Karlsruher Institut für Technologie

safeware engineering
safe and secure software

KASTEL

Security by Design und agiles Vorgehen – Integration oder Widerspruch?

Hubert B. Keller
Leiter Fachgebiet Advanced Automation Technologies – A2T + Reliable, safe & secure Software and Systems – RS4

Karlsruher Entwicklertag 2021 - Die Konferenz für Software Engineering - Agilität, Qualität und Innovation! Karlsruhe, Deutschland || 9. bis 10. Juni 2021

Institut für Automation und angewandte Informatik (IAI)

Agenda

Software Engineering | Status

Software Security | Status

Security by Design

Agile Prozesse

Integration oder Widerspruch?

Resümee

Software Engineering

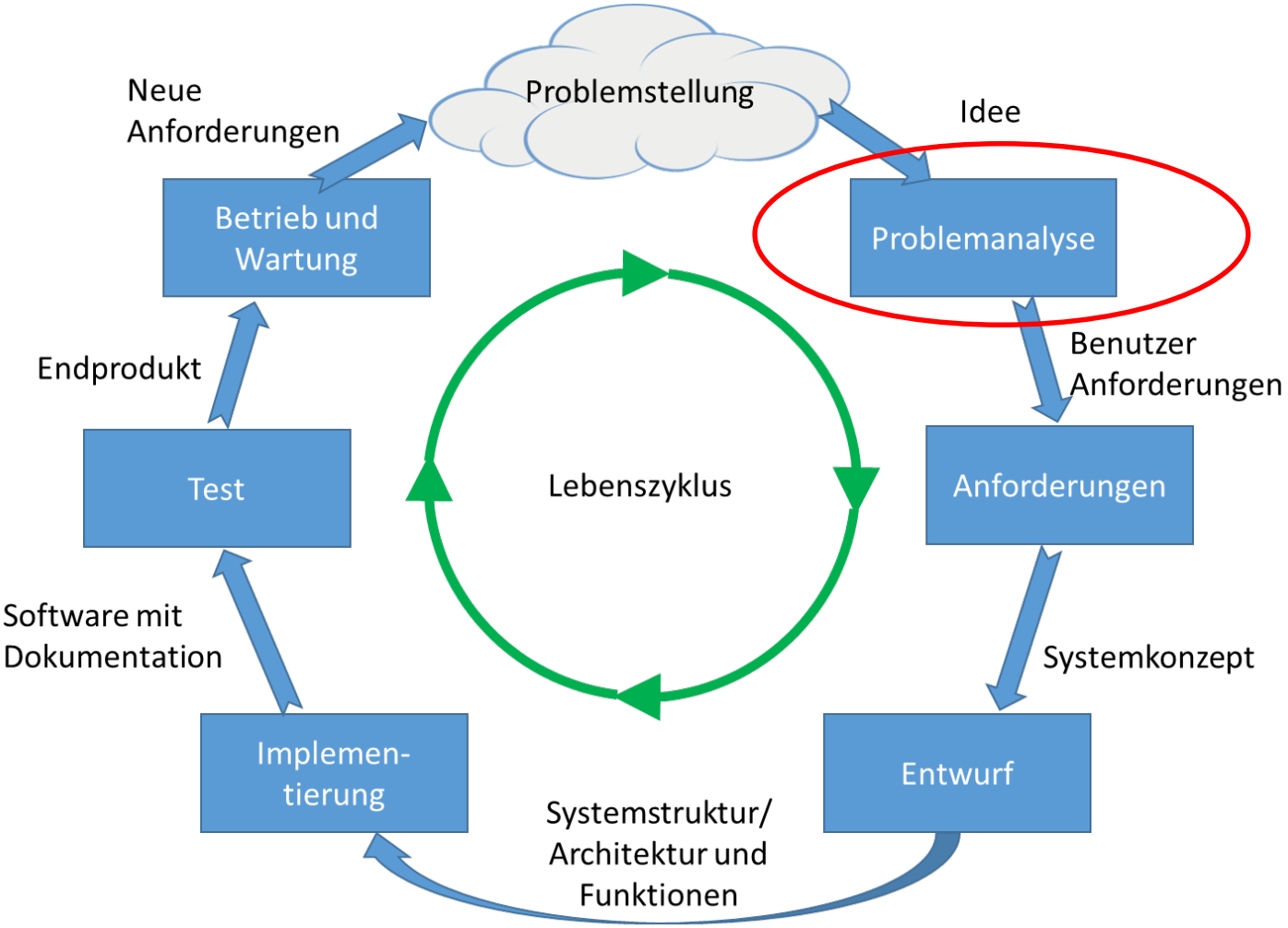
Status

Software | Umfänge und Fehlergehalte

Software-Anwendung	Codeumfang	Fehler Standard 0,5% (V-Modell)	Fehler Höchstqualität: 0,01% (Militär, Aerospace)
Durchschnittliche iPhone~App	40.000	200	4
Herzschrittmacher	80.000	400	8
Photoshop 1.0	128.000	640	13
Space Shuttle Flugsoftware	400.000	2.000	40
Grafikschnittstelle CryEngine 2	1.000.000	5.000	100
Hubble Weltraumteleskop	2.000.000	10.000	200
Windows 3.1	2.500.000	12.500	250
Kontrollsoftware einer US-Militärdrohne	3.500.000	17.500	350
Mars Curiosity Rover	5.000.000	25.000	500
Google Chrome	7.000.000	35.000	700
Photoshop CS 6	10.000.000	50.000	1.000
OpelAmpera	10.000.000	50.000	1.000
Android OS	12.000.000	60.000	1.200
Boeing 787 Dreamliner	14.000.000	70.000	1.400
Linux3.10	16.000.000	80.000	1.600
Firefox Browser	18.000.000	90.000	1.800
F-35 Kampfflugzeug	24.000.000	120.000	2.400
Windows7	40.000.000	200.000	4.000
Microsoft Office 2013	45.000.000	225.000	4.500
Teilchenbeschleuniger Large Hadron Collider	50.000.000	250.000	5.000
Windows Vista	50.000.000	250.000	5.000
Facebook	62.000.000	310.000	6.200
MacOSXI0.4	86.000.000	430.000	8.600
Steuersoftware für moderne Autos	100.000.000	500.000	10.000
Webseite healthcare.gov	500.000.000	2.500.000	50.000
Menschliches Genom	3.300.000.000	16.500.000	330.000

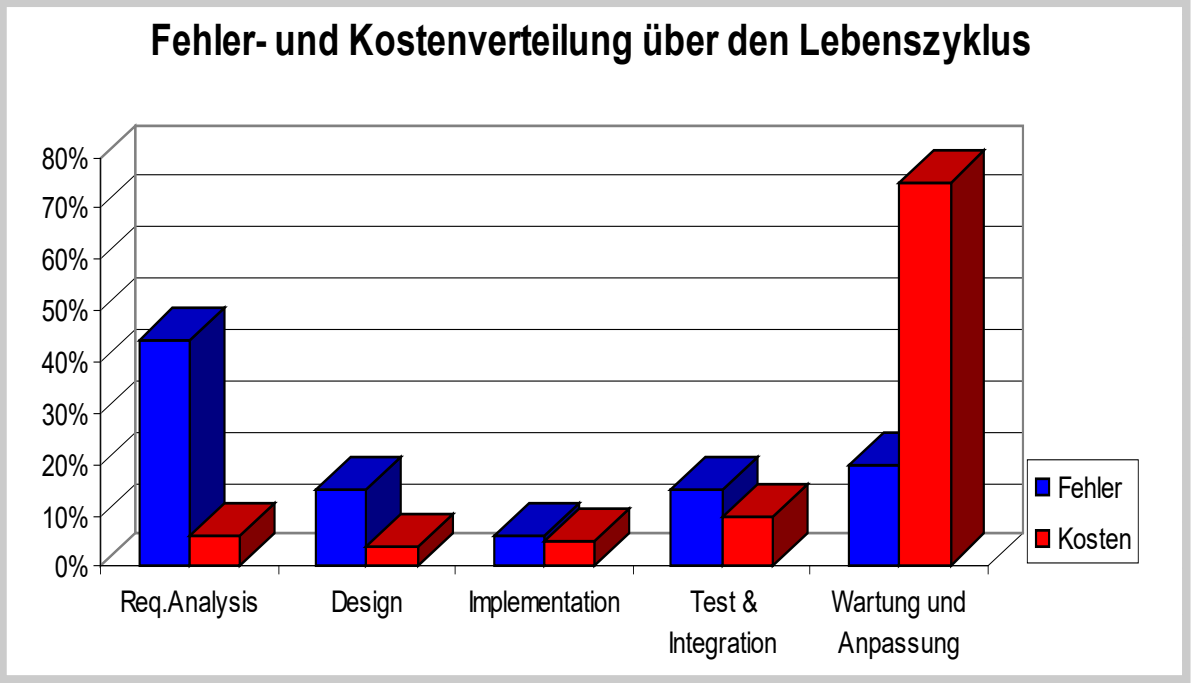
Anmerkung: Komplexität der Software-Strukturen ist nicht berücksichtigt.
Hohe cC verstärkt Fehlerquote massiv!

Life cycle und Phasen im SW Engineering



Fehler und Phasenzuordnung

- HSE Out of Control – Analyse (2003)
  44% aller Fehler → inkorrekte Spezifikation



Analysephase wird deutlich unterschätzt
→ Mehr Fokus auf Analyse statt Codierung

Out of control - Why control systems go wrong and how to prevent failure. Health and Safety Executive (2003), No 238)

Erfolg in der Software Entwicklung

MODERN RESOLUTION FOR ALL PROJECTS

	2011	2012	2013	2014	2015
SUCCESSFUL	29%	27%	31%	28%	29%
CHALLENGED	49%	56%	50%	55%	52%
FAILED	22%	17%	19%	17%	19%

The Modern Resolution (OnTime, OnBudget, with a satisfactory result) of all software projects from FY2011–2015 within the new CHAOS database. Please note that for the rest of this report CHAOS Resolution will refer to the Modern Resolution definition not the Traditional Resolution definition.

CHAOS RESOLUTION BY PROJECT SIZE

	SUCCESSFUL	CHALLENGED	FAILED
Grand	2%	7%	17%
Large	6%	17%	24%
Medium	9%	26%	31%
Moderate	21%	32%	17%
Small	62%	16%	11%
TOTAL	100%	100%	100%

The resolution of all software projects by size from FY2011–2015 within the new CHAOS database.

CHAOS Report of the Standish Group

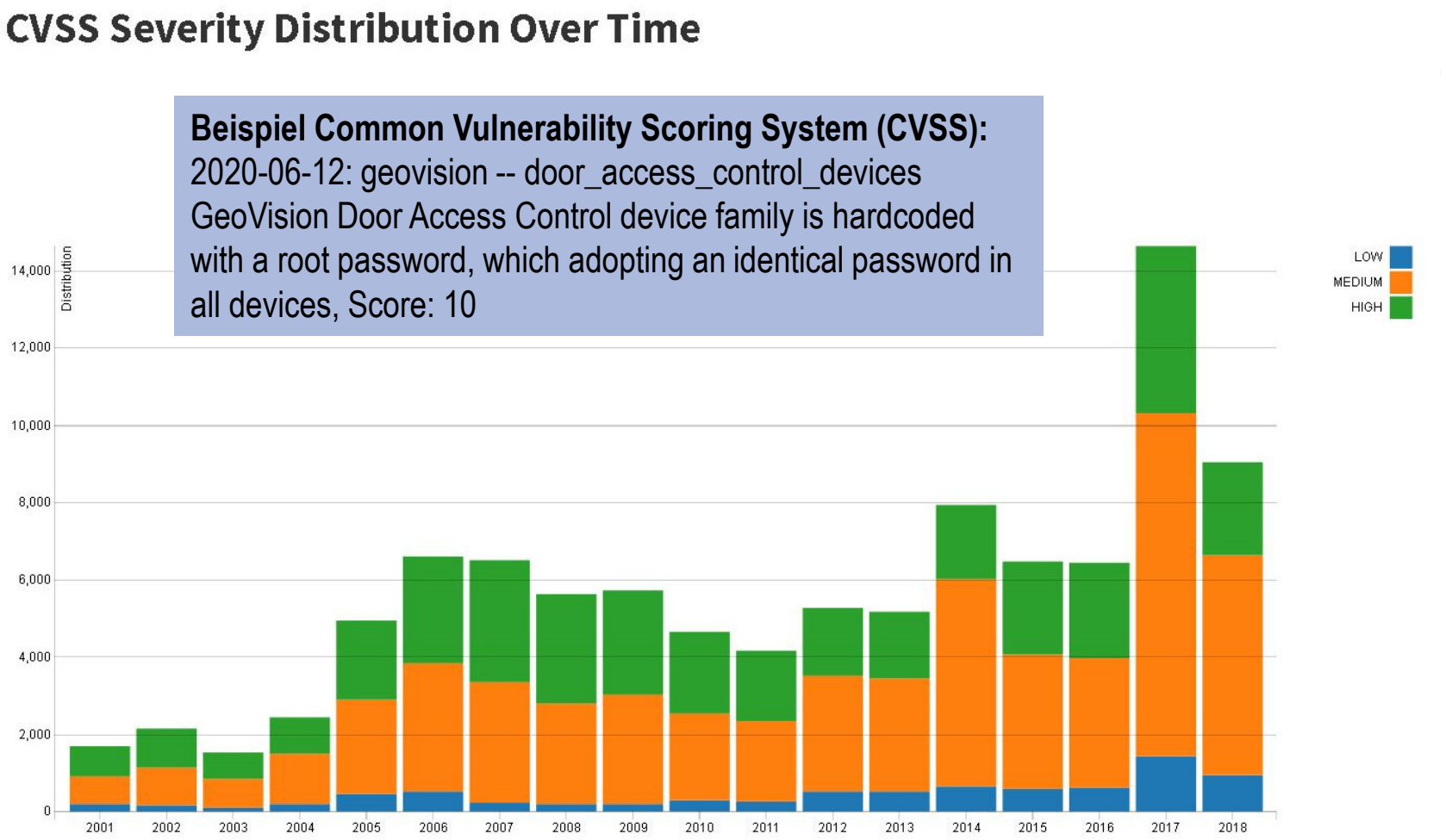
- The project is completed on time and on budget, with all features and functions as initially specifed.
- The project is completed and operational but over-budget, over the time estimate, and/or offers fewer features and functions than specifed.
- The project is canceled at some point during the development cycle or not used after implemented.

Erfolg sieht anders aus!

Software Security

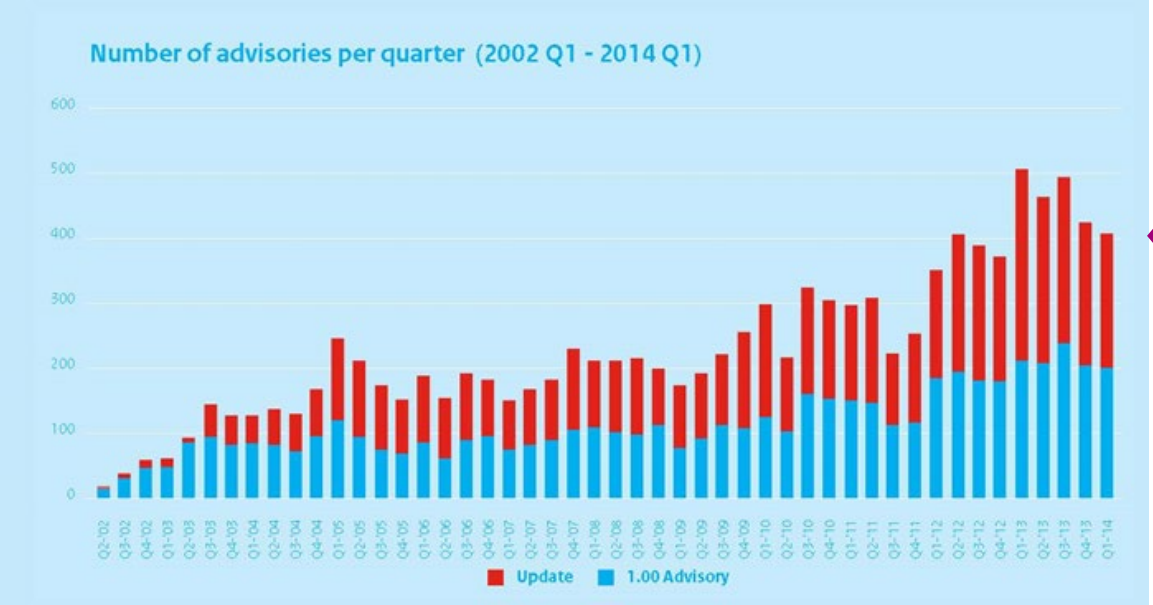
Status

Entwicklung der Kritikalität von SW Schwachstellen



Entwicklung Security Schwachstellen in Software

Update Schwachstelle
IBM - spectrum_protect_plus. 2020-06-15
Score: 10 (highest)
IBM Spectrum Protect Plus 10.1.0 through 10.1.5 could allow a remote attacker to execute arbitrary code on the system.
By using a specially crafted HTTP command, an attacker could exploit this vulnerability to execute arbitrary command on the system. This vulnerability is due to an incomplete fix for CVE-2020-4211.



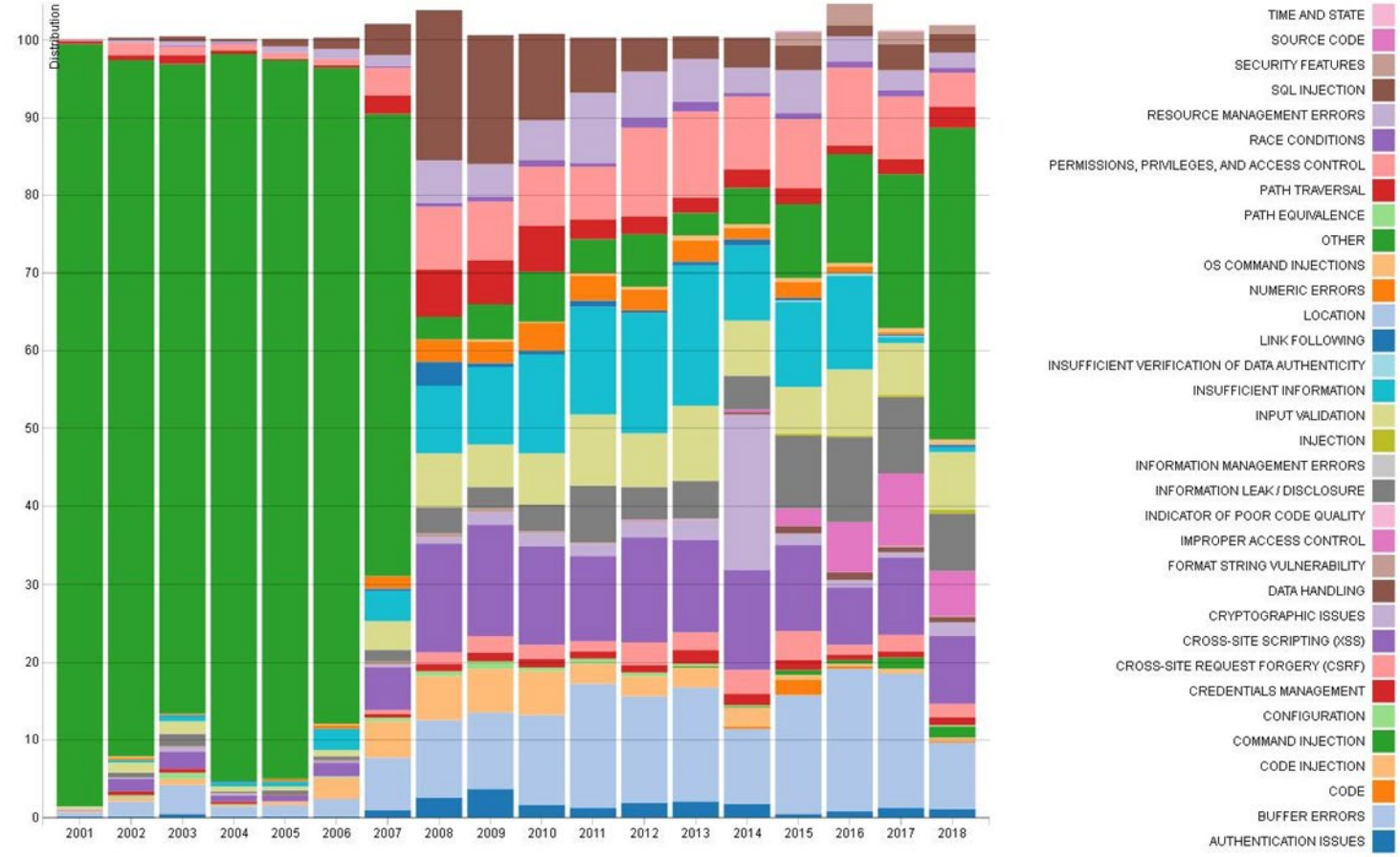
Update = wiederholte gleiche Schwachstelle

Aus: Cyber Security Assessment Netherlands - CSAN-4

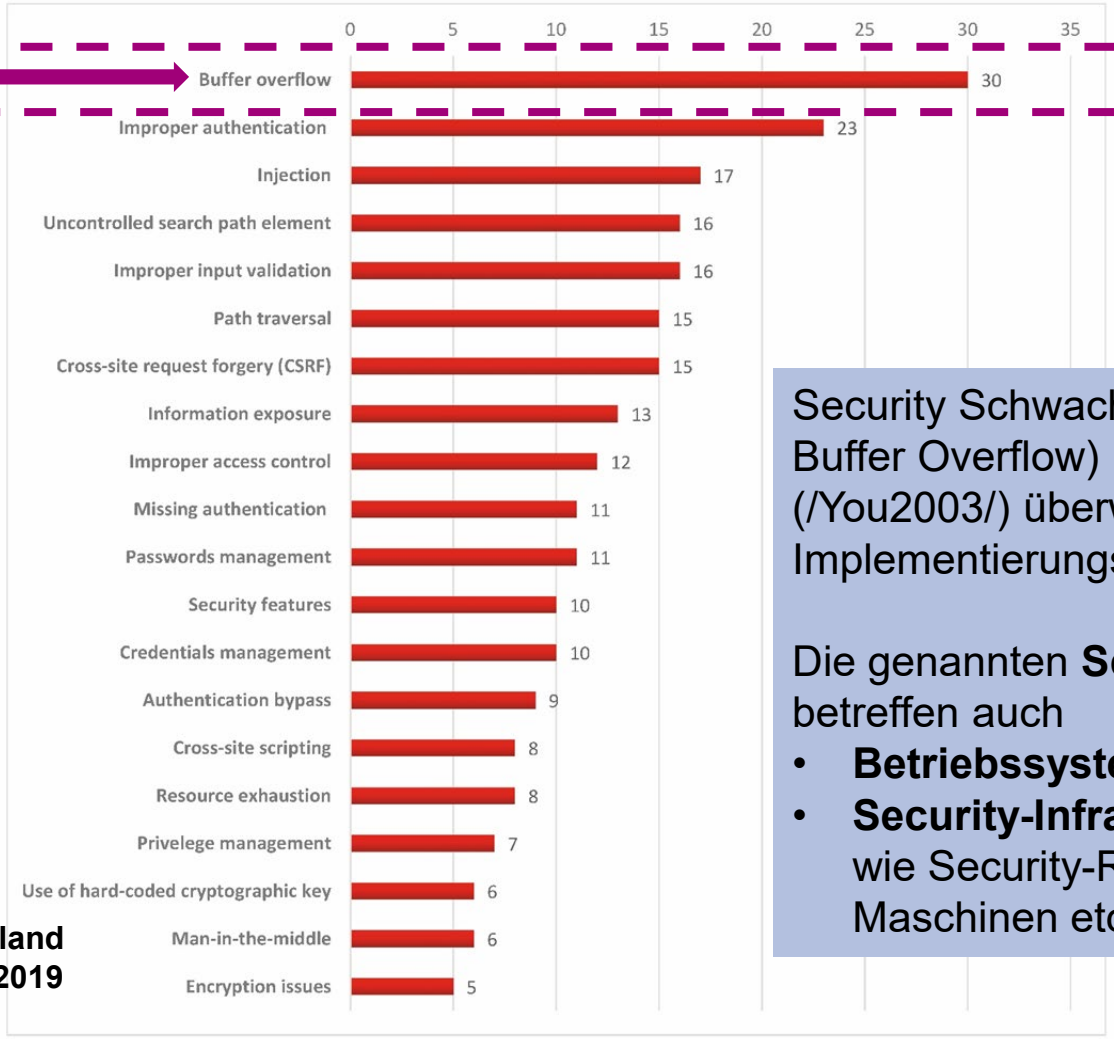
Arten von SW Schwachstellen

Relative Vulnerability Type Totals By Year

The vulnerabilties in the NVD are assigned a CWE based on a slice of the total CWE Dictionary. The visualization below shows a stacked bar graph of the total number of vulnerabilities assigned a CWE for each year. It is possible (although not common) that a vulnerability has multiple CWEs assigned.



Anteile Arten Schwachstellen



US CERT and Homeland Security Data Base 2019

Security Schwachstellen (z.B. Buffer Overflow) liegen laut (/You2003/) überwiegend in Implementierungsschwachstellen.

Die genannten **Schwachstellen** betreffen auch

- **Betriebssysteme**
- **Security-Infrastruktursysteme** wie Security-Router, virtuelle Maschinen etc.

Konsequenz I

- **Software**
  - bietet hohes Innovationspotential (Wertschöpfung)
  - beinhaltet hohe Anzahl an Fehler und Security Schwachstellen

- Beherrschung von **Komplexität** ist ein kritischer Faktor
- Software Engineering ist (noch) keine Standard-Ingenieurs-Disziplin
- Fehler werden in der **Analyse-/Design-Phase induziert**
- Fokus auf Codierung nicht zielführend (agiles Vorgehen?)

- Security Schwachstellen ermöglichen **Angriffe**
  (Authentifizierung - Wer / Autorisierung – Rechte / Implementierung – Schwachstellen Sprache)

→ **Fokus auf Security Eigenschaften von Architektur und Komponenten**

Security by Design

Software Security in Bezug auf Design

Im Kern lässt sich diese **Anforderung** konstruktiv abbilden auf:

- **Security** im **Requirements Engineering** vorzusehen,
- die **Software Architektur** und die Funktionen daran auszurichten,
- sichere **Programmiersprachen** und **Betriebssysteme** einzusetzen und
- alle **Vorkommnisse** zu erfassen, protokollieren und in
- entsprechende **systematische Verbesserungen** umzusetzen
  → prinzipiell vermeiden, nicht reparieren
  → wenn Schwachstelle erkannt, systematisch beheben, nicht reparieren

Secure by Default

- Funktionen und Vorgehensweisen auf dem Stand der Technik der IT Security sind **standardmäßig** in den Komponenten und Systemen **verfügbar**
- **Eindeutige Spezifikation** und Dokumentation der Funktionen von Geräten, Systemen oder Lösungen
- Einhaltung von **Zuverlässigkeitsanforderungen** an die Nutzfunktionen bei der Vernetzung von Komponenten
- Definition der Nutzfunktionen für den bestimmungsgemäßen Gebrauch, **Rückwirkungsfreiheit** von Zusatzfunktion
- **…**

Secure by Design

- IT-Security Konzepte und Funktionen sind **integraler** Bestandteil der Komponenten und Lösungen
- **Reduzierung** der Systemkomplexität um Fehler zu vermeiden

- Konzept und Entwurf entsprechend Security Vorgaben,
  Architektur muss Security Regeln entsprechen

- Kommunikationsschnittstellen und Wege eindeutig spezifizieren mit Angabe der Security Anforderungen

- By default kein Zugang, Zugang explizit vergeben
- Minimale Zugriffsrechte, nur die absolut notwendigen Rechte vergeben

- Datenkonzepte und Funktionen mathematisch präzise umsetzen,
  Kriterien der Sprachwahl, keine zusätzlichen Freiheitsgrade

Secure Coding Practices

Robert Seacord, https://www.securecoding.cert.org/confluence/display/seccode/Top+10+Secure+Coding+Practices

- **Architect and design for security policies. Create a software architecture and design your software to implement and enforce security policies.**

- **Keep it simple. Keep the design as simple and small as possible.**
  Complex designs increase the likelihood that errors will be made.
- **Default deny**. Base access decisions on permission rather than exclusion. This means that, by default, access is denied.
- Adhere to the principle of **least privilege**. Every process should execute with the the least set of privileges necessary to complete the job.

Typstrenge und statisch analysierbare Sprachen

- Use static and dynamic **analysis tools** to detect and eliminate additional security flaws.
- **Validate input**. Validate input from all untrusted data sources. Proper input validation can eliminate the vast majority of software vulnerabilities.
- **Sanitize data** sent to other systems.

Konsequenz II

- Security by Design ist eine **Architektur zentrierte** Aufgabe
- **Securityeigenschaften** sind als Anforderungen zu definieren
- Architektureigenschaften müssen aus der **Anforderungsanalyse** abgeleitet werden (Gesamtsicht notwendig)

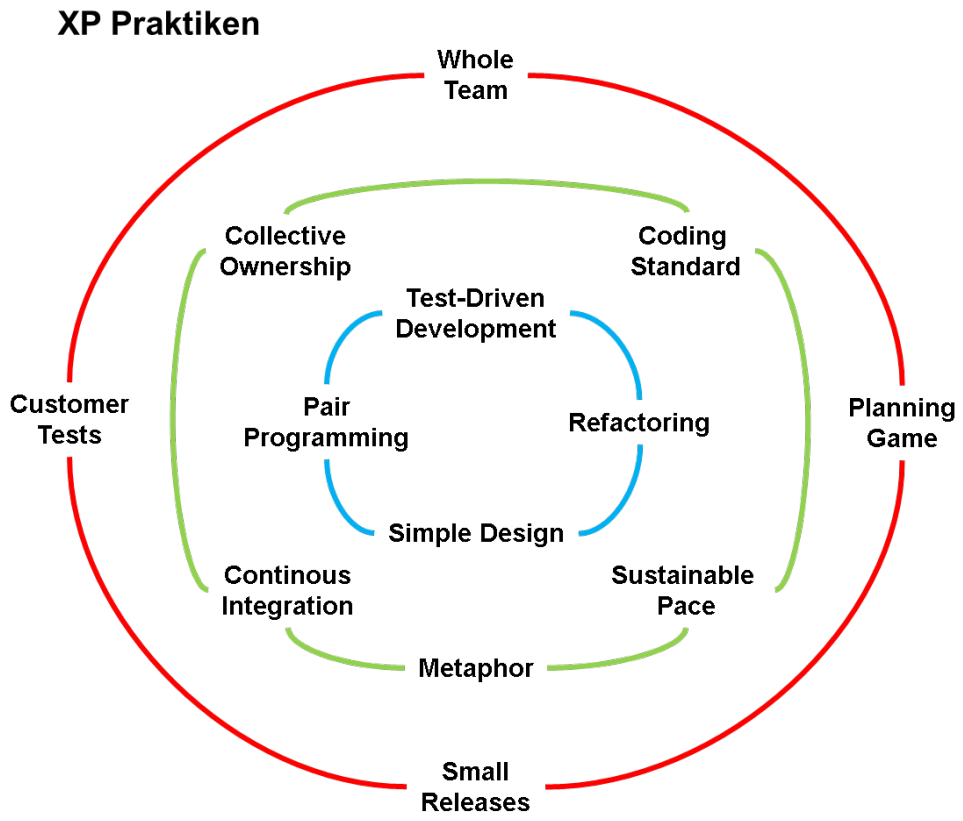
- Zugänge (Schnittstellen) sind hinsichtlich **zulässiger Benutzer** und **zulässiger Rechte** abzusichern
- **Kommunikation** ist abzusichern → **Architekturkomponenten nach außen** sind explizit zu definieren (Exposition / Angriffsrisiko)

- Daten (Kommandos, Parameter, etc.) sind **typstreng** zu implementieren und zur **Laufzeit** zu prüfen (Typ, Wert, Index)

→ **SW Vorgehensmodell** muss auf **Architektur**, **Komponenten** und deren **Security Anforderungen** fokussieren

Agile Prozesse

Charakter agiler Vorgehensmodelle (z. B. XP, Scrum)



- Hochgradig iterative Softwareentwicklungsschritte (Sprint)
- Planung mit **überschaubarer Story**
- **Rudimentäre Spezifikation** in Form von automatisierten Tests mit Testfällen
- **Systemmetapher** statt Architektur
- Einfacher Entwurf – KISS
- …

Security Aspekte agiler Modelle

- Fehlen einer expliziten Spezifikation und einer Entwurfsdokumentation führt nur **implizit zu einer Zielarchitektur mit Security**

- Grundlegende **Architekturentscheidungen** werden nicht getroffen, können sich aber auch nicht aus dem „agilen“ Vorgehen ergeben (falls Zuverlässigkeit von Beginn an vorhanden sein soll)

- Code-Zentrierung liefert nur **lokale Sicht**
  → Security ist eine **globale** Eigenschaft (Architektur+Funktion)

- **Entwurfsdokumentation** in Form der Testfälle und des Codes verfügbar

→ Security Assets sind **Pre-Requirements** (explizites Design)

Vorgehensmodelle, Effizienz und Zielgenauigkeit

Formale Prozesse
(V-Modell XT, Cleanroom, …)
- Phasen mit Gegenprüfung
- Detaillierte Spezifikation
- …
- CMMI Sicht

Agile Prozesse (SCRUM XP, …)
- Backlog
- Sprint
- Pair Programming
- Rollen
- Story Card
- …

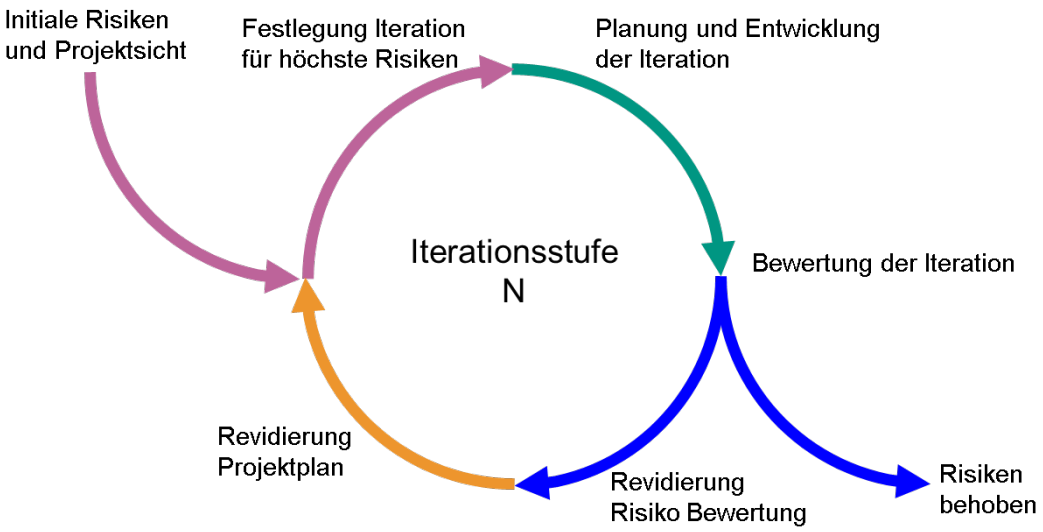
Vollständige Spezifikation
→ Utopie

?

Architektur automatisch aus Ablauf
→ Utopie

RUP – Rational Unified Process

- **Iterative Softwareentwicklung um Risiken vermeiden**
  - Frühe Umsetzung mit **Kernarchitektur für Kernrequirements** und iterative Erweiterung (Security Eigenschaften)
- **Komponentenorientierte Architekturen mit hoher Wiederverwendbarkeit**
  - Anpassungsfähig und flexibel
- Visuelle Modellierung mit UML
- …



Vorgehensmodell und Architektur-bezogene Security

V-Modell XT (, Cleanroom, …)
Vollständige Spezifikation
→ Utopie

Agile Prozesse SCRUM (XP, …)
Architektur automatisch
aus Ablauf
→ Utopie

Rational Unified Process
mit Best Practice Methoden

- Iterative Softwareentwicklung
- Verwaltung von Anforderungen
- **Komponentenbasierte Architekturen**
  → Security by Design
- Software visuell modellieren
- Qualität der Software verifizieren
- Änderungen der Software kontrollieren

Konsequenz III

- Agile Prozesse **analysieren nicht bis zu den Kern-Requirements**
- Sprint / Story ist Basis für eine nur **lokale Entscheidung**
  (Funktionssicht lokal)
- Architektur ergibt sich eher **implizit**
- Umsetzung **global** relevanter Security Anforderungen offen
- Security Anforderungen sind **Architektur-bezogen**
- Code-Zentrierung verliert abstrakt modellmäßige mathematische Präzision
  (Repräsentation für Compilierung und Laufzeit)

Integration oder Widerspruch?

Security by Design und agile Entwicklung

Zitat zu Security im agilen Prozess (agiler Chefentwickler/Consultant)

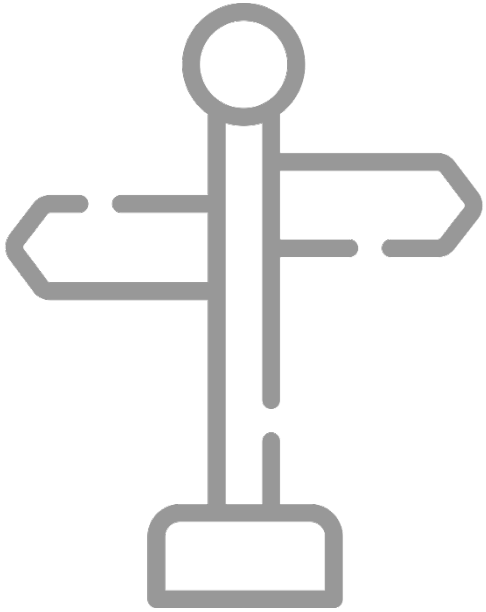
- *Anforderungen sind im Laufe der agilen Entwicklung zunehmend*
  - *Systemdesign und Architekturentwurf tendieren zu einem Konglomerat*
  - *Systematik geht verloren, es geht um die Einhaltung des nächsten Sprints bzw. Release*
- *Sicherheit als nichtfunktionales Thema wird temporär und später im Projekt involviert*
  - *Design soll „schnell sicher gemacht werden“*
  - *Sicherheit „erzeugt nur Aufwände und verdient kein Geld“*
  - *Sicherheit wird als Blocker der Feature-Entwicklung wahrgenommen*

Security by Design und agile Entwicklung

- Scrum Guide: Spielregeln beinhalten kein Security
- Hacking (evil user, malicious user) als Bestandteil der Tests
  → ausreichend für Design?

→ Analyse Angreifer und Analyse Security Assets eines Systems?

→ Architektur-bezogene Security Entscheidungen?



Resümee

Einordnung

Widerspruch?

- Agiles Vorgehen setzt auf nur inkrementelle Erkenntnisse
- Ergänzungen mit agilem Charakter ist grundsätzlich schwierig

→ Zwei harmonierende Ansätze notwendig, welche?

Integration?

- Integration wäre grundsätzlich notwendig
- Das **wie** ist bei agil schwierig!

→ **Gesamtsichteigenschaften** vorschalten?

Anmerkung:

- DevOps (Infrastrukturprozess) und DevSecOps (Security für DevOps) als Zusatzmodelle für agiles Vorgehen (Prozess) steigern Komplexität. Lösen nicht das Problem

Einordnung

Security ist eine Eigenschaft einer Architektur

- Betrifft äußere Hülle (global)
  → Kommunikationsschnittstellen
  → Daten, Parameter, Kommandos
  → Zugriffsrechte, Authentifizierung, Verschlüsselung, Signaturen

- Adressiert spezifische Komponenten und Verwaltungseinheiten

Agile Prozesse

- Definieren keine Kern-Requirements
- Führen keine Gesamtanalyse durch
- Leiten weder Gesamt- noch Kernarchitektur ab

→ Analyse Architekturbezogene Security als vorgeschalteter Schritt denkbar


Danke für Ihre Aufmerksamkeit!