Microsoft SDL in practice

Alex Thissen
Principal Architect, Achmea

alex.thissen@achmea.nl @alexthissen

achmea

Alex Thissen

- Architect with a focus on Microsoft technologies and products
  - Security
  - Competencies
- Trainer/coach in software development
- Regional Director for The Netherlands
- Most Valuable Professional
  for Visual C#

Microsoft Regional Director PROGRAM

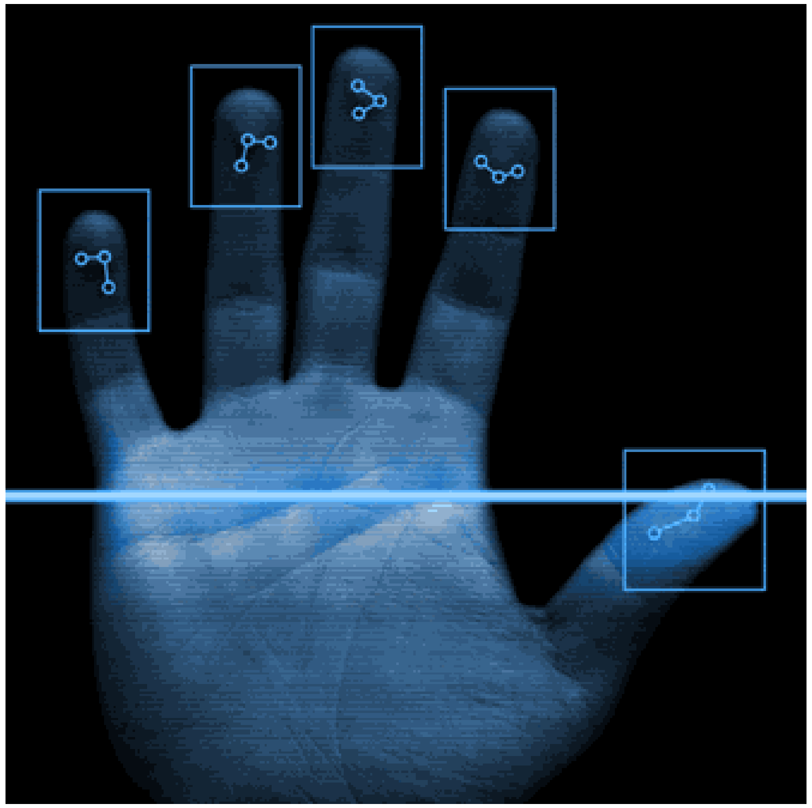
MVP Microsoft Most Valuable Professional

Agenda

- Overview of Microsoft SDL
- Phases of SDL
- Implementing SDL at Achmea
- Lessons learned
- Questions and answers

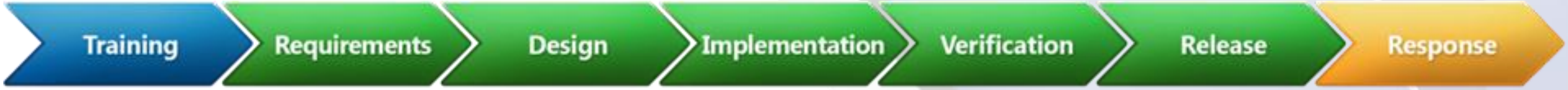
Think security

- Force yourself to pay attention to security during application development
- Security is often first victim

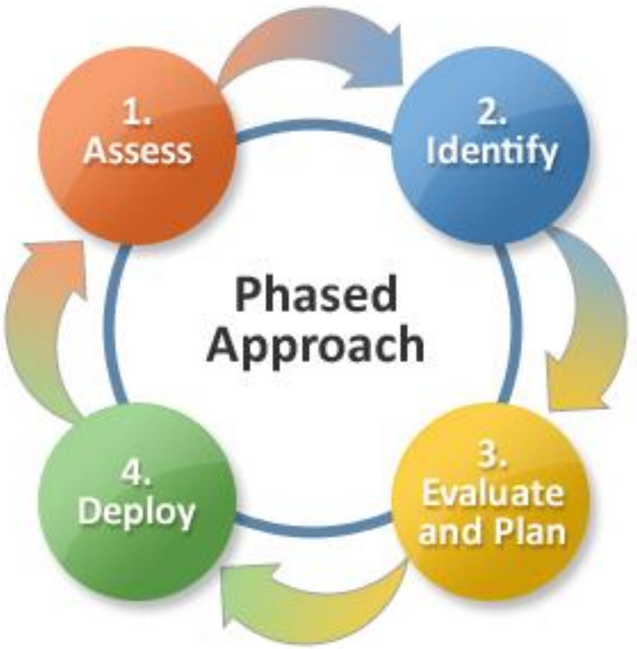


Microsoft® Security Development Lifecycle

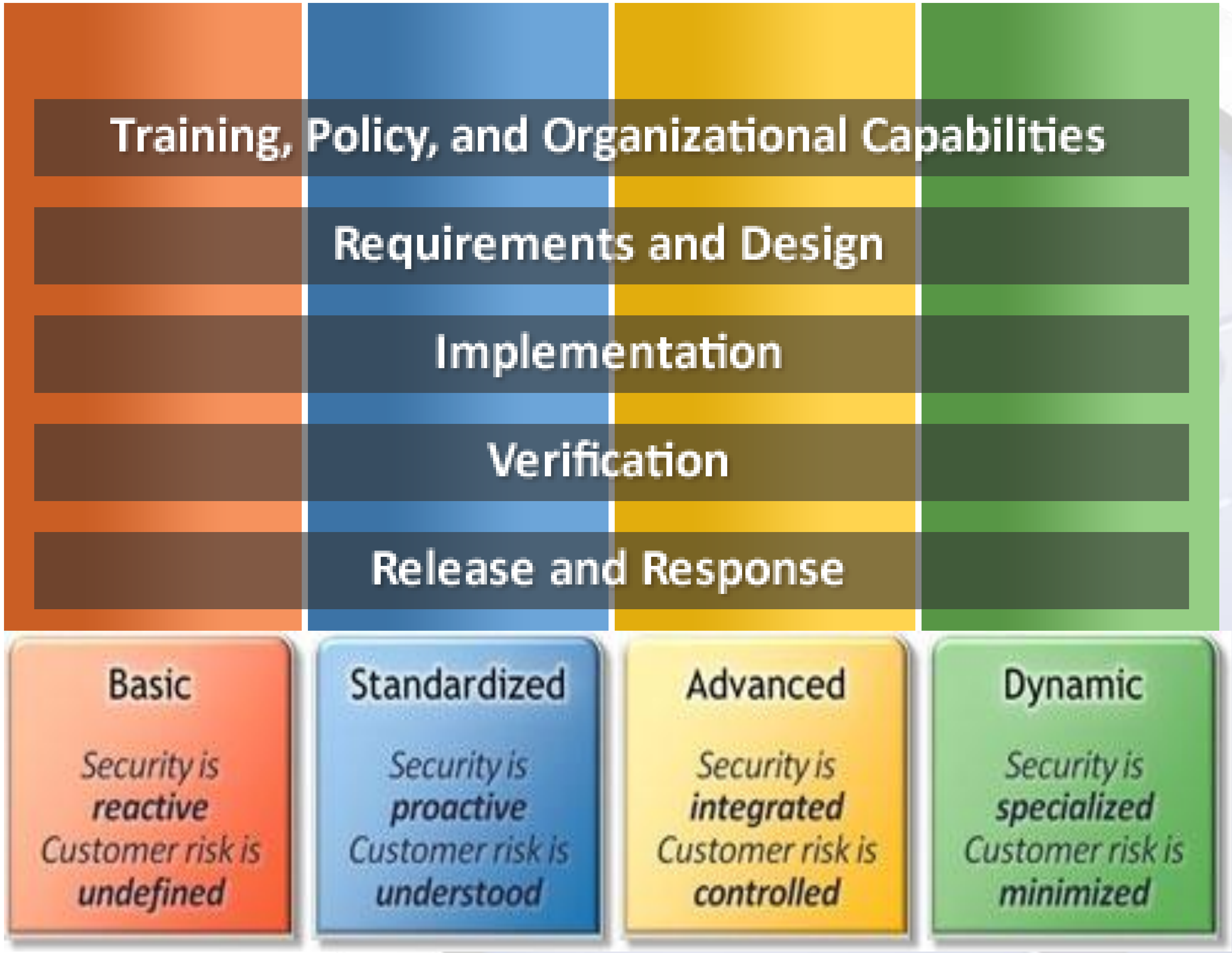
- Embedding security into software and culture
- Platform agnostic approach
  - Proven benefits
- Microsoft internal adoption
  - Extensive experience with security
  - Trustworthy computing



SDL optimization model



The SDL Optimization Model


Training, Policy, and Organizational Capabilities

Requirements and Design

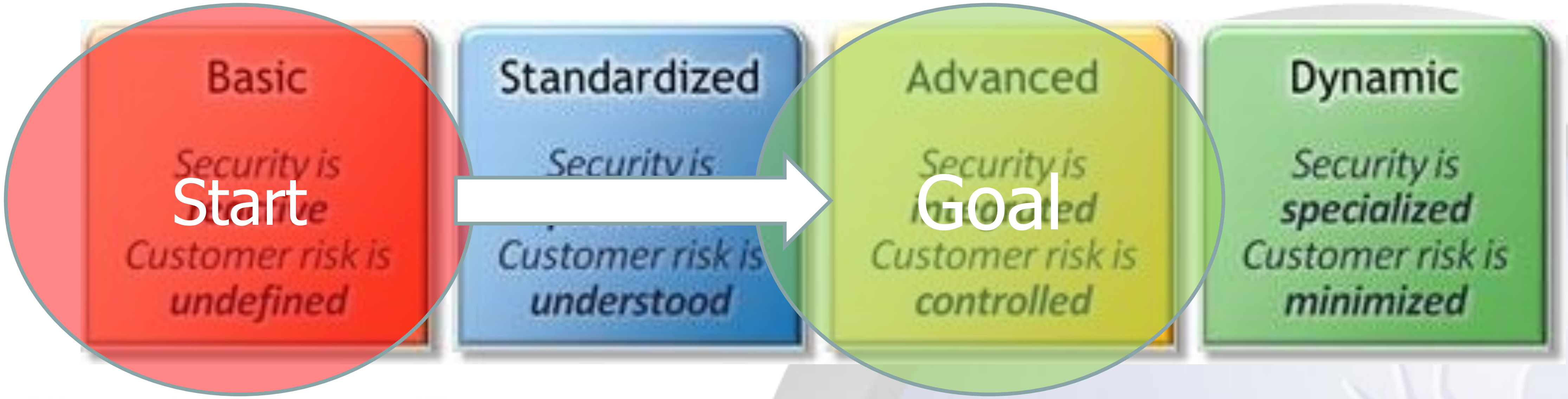
Implementation

Verification

Release and Response

Basic	Standardized	Advanced	Dynamic
Security is *reactive* Customer risk is *undefined*	Security is *proactive* Customer risk is *understood*	Security is *integrated* Customer risk is *controlled*	Security is *specialized* Customer risk is *minimized*

Achmea SDL optimization



Start

Goal

Basic

*Security is **reactive***
*Customer risk is **undefined***

Standardized

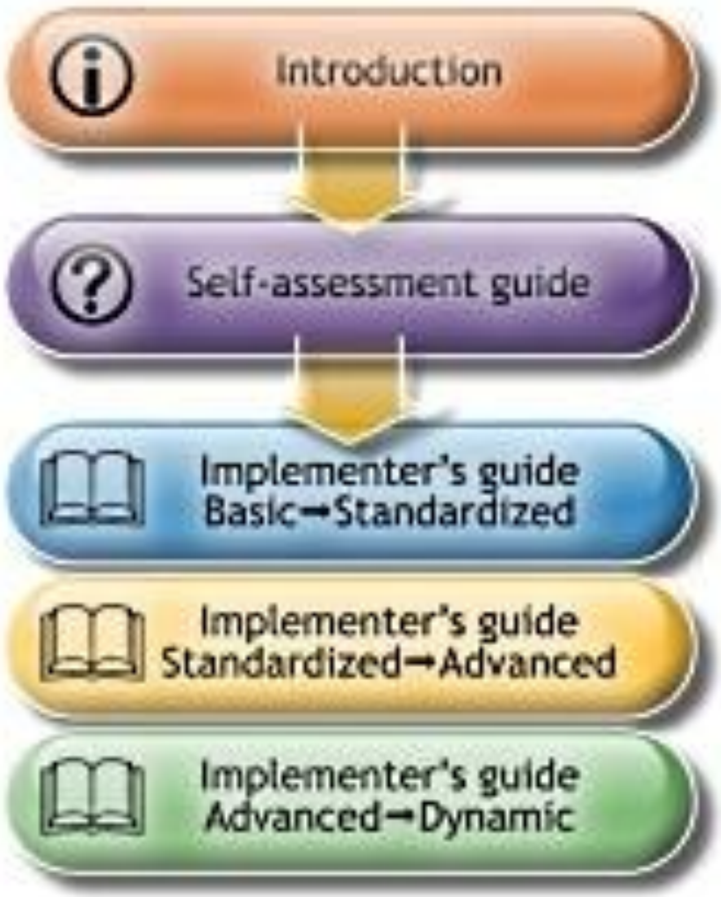
Security is [illegible]
*Customer risk is **understood***

Advanced

*Security is **integrated***
*Customer risk is **controlled***

Dynamic

*Security is **specialized***
*Customer risk is **minimized***



Introduction

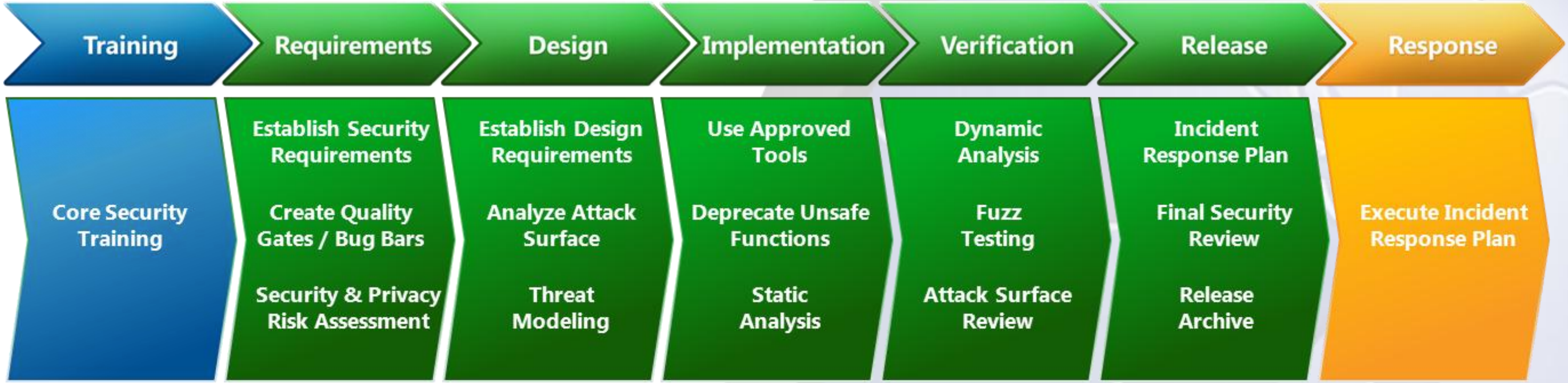
Self-assessment guide

Implementer's guide
Basic→Standardized

Implementer's guide
Standardized→Advanced

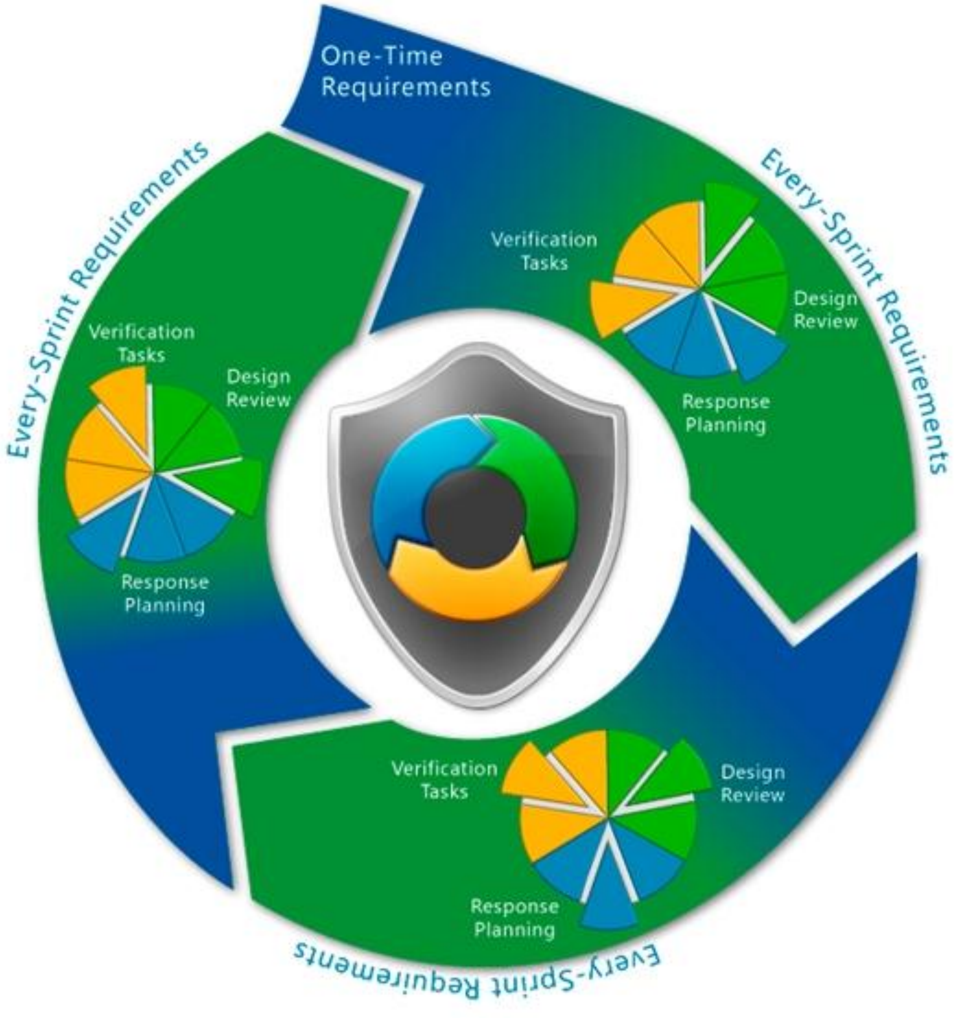
Implementer's guide
Advanced→Dynamic

Phases of Simplified SDL



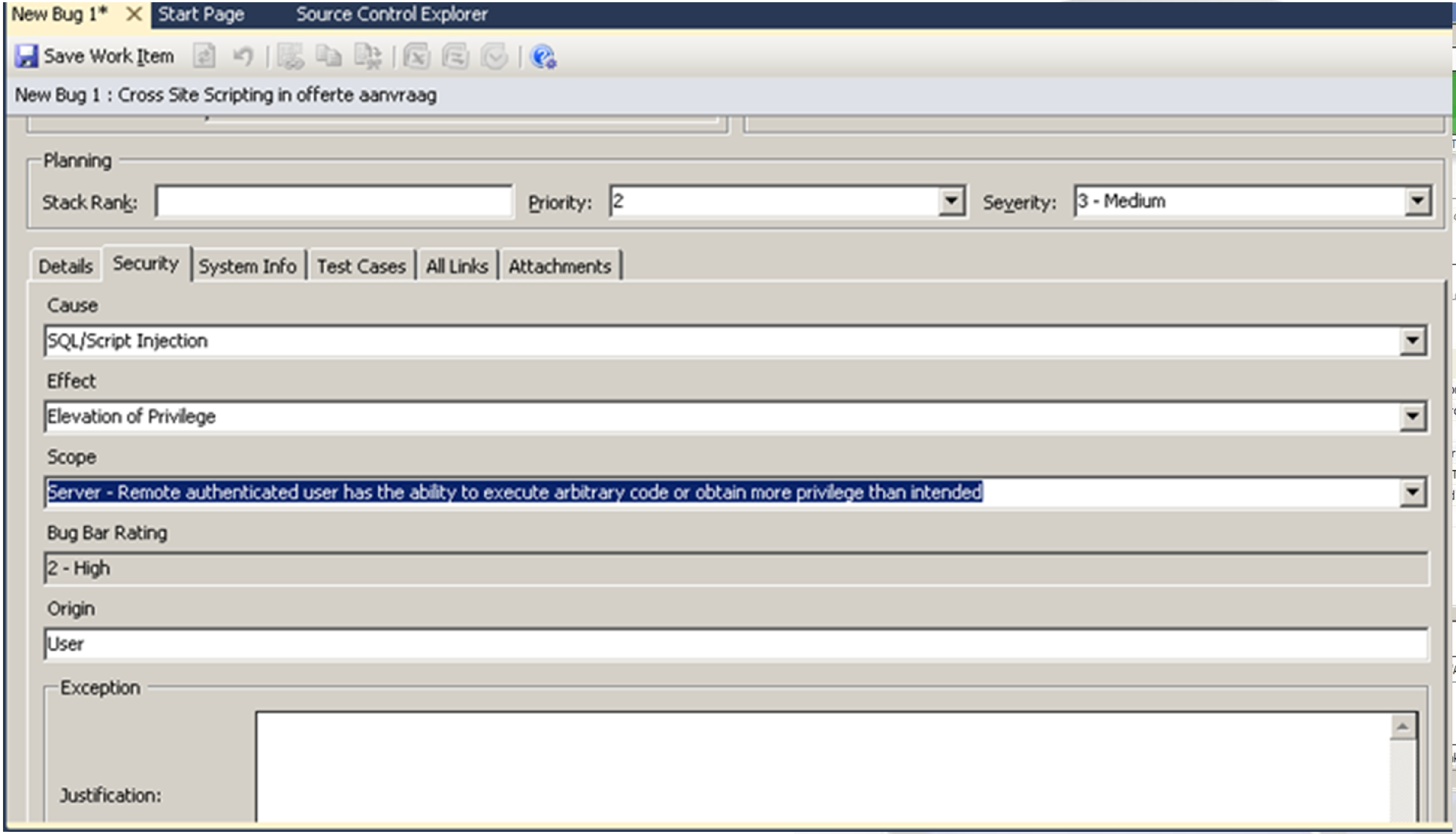
Training	Requirements	Design	Implementation	Verification	Release	Response
Core Security Training	Establish Security Requirements Create Quality Gates / Bug Bars Security & Privacy Risk Assessment	Establish Design Requirements Analyze Attack Surface Threat Modeling	Use Approved Tools Deprecate Unsafe Functions Static Analysis	Dynamic Analysis Fuzz Testing Attack Surface Review	Incident Response Plan Final Security Review Release Archive	Execute Incident Response Plan

Combining SDL and agile



- Requirements defined by frequency, not phase
  - Every-Sprint (most critical)
  - One-Time (non-repeating)
  - Bucket (all others)

Embedding SDL in process



New Bug 1 : Cross Site Scripting in offerte aanvraag

Planning

Stack Rank:

Priority: 2

Severity: 3 - Medium

Cause: SQL/Script Injection

Effect: Elevation of Privilege

Scope: Server - Remote authenticated user has the ability to execute arbitrary code or obtain more privilege than intended

Bug Bar Rating: 2 - High

Origin: User

Exception

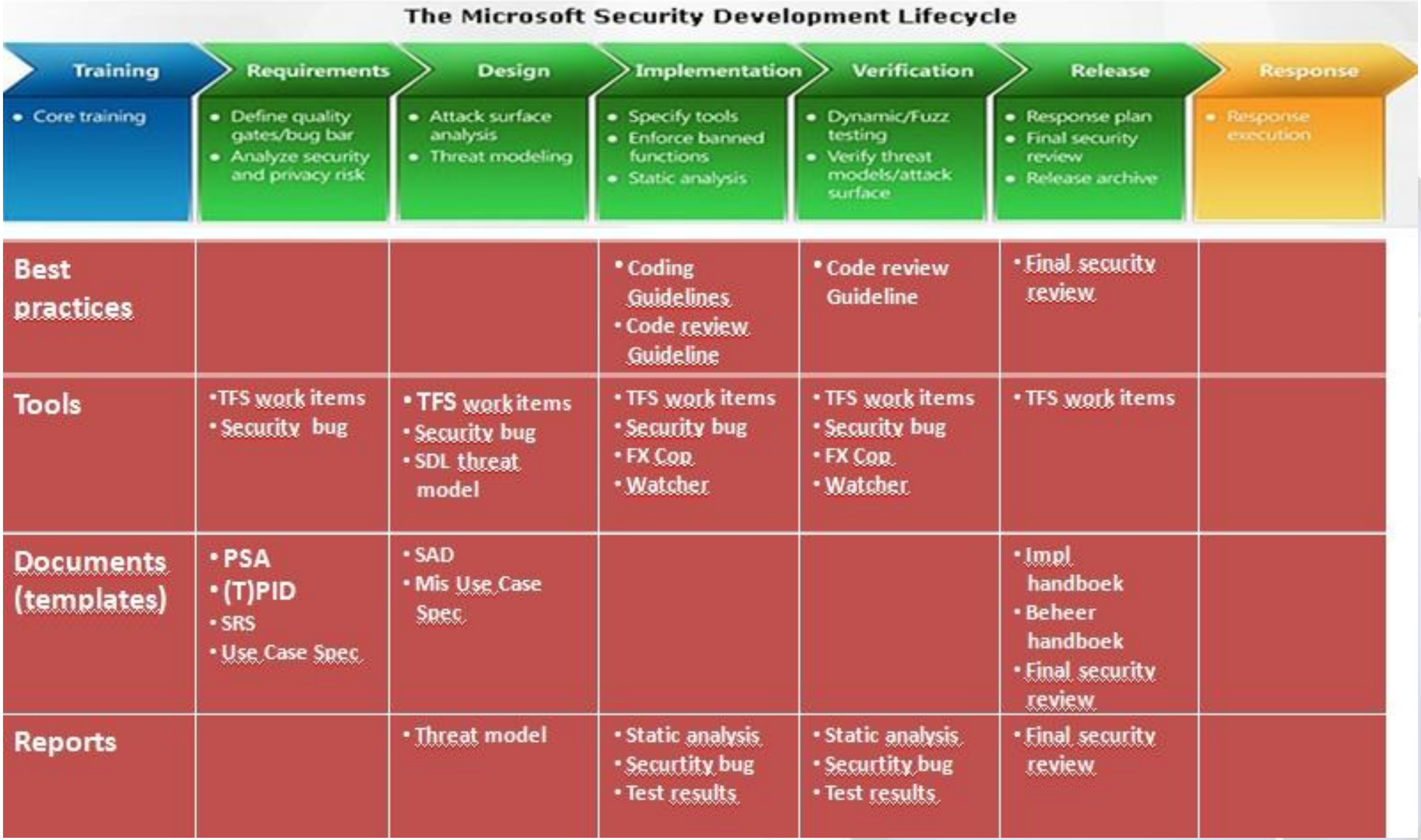
Justification:

IMPLEMENTING SDL AT ACHMEA

Focus at Achmea

- Emphasis on implementation at MScc
  - Line-of-business apps
  - Web portals
- Part of chain: bigger scope
- Embed SDL into “existing” development process
  - Sync with quality gates

Deliverables SDL for Achmea



The Microsoft Security Development Lifecycle

Training
- Core training
Requirements
- Define quality gates/bug bar
- Analyze security and privacy risk
Design
- Attack surface analysis
- Threat modeling
Implementation
- Specify tools
- Enforce banned functions
- Static analysis
Verification
- Dynamic/Fuzz testing
- Verify threat models/attack surface
Release
- Response plan
- Final security review
- Release archive
Response
- Response execution

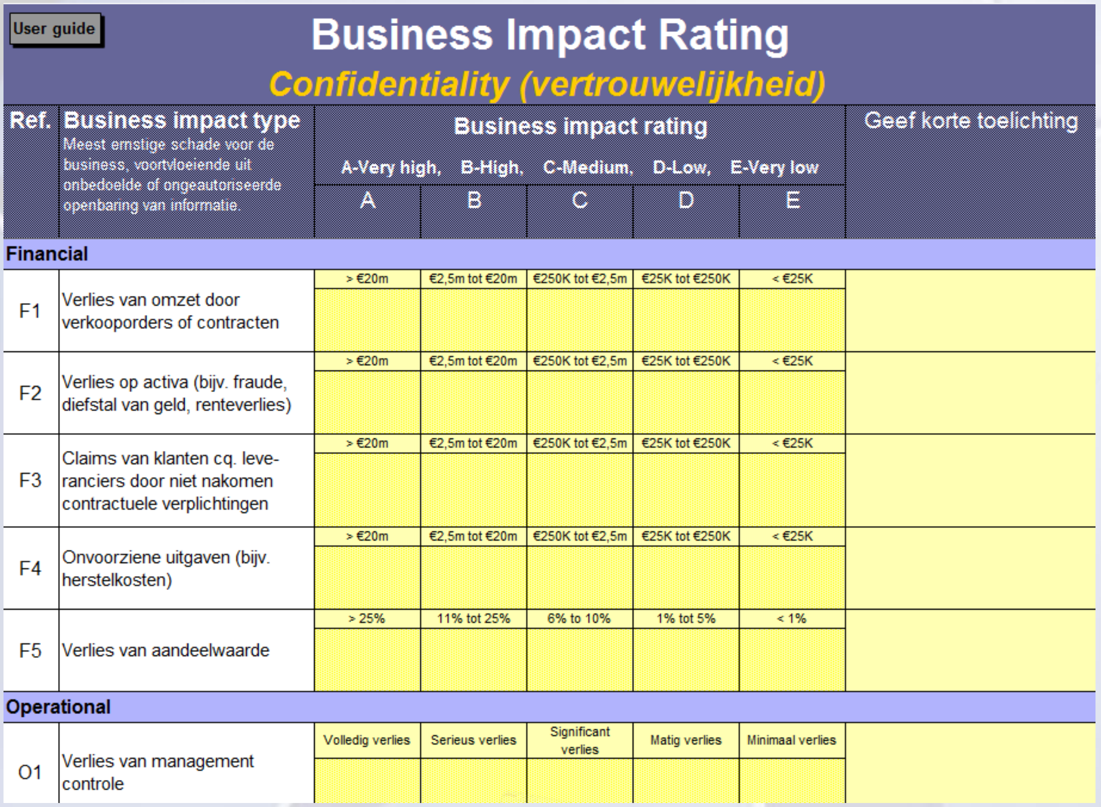
Best practices			• Coding Guidelines • Code review Guideline	• Code review Guideline	• Final security review	
Tools	• TFS work items • Security bug	• TFS work items • Security bug • SDL threat model	• TFS work items • Security bug • FX Cop • Watcher	• TFS work items • Security bug • FX Cop • Watcher	• TFS work items	
Documents (templates)	• PSA • (T)PID • SRS • Use Case Spec	• SAD • Mis Use Case Spec			• Impl handboek • Beheer handboek • Final security review	
Reports		• Threat model	• Static analysis • Securtity bug • Test results	• Static analysis • Securtity bug • Test results	• Final security review	

Training

- Online assessment and awareness course
- Security expert training
- Roadshow for all MScc employees
  - Focus on different phases in SDL for different roles

Requirements

- Business Impact Analysis (BIA)
  - Determines CIA rating
  - Weighs in on initial Architecture design and documentation



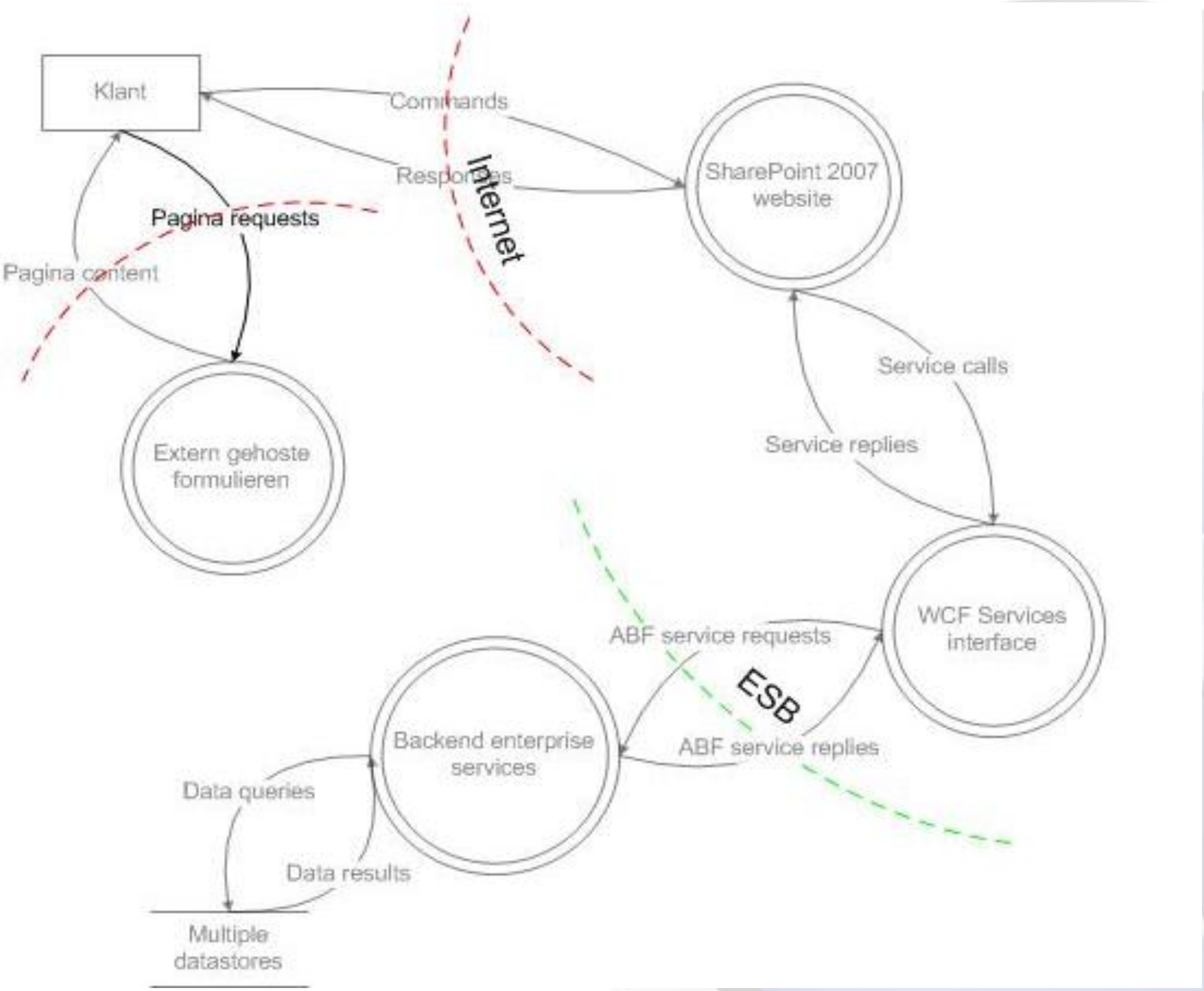
Business Impact Rating

Confidentiality (vertrouwelijkheid)

Ref.	Business impact type Meest ernstige schade voor de business, voortvloeiende uit onbedoelde of ongeautoriseerde openbaring van informatie.	Business impact rating					Geef korte toelichting
		A-Very high,	B-High,	C-Medium,	D-Low,	E-Very low	
		A	B	C	D	E	
Financial							
F1	Verlies van omzet door verkooporders of contracten	> €20m	€2,5m tot €20m	€250K tot €2,5m	€25K tot €250K	< €25K	
F2	Verlies op activa (bijv. fraude, diefstal van geld, renteverlies)	> €20m	€2,5m tot €20m	€250K tot €2,5m	€25K tot €250K	< €25K	
F3	Claims van klanten cq. leveranciers door niet nakomen contractuele verplichtingen	> €20m	€2,5m tot €20m	€250K tot €2,5m	€25K tot €250K	< €25K	
F4	Onvoorziene uitgaven (bijv. herstelkosten)	> €20m	€2,5m tot €20m	€250K tot €2,5m	€25K tot €250K	< €25K	
F5	Verlies van aandeelwaarde	> 25%	11% tot 25%	6% to 10%	1% tot 5%	< 1%	
Operational							
O1	Verlies van management controle	Volledig verlies	Serieus verlies	Significant verlies	Matig verlies	Minimaal verlies	

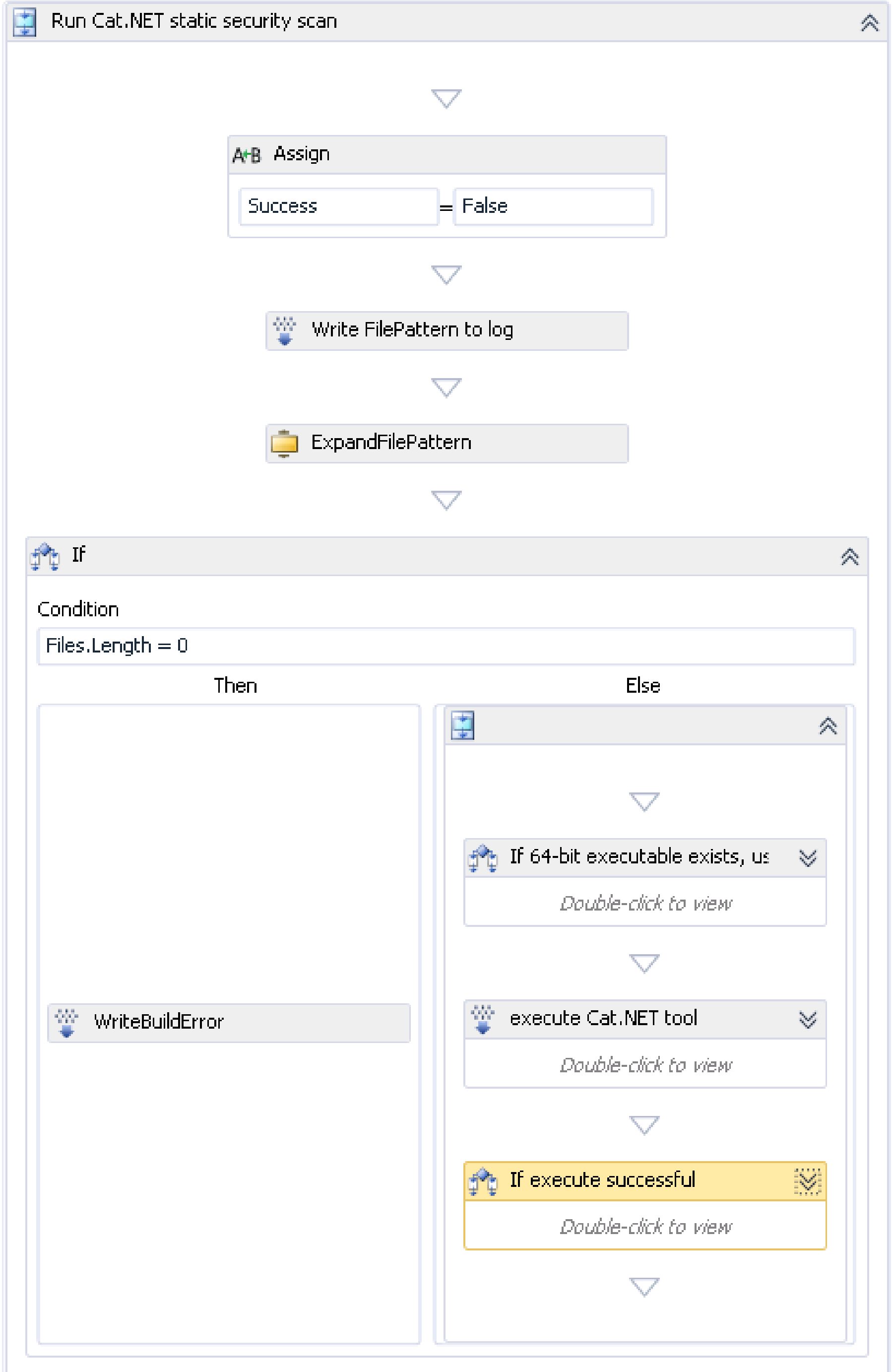
Design

- Con
  Thr
  - Ch
- Thr
  - Us
  - De
- Part



Implementation

- Adopted Patterns & Practices guidance
  - Best practices
  - Guidelines and checklists
  - Tooling
- Included CAT.NET in build
- Watcher



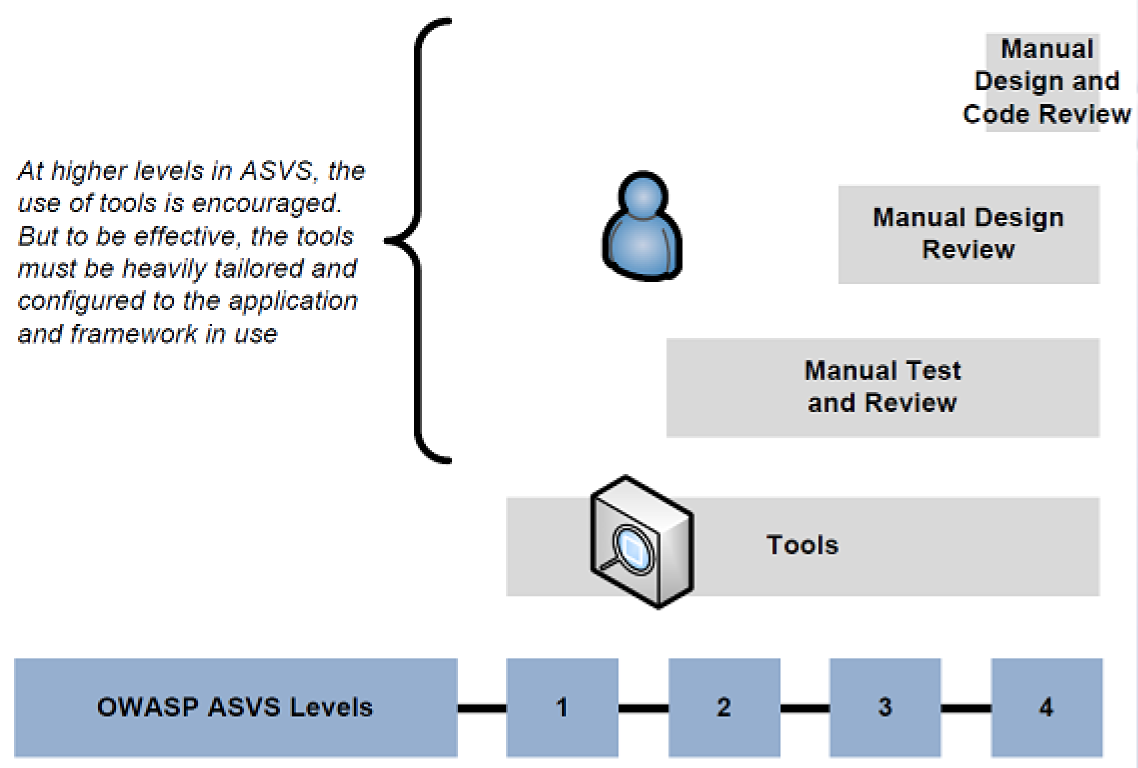
Success = False

Condition

Files.Length = 0

Verification

- BTOcc testplan adopted from OWASP
  - Testing for OWASP Top 10
  - ASVS testing
  - Dynamic, static and manual penetration testing
- Code reviews



At higher levels in ASVS, the use of tools is encouraged. But to be effective, the tools must be heavily tailored and configured to the application and framework in use

Manual Design and Code Review

Manual Design Review

Manual Test and Review

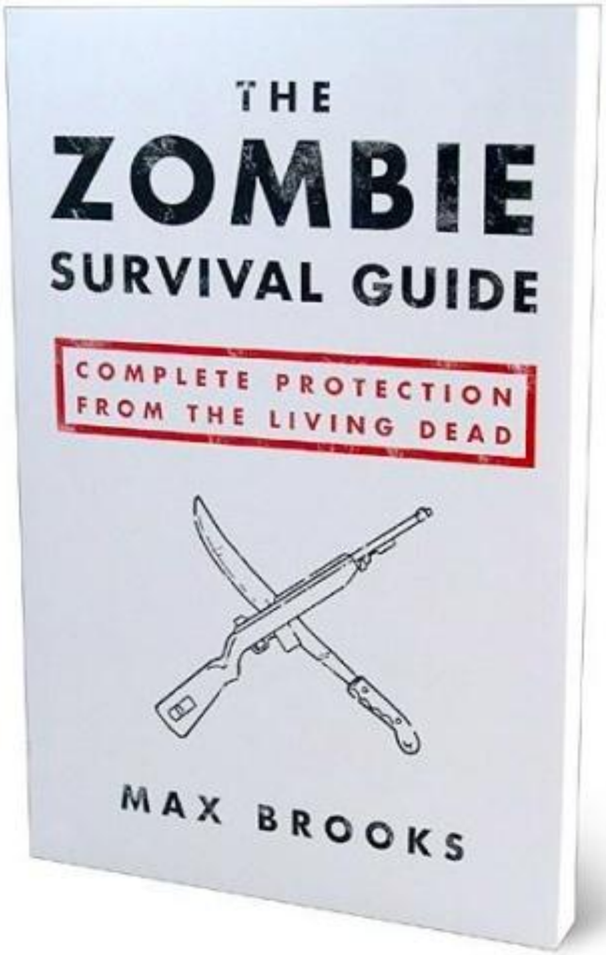
Tools

OWASP ASVS Levels — 1 — 2 — 3 — 4

Release

- Final Security Review (FSR)
  - Check on deliverables of previous phases
- Approval by Design Authority
- Ultimate quality gate

Response plan



- Incident response part of other departments
  - IT Operations (IDS, monitoring)
  - Security departments
- Close loop by applying lessons learned

LESSONS LEARNED

Taking hurdles

- Security as a hurdle
  - “False positives”
- Break perception
  - “Security takes time, budget and in not cool”
- Missing or sub-optimal tooling

Visibility

- Make sure you have security experts
  - Advocating security
  - People to ask questions
- Pick people that like it
- Find management that demands it

Achievable goals

- Small steps
- Not all at once
- Prioritize and pick from top 3

Continuous metrics

- Include security metrics in build
- Tooling is essential
- Testing only at end leads to disaster

Business and management

- Buy-in from management is essential
- Awareness at business is critical
- Don’t end in a showdown with business

Ongoing training

- Training alone is not enough
  - Offer help on-the-job
  - Not just before but during project as well
- Fast-moving field of security, attacks, vulnerabilities

Responsibility

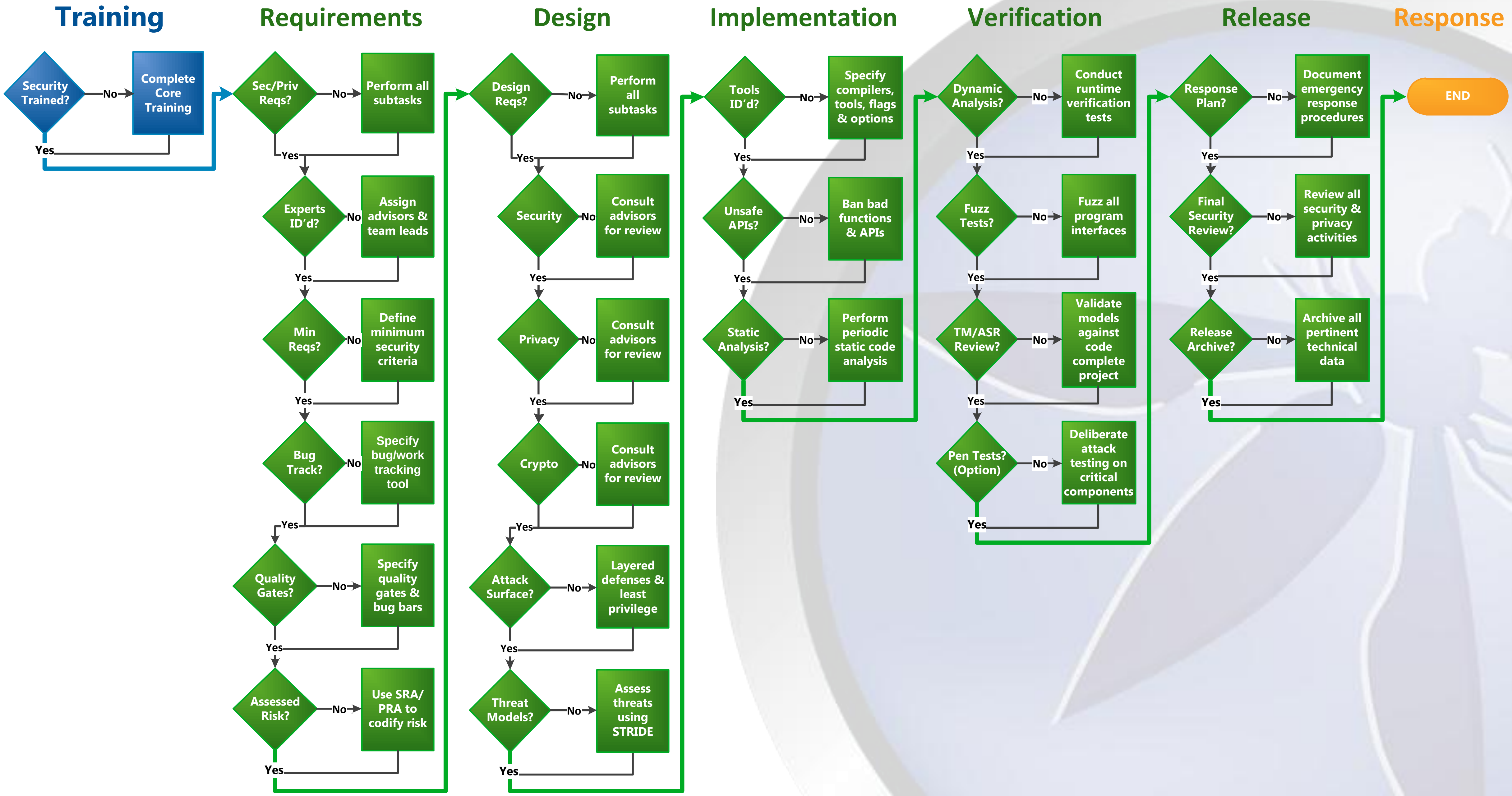
- Define clear roles
  - Who does what?
- Sharing responsibility

WRAPPING UP

Summary

- Embed security in your process
- It’s not easy
- Microsoft SDL turned out to be a good choice
- OWASP initiatives helped a lot
- You’re never done

Questions and Answers



Training

Security Trained? —No→ Complete Core Training
Yes

Requirements

Sec/Priv Reqs? —No→ Perform all subtasks
Yes
Experts ID'd? —No→ Assign advisors & team leads
Yes
Min Reqs? —No→ Define minimum security criteria
Yes
Bug Track? —No→ Specify bug/work tracking tool
Yes
Quality Gates? —No→ Specify quality gates & bug bars
Yes
Assessed Risk? —No→ Use SRA/ PRA to codify risk
Yes

Design

Design Reqs? —No→ Perform all subtasks
Yes
Security —No→ Consult advisors for review
Yes
Privacy —No→ Consult advisors for review
Yes
Crypto —No→ Consult advisors for review
Yes
Attack Surface? —No→ Layered defenses & least privilege
Yes
Threat Models? —No→ Assess threats using STRIDE
Yes

Implementation

Tools ID'd? —No→ Specify compilers, tools, flags & options
Yes
Unsafe APIs? —No→ Ban bad functions & APIs
Yes
Static Analysis? —No→ Perform periodic static code analysis
Yes

Verification

Dynamic Analysis? —No→ Conduct runtime verification tests
Yes
Fuzz Tests? —No→ Fuzz all program interfaces
Yes
TM/ASR Review? —No→ Validate models against code complete project
Yes
Pen Tests? (Option) —No→ Deliberate attack testing on critical components
Yes

Release

Response Plan? —No→ Document emergency response procedures
Yes
Final Security Review? —No→ Review all security & privacy activities
Yes
Release Archive? —No→ Archive all pertinent technical data
Yes

Response

END