HTWK

IM Fakultät Informatik und Medien

Sicherheitsaspekte in der (agilen) Softwareentwicklung

Finn Hoedt, Maurice Raymond Putzger, Bastian Wecke
Informatik und Medien, Modul Software Engineering (Prof. Dr. Andres Both)

23. JANUAR 2025

Einleitung
Szenario

Wir, das junge und moderne Startup namens "**SafeAgileTech**", wollen für die Medizinbranche eine Anwendung zur Verwaltung von Patientendaten in Echtzeit entwickeln.

Wir arbeiten natürlich **agil** und unsere Anwendung muss absolut **sicher** sein.

Einleitung

Was ist Sicherheit?

Vertraulichkeit

- Schutz von Informationen vor unbefugtem Zugriff [1]

Integrität

- Daten sind konsistent und unverändert, außer sie werden bewusst verändert [1]

Verfügbarkeit

- Garantiert das Informationen stets zugänglich sind [1]
- Software bleibt trotz gezielten Angriffen stabil und erreichbar [4]

Zuverlässigkeit

- Systeme und Software verhalten sich erwartungsgemäß [1]

Probleme Security in Agile

[4]

- Zeitintensive Sicherheitsmaßnahmen passen oft nicht in kurze Iterationen
- Sicherheitsaspekte werden zugunsten kurzer Release-Zyklen häufig vernachlässigt

Probleme Security in Agile

[4]

- Sicherheitsbewertungen erfordern umfangreiche Dokumentationen
- Funktionale Tests decken nicht alle Sicherheitsanforderungen und Angriffsszenarien ab
- Entwickler als Sicherheitsprüfer mindern die Objektivität der Bewertungen
- Mehr Entwicklungsaufwand und -kosten durch Sicherheitsmaßnahmen

Shift Left
Grundidee

- Verschiebung von Aktivitäten (z.B. Testing, Sicherheit) in die **frühen Phasen** des Softwareentwicklungszyklus (SDLC) [5]
- Ursprünglich im Software-Testing-Bereich eingeführt → Shift Left Testing [5]
- Geprägt durch Larry Smith (2001) [6]

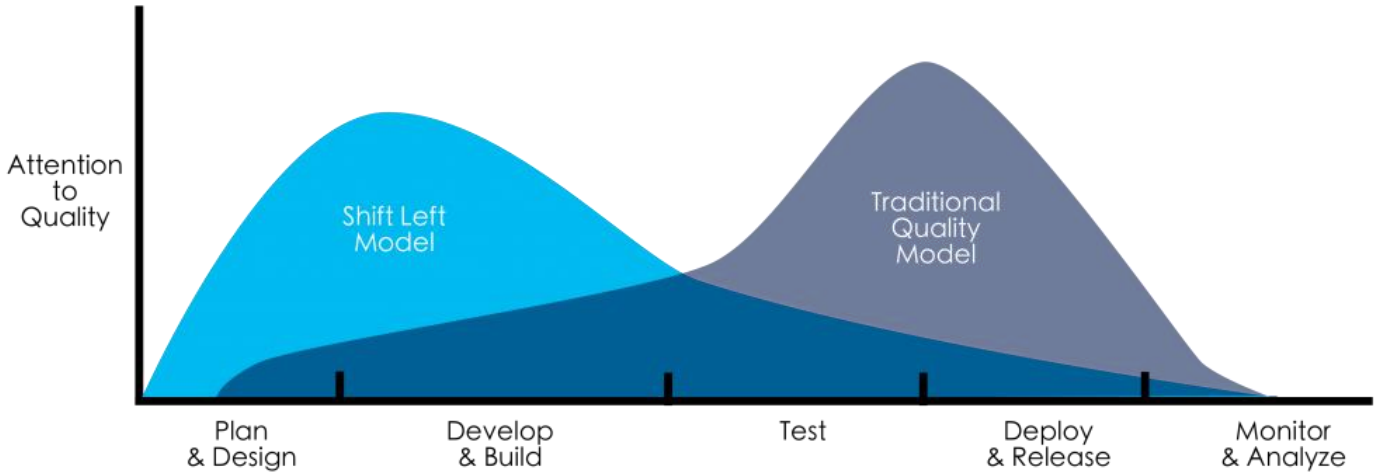

Abb. 1 zeitliche Verschiebung [7]

Shift Left Security

- Verschiebung von **sicherheitskritischen Prozessen** in die früheren Phasen der Entwicklung [5]
- Sicherheitsmaßnahmen als täglicher Entwicklungsprozess und nicht als nachgelagerter Schritt [5]

>> **SafeAgileTech**

- Sicherheitslücken gar nicht oder am Ende der Entwicklung zu entdecken wäre für die Patientendaten riskant
- Wir shiften Security nach links, jedoch wie?

Shift Left
Arten – Agile/DevOps

- Tests in jedem Zyklus (Sprint) und somit in frühen Entwicklungsphasen
- **Automatisierung** von Testfällen in einer **CI/CD**
- Enge Zusammenarbeit von Developer, Operational/Testing und Security Teams
- Tests schon vor der Implementierung (TDD)

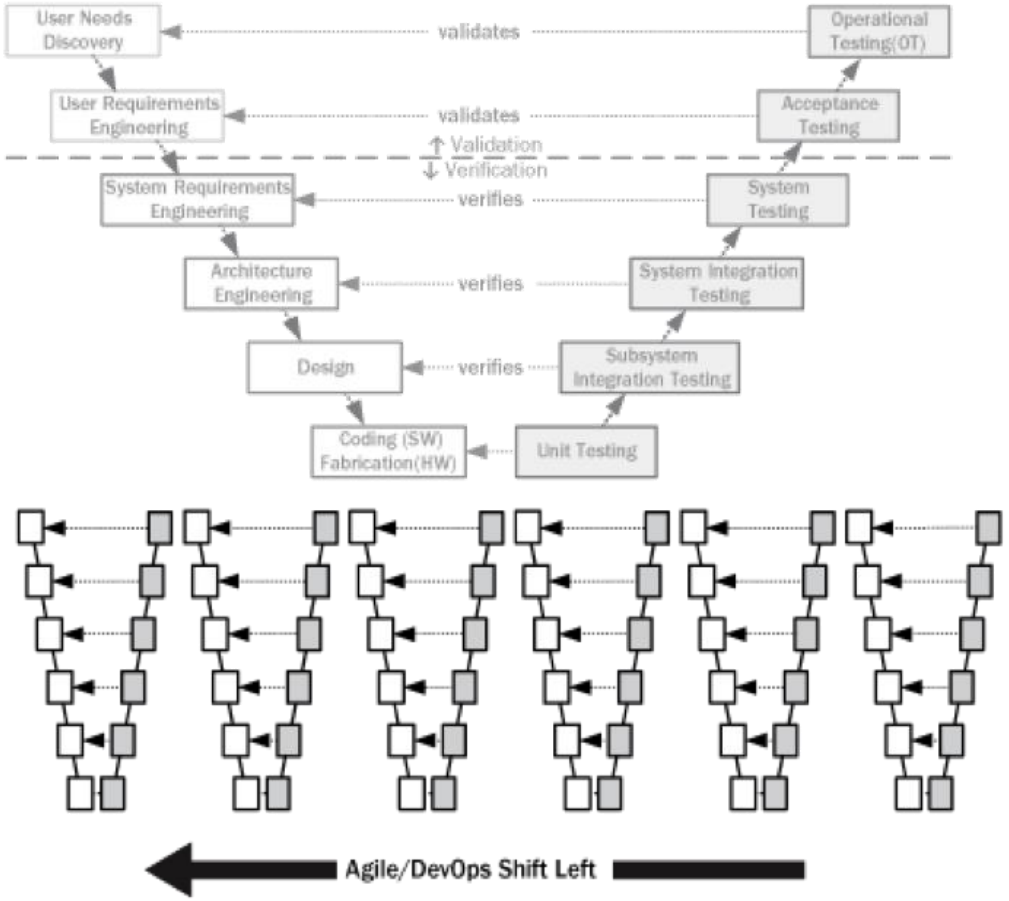

Abb. 2 Agile/DevOps Shift Left [8]

[7]

Shift Left
Arten – Model-Based

- Fehler/Schwachstellen in der Planungs- und Designphase zu erkennen
- (automatische) Testfallerzeugung aus Systemmodellen
- Systemmodelle = UML, Datenflussdiagramme etc.

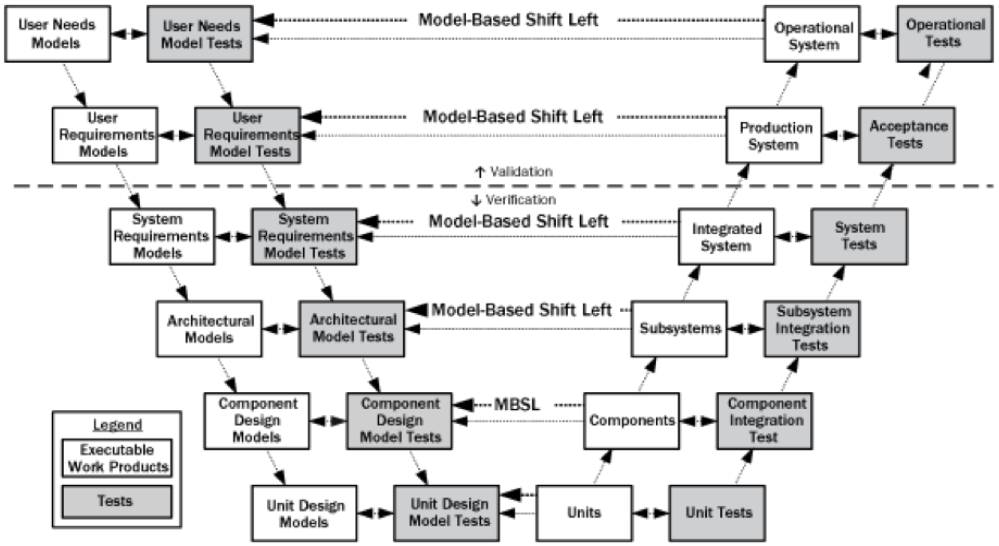

Abb. 3 Model-Based Shift Left [8]

[7]

Shift Left
Konzepte zur Umsetzung

Sicherheitsanforderungen nach links schieben mithilfe von ...

- Continuous Integration and Development (**CI/CD**)
  - Sicherheitstests in der gesamten Pipeline hinweg

- Development, Security and Operations (**DevSecOps**) [9]
  - Erweiterung von DevOps
  - Ziel: Agilität der DevOps Teams beibehalten und gleichzeitig Sicherheit gewährleisten
  - Enge Zusammenarbeit mit Entwicklern

- Automatisierung von Sicherheitstests
  - Static Application Security Testing (**SAST**)
  - Dynamic Application Security Testing (**DAST**)

Application Security Testing

[12]

Static Application Security Testing (SAST):

- "White-Box"-Methode
- Analyse des Quellcodes auf Sicherheitslücken
- Keine laufende Anwendung erforderlich
- Erkennt Sicherheitslücken früh in der Entwicklungsphase
- Hilft Entwicklern, Sicherheitsprobleme proaktiv zu beheben

Dynamic Application Security Testing (DAST)

- "Black-Box"-Methode
- Simulation von Angriffen auf die Anwendung
- Erkennt Schwachstellen, die nur zur Laufzeit auftreten
- Keine Kenntnis des Quellcodes erforderlich

Application Security Testing

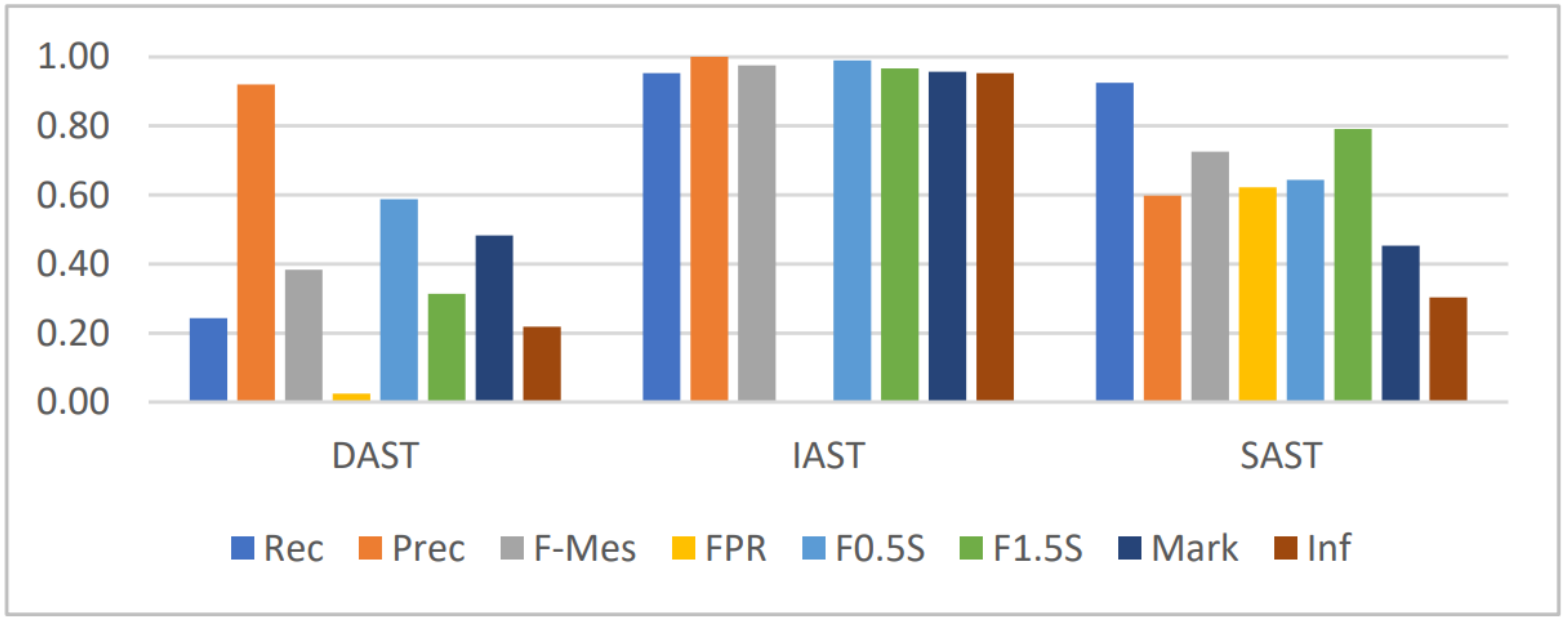

Abb. 4 Vergleich AST-Tools ohne Kombination [12]

Application Security Testing

[12]

Kriterien	SAST	DAST
Typ	White-Box	Black-Box
Analyse	Quellcode	Anwendungen und APIs
Quellcodezugriff	Ja	Nein
Sprachabhängig	Ja	Nein
Einsatzzeitpunkt	früh	spät
Ergebnisse	hoher Recall & viele FP	niedriger Recall & hohe Precision

Shift Left
Vorteile

- Späte Änderungen am Code sind teuer >> frühe Erkennung von Fehlern/Mängeln reduziert Kosten [10]

„Bugs are cheap when caught young“ – Larry Smith [6]

- Genug Zeit für Validierung und Umsetzung von Sicherheit [5]
- Zeit-Einsparungen durch Automatisierung [7]
- Parallelisierung von Security, QA und Development [5]
- Verbesserte Kommunikation [9]

>> Sichere Software

S4S
Scrum4Safety

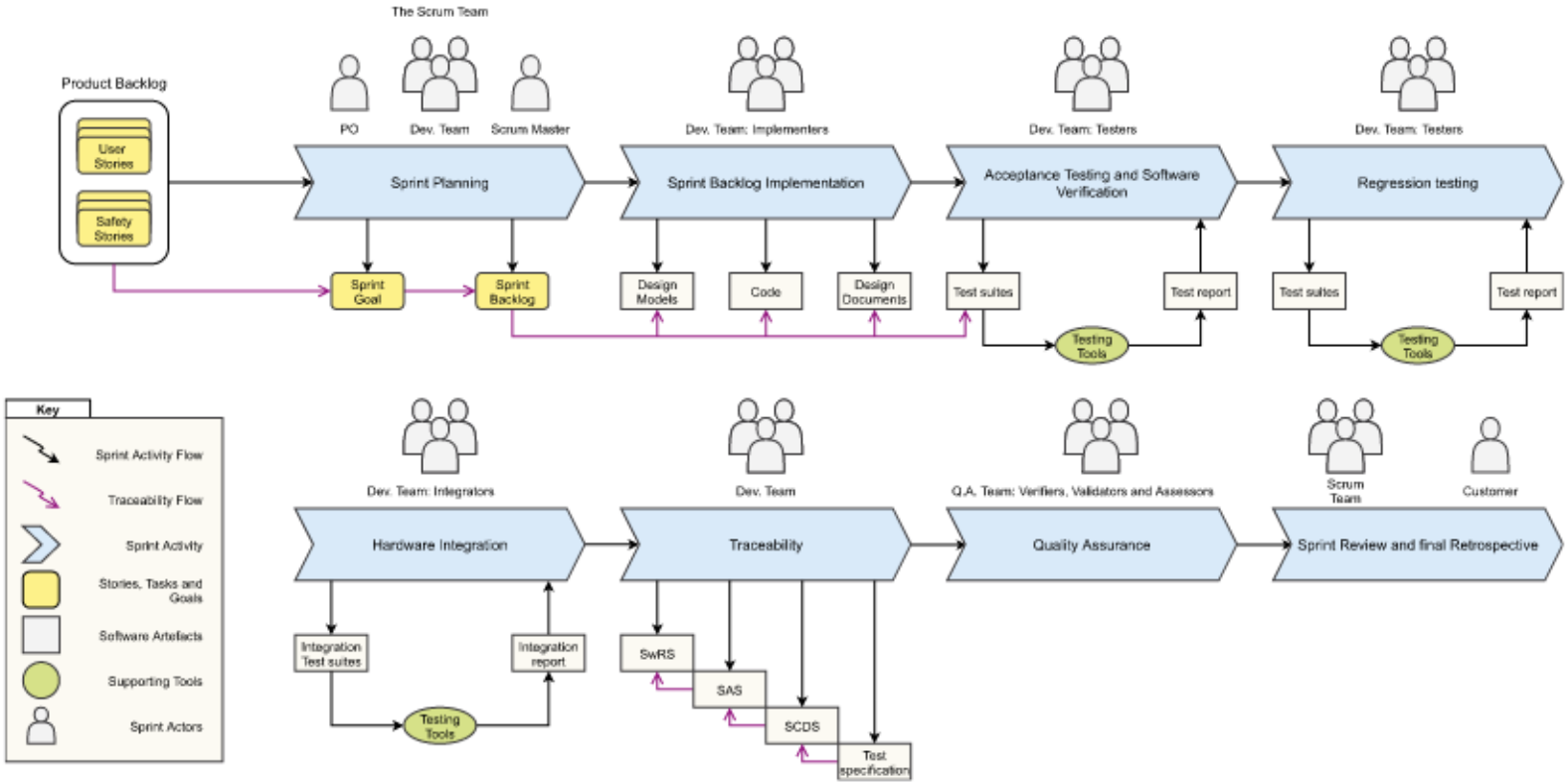

Abb. 5 S4S Safe-Sprint Struktur [2]

Finn Hoedt, Maurice Raymond Putzger, Bastian Wecke
Informatik und Medien, Modul Software Engineering (Prof. Dr. Andres Both)
Hochschule für Technik, Wirtschaft und Kultur Leipzig

I-ITWK

S4S
Scrum4Safety

[2]

→ **Erweiterung des Scrum-Frameworks**

- Sprint Hardening
  - Iteration benötigt Benutzerdokumentation & Konformitätsnachweis
  - validierte Softwareversion
- Continuous Compliance
  - kontinuierliche Verifizierungs- und Validierungsaktivitäten (V&V)
  - kritische Fehler schnell entdecken
- Living Tracebility
  - jederzeit klare Rückverfolgbarkeit der Implementierung bezüglich der Design-Entscheidungen
  - dient der Zertifizierung

- **Safe-Sprint**
  - ➢ Ergebnis: Verifiziertes und Validiertes Software-Inkrement

Finn Hoedt, Maurice Raymond Putzger, Bastian Wecke
Informatik und Medien, Modul Software Engineering (Prof. Dr. Andres Both)
Hochschule für Technik, Wirtschaft und Kultur Leipzig

Ansätze aus Shift-Left

[2]

Implementierte Ansätze

- Aufnahme von Sicherheitsbestimmungen in den Backlog durch Safetystories
- frühe und kontinuierliche Einbindung von Sicherheitsüberprüfungen in den Entwicklungszyklus
- Durchführung von Testmaßnahmen (SAST, DAST) durch DevSecOps
- Automatisierung und Einbindung in CI/CD

Finn Hoedt, Maurice Raymond Putzger, Bastian Wecke
Informatik und Medien, Modul Software Engineering (Prof. Dr. Andres Both)
Hochschule für Technik, Wirtschaft und Kultur Leipzig

S4S
Anwendbarkeit

[2]

→ **Überprüfung der Anwendbarkeit in einem Forschungsprojekt in der Eisenbahnindustrie**

- **Effizienz**
  - 4-Wochen Safe-Sprints
  - 75% der Zeit Verifikation und Validierung
- **Sicherheit**
  - Fehler und Sicherheitslücken wurden durch kontinuierliche Tests frühzeitig entdeckt und behoben
- **Transparenz**
  - Dokumentation der Entwicklungsaktivitäten wurden kontinuierlich erweitert, um Zertifizierung zu erleichtern

Finn Hoedt, Maurice Raymond Putzger, Bastian Wecke
Informatik und Medien, Modul Software Engineering (Prof. Dr. Andres Both)
Hochschule für Technik, Wirtschaft und Kultur Leipzig

SAFe
Scaled Agile Framework

[11]

→ **agile Methoden im großen Maßstab**

- Team Level
  - ➢ Scrum Teams
- Program Level
  - ➢ Agile Release Train (ART)
- Program Increment (PI)
  - ➢ PI-Planung
  - ➢ mehrere Iterationen
  - ➢ System-Demo

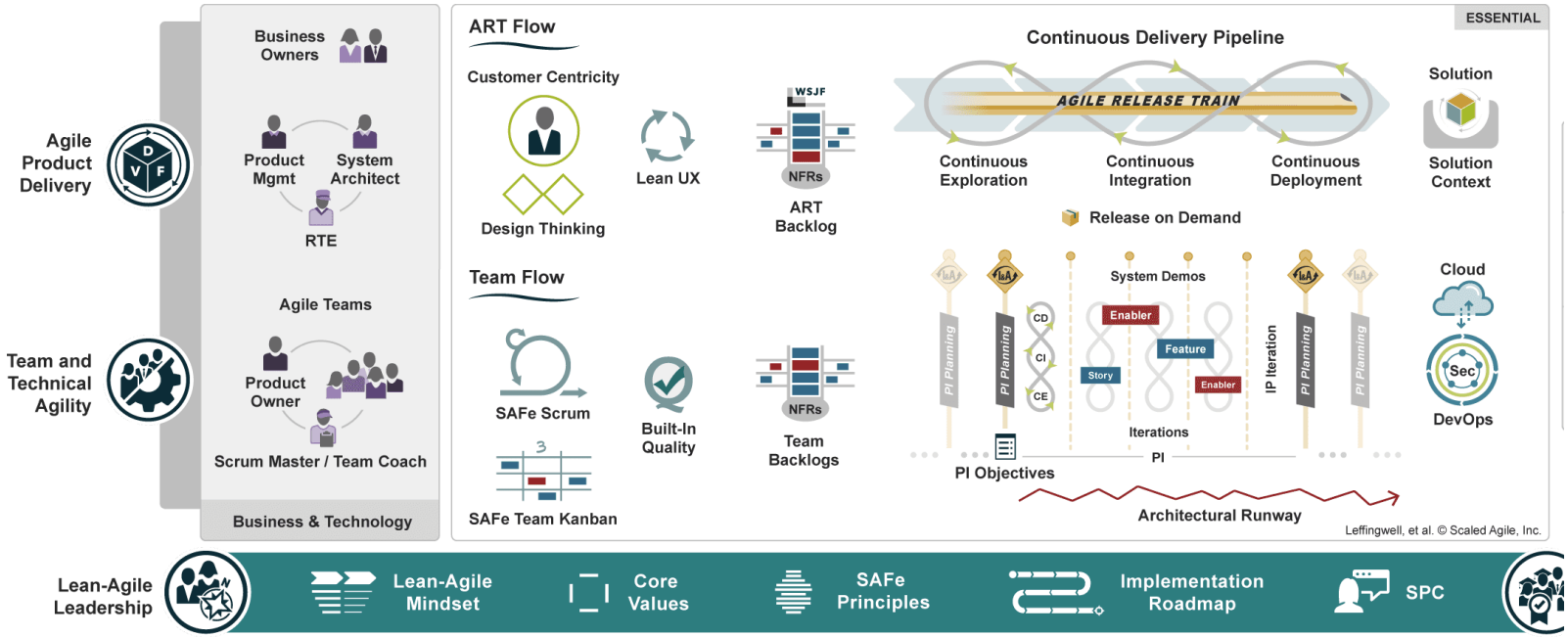

Abb. 6 SAFe Aufbau [11]

Finn Hoedt, Maurice Raymond Putzger, Bastian Wecke
Informatik und Medien, Modul Software Engineering (Prof. Dr. Andres Both)
Hochschule für Technik, Wirtschaft und Kultur Leipzig

I-ITWK

S^2C–SAFe
Security Standard Compliant Scaled Agile Framework

[3]

→ **Erweiterung des SAFe-Frameworks**

- integriert **IEC 62443-4-1** Sicherheitsanforderungen
- **Continuous Security**

- **Security Requirements (SR)**
  - Sicherheitsanforderungen im Backlog
  - stärkere Einbindung des Product Owner und System Architekten in Sicherheitsaspekte
- **Secure Implementation (SI)**
  - Einführung Coding Standards in PI-Planung
  - Sicherheitsaspekte in Definition of Done
- **Security Verification & Validation (SVV)**
  - unabhängige Tester-Rollen
  - Integration von Sicherheitstests

Finn Hoedt, Maurice Raymond Putzger, Bastian Wecke
Informatik und Medien, Modul Software Engineering (Prof. Dr. Andres Both)
Hochschule für Technik, Wirtschaft und Kultur Leipzig

I-ITWK

S^2C–SAFe
Ansätze aus Shift-Left

[3]

Implementierte Ansätze

- Berücksichtigung von Sicherheit in allen Prozessen (CS)
- Frühe Integration von Sicherheitsanforderungen (SR) durch Aufnahme von Sicherheitsanforderungen den Backlog
- Beachtung von Sicherheitsanforderung bei der Implementierung durch Aufnahme in Definition of Done
- Kontinuierliche Sicherheitsüberprüfungen (SVV)

Finn Hoedt, Maurice Raymond Putzger, Bastian Wecke
Informatik und Medien, Modul Software Engineering (Prof. Dr. Andres Both)
Hochschule für Technik, Wirtschaft und Kultur Leipzig

S^2C–SAFe
Anwendbarkeit

[3]

→ **Überprüfung der Anwendbarkeit in einem Forschungsprojekt durch Siemens**

- **Erkenntnisse**
  - praktikable Überbrückung der Diskrepanz zwischen agilen Methoden und Sicherheitsanforderungen
  - erleichterte Kommunikation zwischen Entwicklung und Verifikation & Validierung
- **Herausforderungen & Risiken**
  - fehlende Sicherheitsexpertise
  - Komplexität der Sicherheitsanforderungen
  - Unterschiedliche Wahrnehmungen bezüglich Sicherheit
  - Priorisierung von Sicherheitsanforderungen

Finn Hoedt, Maurice Raymond Putzger, Bastian Wecke
Informatik und Medien, Modul Software Engineering (Prof. Dr. Andres Both)
Hochschule für Technik, Wirtschaft und Kultur Leipzig

Diskussion

1. **Sinnvoll, aber schwer umzusetzen**
   - Shift Left Security hilft Fehler und Sicherheitslücken frühzeitig zu erkennen
   - Investitionen in Schulungen, Tools und Automatisierung notwendig
   - Shift Left ist keine Methode, sondern eine Richtlinie die projektspezifisch angepasst werden muss

2. **Abwägung von Freiheit und Sicherheit**
   - Freiheit des Entwicklungsteams wird eingeschränkt zugunsten von Sicherheit
   - Verantwortung für Sicherheit wird gleichzeitig im ganzen Team geteilt

3. **Richtig umsetzen oder gar nicht**
   - Analog zu agilem Arbeiten
   - Problematisch: In sicherheitskritischen Projekten kann falsches Verständnis von Shift Left Security katastrophale Folgen bewirken

Zusammenfassung

Integration von Sicherheit in agile Prozesse ist schwierig:

- kurze Iterationen ⟷ zeitaufwändige Sicherheitsprozesse [4]

Shift-Left:

- Verschiebung von **sicherheitskritischen Prozessen** die frühen Phasen des Softwareentwicklungszyklus (SDLC) [5]
- Sicherheitslücken und Bugs werden früher erkannt

Sicherheits-Frameworks:

- Implementieren Konzepte aus Shift-Left
- erweitern um branchenspezifische Anforderungen

Literaturverzeichnis

[1] - Nägele, S., Schenk, N., & Matthes, F. (2023). The Current State of Security Governance and Compliance in Large-Scale Agile Development: A Systematic Literature Review and Interview Study. *2023 IEEE 25th Conference on Business Informatics (CBI)*, 1–10. https://doi.org/10.1109/CBI58679.2023.10187439

[2] - Barbareschi, M., Barone, S., Carbone, R., & Casola, V. (2022). Scrum for safety: An agile methodology for safety-critical software systems. Software Quality Journal, 30(4), 1067–1088. https://doi.org/10.1007/s11219-022-09593-2

[3] - Moyón, F., Méndez Fernández, D., Beckers, K., & Klepper, S. (2020). *How to Integrate Security Compliance Requirements with Agile Software Engineering at Scale?* (pp. 69–87). https://doi.org/10.1007/978-3-030-64148-1_5

[4] - Oueslati, H., Rahman, M. M., & Othmane, L. ben. (2015). Literature Review of the Challenges of Developing Secure Software Using the Agile Approach. *2015 10th International Conference on Availability, Reliability and Security*, 540–547. https://doi.org/10.1109/ARES.2015.69

[5] - Dawoud, A., Finster, S., Coppik, N., & Ashiwal, V. (2024). Better Left Shift Security! Framework for Secure Software Development. *2024 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, 642–649. https://doi.org/10.1109/EuroSPW61312.2024.00078

[6] - Smith, L. (n.d.). Shift-Left Testing. *Dr. Dobb's*. Retrieved January 12, 2025, from http://www.drdobbs.com/shift-left-testing/184404768

[7] - Rani, V. S., Babu, D. A. R., Deepthi, K., & Reddy, V. R. (2023). Shift-Left Testing in DevOps: A Study of Benefits, Challenges, and Best Practices. *2023 2nd International Conference on Automation, Computing and Renewable Systems (ICACRS)*, 1675–1680. https://doi.org/10.1109/ICACRS58579.2023.10404436

[8] - Firesmith, D. (2015, September 5). Four Types of Shift Left Testing. *Four Types of Shift Left Testing*. https://web.archive.org/web/20150905082941/https://insights.sei.cmu.edu/sei_blog/2015/03/four-types-of-shift-left-testing.html

[9] - Rajapakse, R. N., Zahedi, M., Babar, M. A., & Shen, H. (2022). Challenges and solutions when adopting DevSecOps: A systematic review. *Information and Software Technology*, *141*, 106700. https://doi.org/10.1016/j.infsof.2021.106700

[10] - Boehm, B. W. (1984). Software Engineering Economics. In M. Broy & E. Denert (Eds.), *Pioneers and Their Contributions to Software Engineering: Sd&m Conference on Software Pioneers, Bonn, June 28/29, 2001, Original Historic Contributions* (pp. 99–150). Springer. https://doi.org/10.1007/978-3-642-48354-7_5

[11] - Knaster, R., Leffingwell, D. (2020) SAFe 5.0 Distilled: Achieving Business Agility with the Scaled Agile Framework. Boston: Addison-Wesley

[12] - Mateo Tudela, F., Bermejo Higuera, J.-R., Bermejo Higuera, J., Sicilia Montalvo, J.-A., & Argyros, M. I. (2020). On Combining Static, Dynamic and Interactive Analysis Security Testing Tools to Improve OWASP Top Ten Security Vulnerability Detection in Web Applications. *Applied Sciences*, Article 24. https://doi.org/10.3390/app10249119