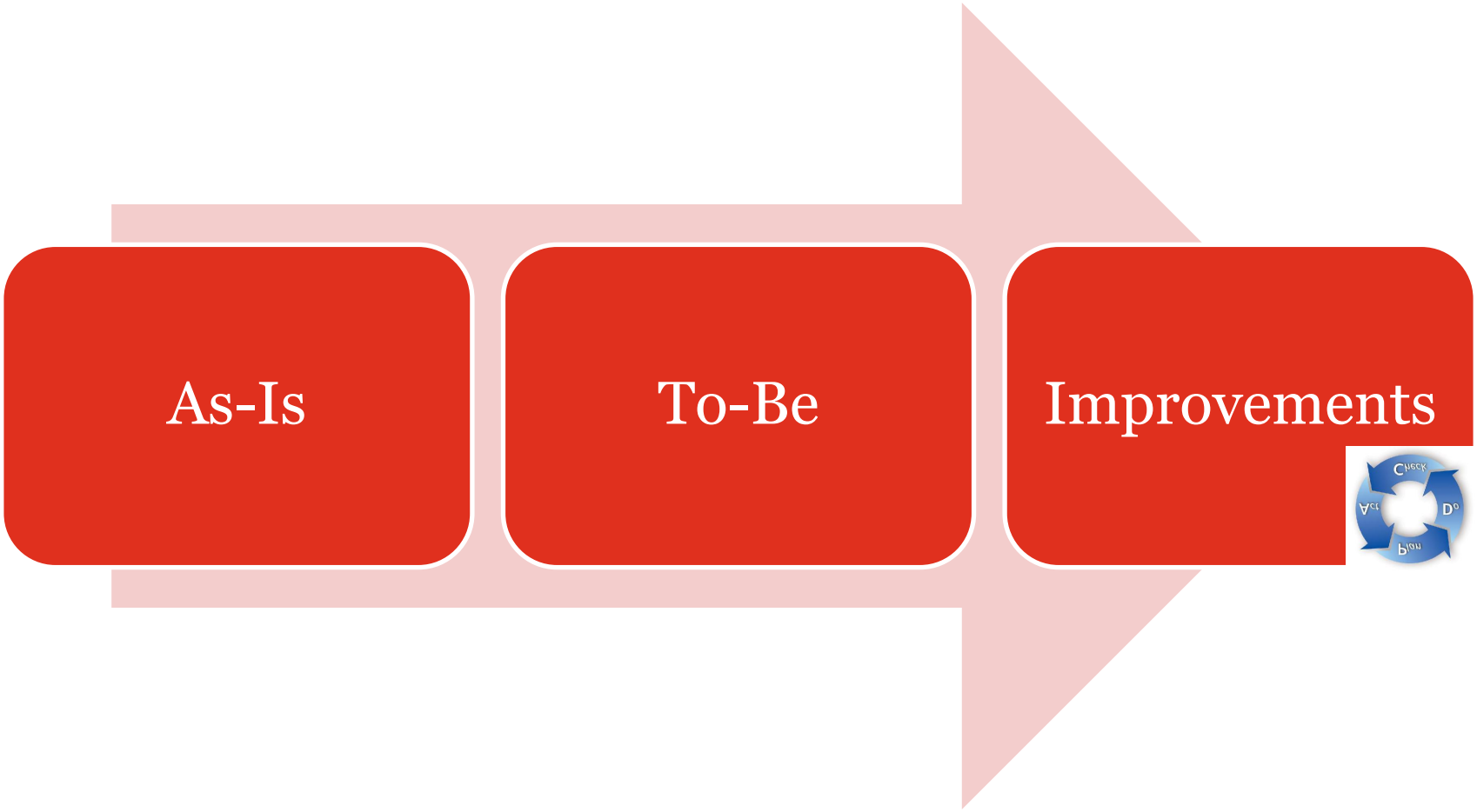
- Different development teams
- Different technology stacks
- Business-IT alignment issues
- Outsourced development
- …



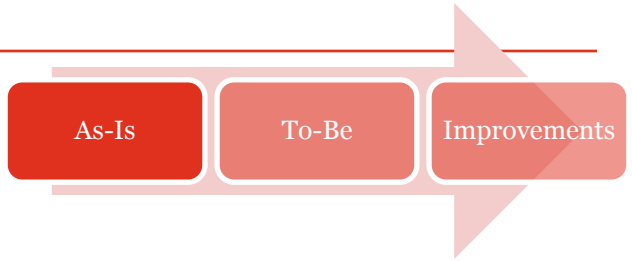
Common SDLC strategies

Enterprise-wide	• Focus on overall methods and practices • Fundamental approach
Project-specific	• Focus on 1 particular project • Targeted approach
Problem-specific	• Focus on 1 specific problem • Ad-hoc approach

Typical Approach



As-Is



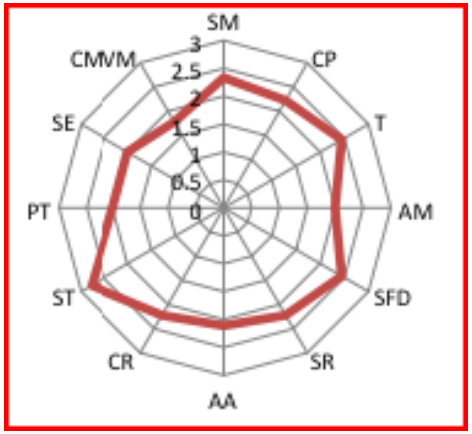
Maturity Evaluation (in your favorite model)

Depending on (your knowledge of) the organisation, you might be able to do this on your own

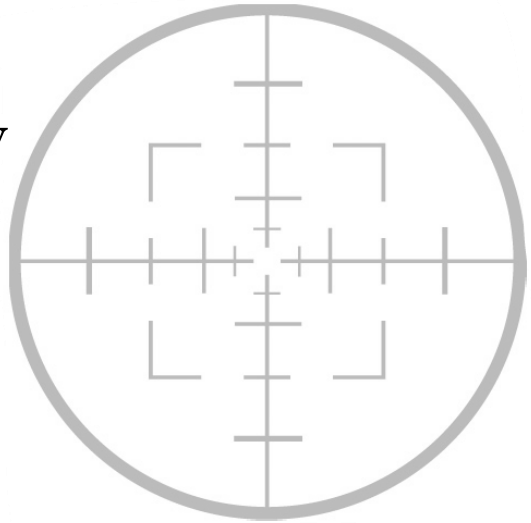
If not, interviews with different stakeholders will be necessary

Analyst, Architect, Tech Lead, QA, Ops, Governance

Discuss outcome with the stakeholders and present findings to the project advisory board



Scoping



For large companies, teams will perform differently

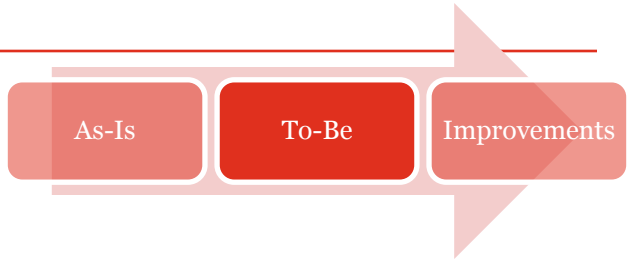
=> difficult to come up with a single result

Consider

- Reducing the scope to a single, uniform unit
- splitting the assessment into different organizational subunits

Splitting might be awkward at first, but can be helpful later on for motivational purposes

To-Be



Identify the targets for your company

Define staged roadmap and overall planning

Define application migration strategy

Gradual improvements work better than big bang

Have this validated by the project advisory board

Staged Roadmap

Security Practices/Phase	Start	One	Two	Three
Strategy & metrics	0,5	2	2	2
Policy & Compliance	0	0,5	1	1,5
Education & Guidance	0,5	1	2	2,5
Threat Assessment	0	0,5	2	2,5
Security Requirements	0,5	1,5	2	3
Secure Architecture	0,5	1,5	2	3
Design Review	0	1	2	2,5
Code Review	0	0,5	1,5	2,5
Security Testing	0,5	1	1,5	2,5
Vulnerability Management	2,5	3	3	3
Environment Hardening	2,5	2,5	2,5	2,5
Operational Enablement	0,5	0,5	1,5	3
Total Effort per Phase		7,5	7,5	7,5

Implementation

Implementation of dedicated activities according to the plan

Iterative, Continuous Process

Leverage good existing practices

Entry Points

- Pick the weak spots that can demonstrate short-term ROI

- Typical examples
  - Awareness training
  - Coding Guidelines
  - External Pentesting

- Success will help you in continuing your effort

Application categorization

Granularity !

Inter-
Connectivity !

Use this to rationalize security effort (according to the application risk)

Communication & Support

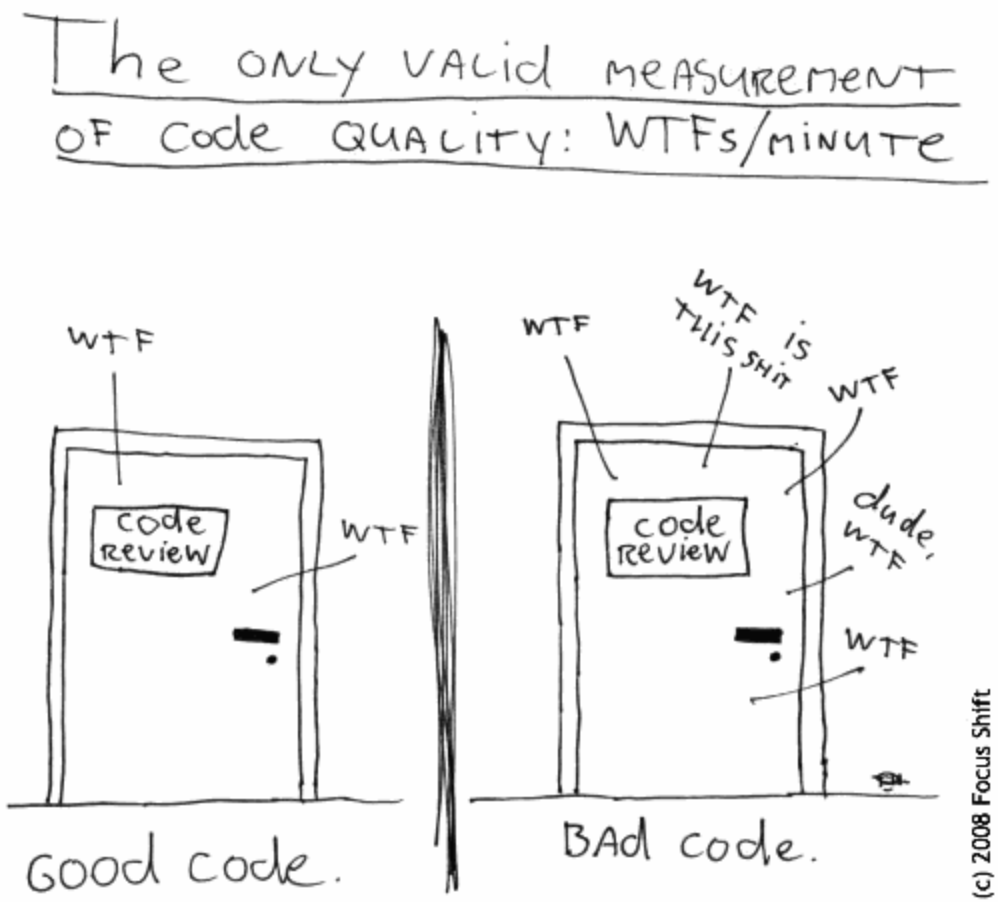
Critical success factor !

Spreading the message – broad audience

Setup a secure applications portal !

Regular status updates towards management

Monitoring & Metrics



Responsabilties

Core Security team

Security Sattellite

- Analysts
- **Architects**
- **Developers**
- Operations
- Management

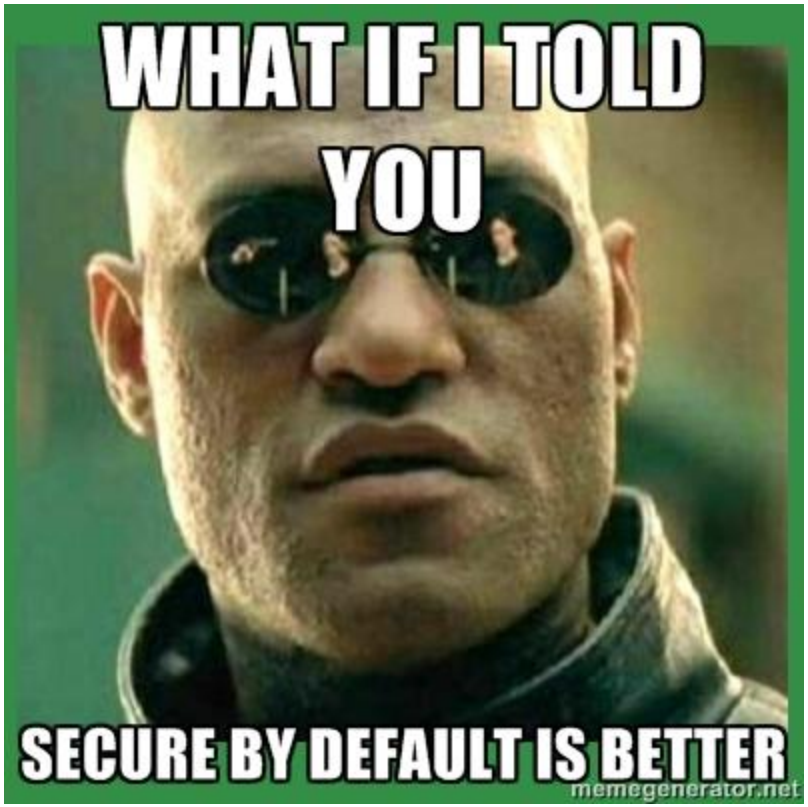
Formalized RACI will be a challenge

The Power of Default Security

Construct development frameworks that are secure by default

Minimizes work for developers

Will lower number of vulns.



What about Agile Development ?

Security improvements must be aligned to the company practices.

Security typically better aligns to waterfall-like processes, however can be used in agile methods as well

- Organisation of activities is different
- Setup of activities needs to be adapted to the techniques used in the concrete process (e.g., abuser stories for threat modelling)

SDLC Cornerstones (revisited)

Agenda

1. Motivation
2. Process Models
3. Maturity Models
4. Implementation: Tips & Challenges
5. **Discussion**

Discussion Topics

Practical experiences

Agile

Mobile

…

Conclusions

SDLC is the framework for most of this week's sessions

No model is perfect, but they provide good guidance

Find balance for all cornerstones

Risk Management is key for rationalizing effort

Beware the big bang