fortiss

Wartung von Software proaktiv unterstützen

Aktuelle Forschungsthemen und Techniken

Peter Bludau
fortiss GmbH
Center for Code Excellence

Über fortiss

- **Rechtsform:** gemeinnützige GmbH
- **Gründung:** 2009
- **Eigentümer:** Freistaat Bayern (2/3) und Fraunhofer-Gesellschaft (1/3)

2
Standorte

150
Mitarbeiter

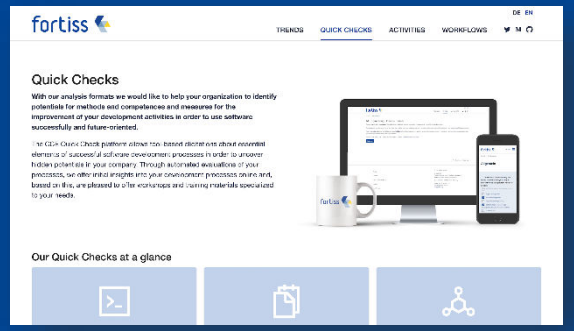
60
laufende Projekte

Mission

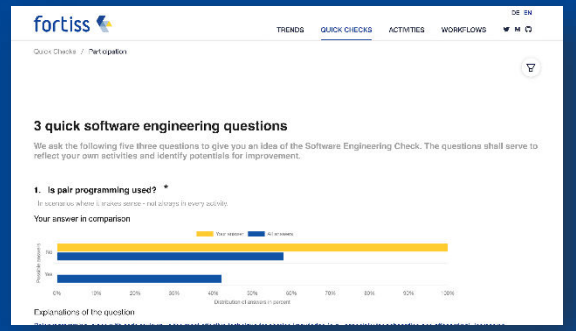
- **Forschung:** Wir sind Innovationstreiber für die Gestaltung der digitalen Transformation für Wirtschaft und Verwaltung in Bayern, indem wir digitale Technologien erforschen, entwickeln und umsetzen.
- **Transfer:** Wir bereiten Ausgründungen für die nachhaltige Verankerung von Ergebnissen in der Praxis vor. Unsere Forschungs-, Entwicklungs- und Transferprojekte sind Basis neuer Produkte, Dienstleistungen und Geschäftsmodelle.
- **Service:** Wir bieten Innovations- und Forschungsberatung und sind die Anlaufstelle und anerkannter Forschungs- und Ansprechpartner für die ansässige Wirtschaft und Verwaltung in Bayern zu allen Themen des digitalen Wandels.

Center for Code Excellence

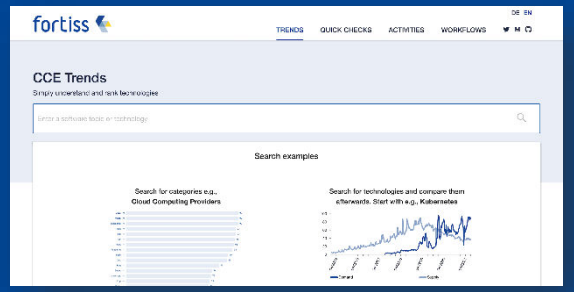
Herausragende Softwarequalität



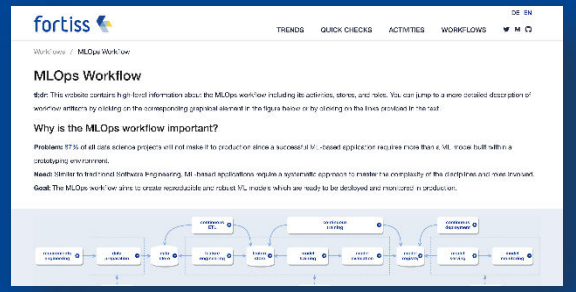
Quick Checks



Quick Check Bewertung



Technologietrends



Engineering Workflows

Wartbarkeit als wichtiges Qualitätsmerkmal

- Wartung ist nicht-funktional
- Wartung kostet Ressourcen

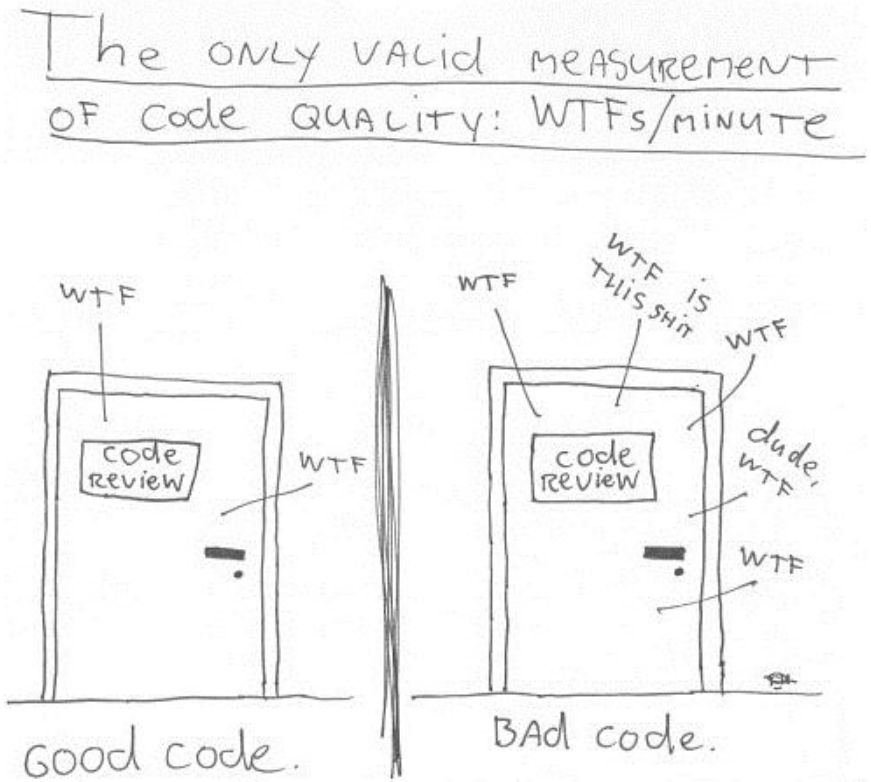
- Wartungsaufgaben:
  - Entwickler testen Software
  - Entwickler passen Software an
  - Entwickler finden und lösen Probleme in der Software

Bei Wartung steht der Entwickler im Vordergrund

Warum wartbare Software erstellen?

- Es gibt viele Metriken um Wartbarkeit von Systemen zu messen und Qualität sicher zu stellen
- Es gibt wissenschaftliche Publikationen, die zeigen, dass eine klare Architektur solche Qualitätsmetriken verbessern kann

- **Ziel: Entwickler unterstützen und entlasten**
  - Fehler reduzieren
  - Produktivität steigern
  - Frustration senken



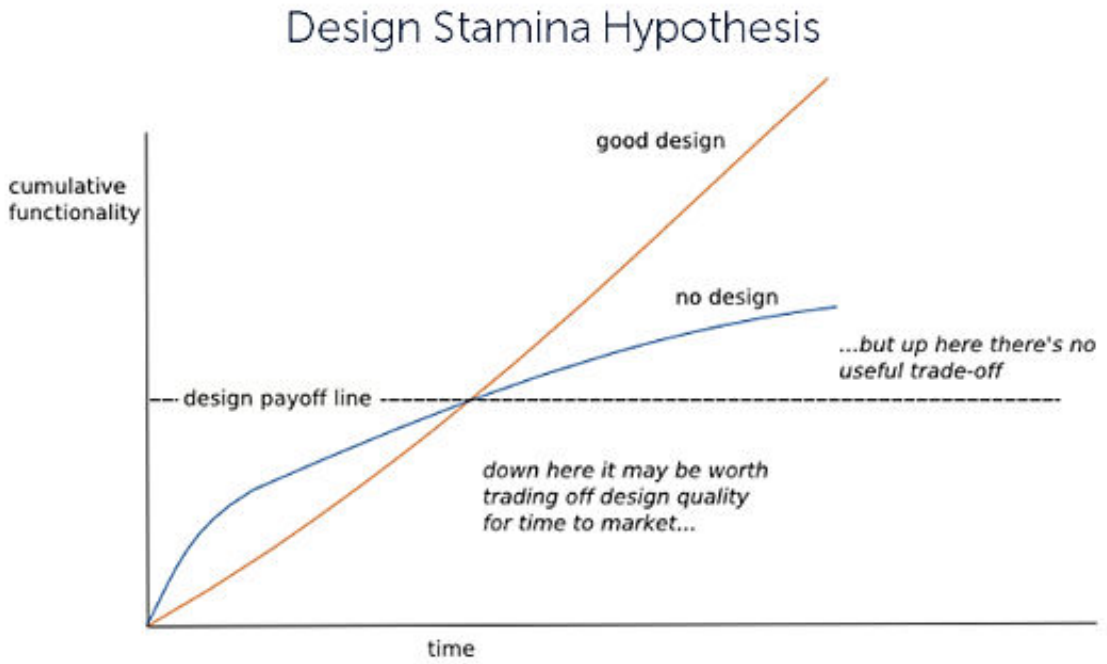
Architektur als Treiber für wartbare Software

Ab wann lohnt es sich Wartbarkeit in den Fokus zu stellen?

- Wartbare Software kostet initialen Aufwand
- Schlechter Code kostet Zeit und Nerven

- **Risiken**
  - Neue Funktionalität wird als aufwändig oder riskant wahrgenommen
  - Verbessernde Aufwände werden aufgeschoben
  - Technologieabhängigkeit
  - Niedrige Kohäsion, hohe Kopplung

- Es gibt einen Schnittpunkt zwischen Zeitersparnis und Anhäufung technischer Schulden



Source: https://martinfowler.com/bliki/DesignStaminaHypothesis.html

Entwickler proaktiv unterstützen

Forschungsperspektive

- **durch Architektur**
  - Unabhängigkeit von Frameworks und An-Systemen
  - Testbarkeit
  - Lesbarkeit
- **durch Prozesse**
  - Anforderungsmanagement
  - Dokumentation
- **durch Entwicklungsaktivitäten**
  - Überblick über den aktuellen Stand der Software
  - Kontextabhängige Code-Vorschläge und -Anpassungen

Entwickler proaktiv unterstützen

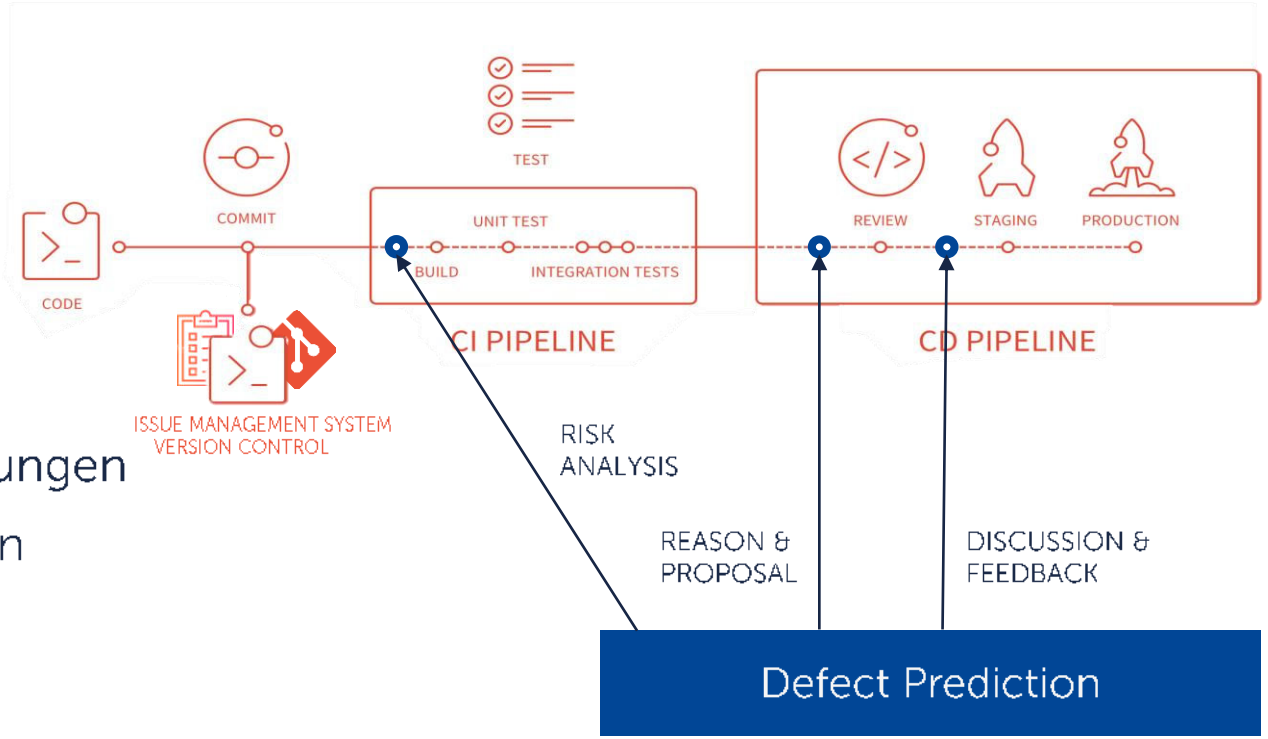
Entwicklungsaktivitäten - Anwendungsfälle

- **Beschleunigung des Feedbacks für Entwickler**
  - Fehlervorhersage und -behebung
  - Wartbarkeitsanalysen
- **Unterstützung bei der Erstellung, Durchführung und Analyse von Tests**
  - Testgenerierung
  - Testauswahl und Priorisierung
  - Fehlergruppierung und Ursachenanalyse
- **Verbesserung der Produktivität in Entwicklerteams**
  - Entwicklungsprozess- und Fortschrittsanalyse

Beschleunigung des Feedbacks für Entwickler

Fehlervorhersage

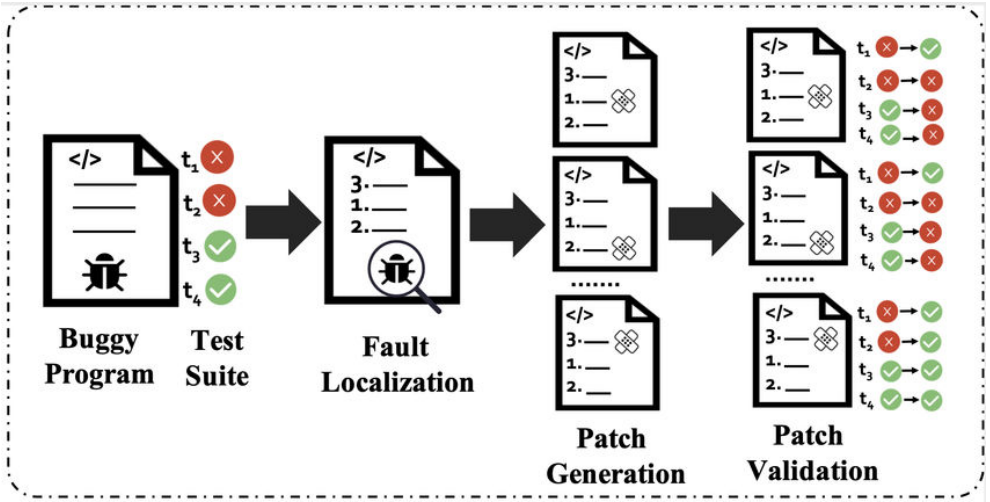
- **Problem**:
  - Begrenzte Zeit für Code Reviews
  - Unerfahrenheit mit Änderungen
- **Ansatz**:
  - Finden von potentiell fehlereinführenden Änderungen
  - Automatisierte Risikobewertung von Änderungen
- **Ergebnis**:
  - Priorisierung von Reviewaufwänden
  - Schnelleres Feedback durch Lokalisierung riskanter Stellen im Code



Beschleunigung des Feedbacks für Entwickler

Automatisierte Fehlerbehebung

- **Problem**:
  - Unkenntnis von Änderungen oder genauer Funktionalität
  - Mehrere fehlschlagende Tests
- **Ansatz**:
  - Analyse von ausgeführtem Code durch fehlschlagende Tests
  - Generierung und automatisierte Prüfung von Lösungen
- **Ergebnis**:
  - Schnelleres Feedback für fehlschlagende Tests
  - Automatisierte Vorschläge zur Lösung des Problems

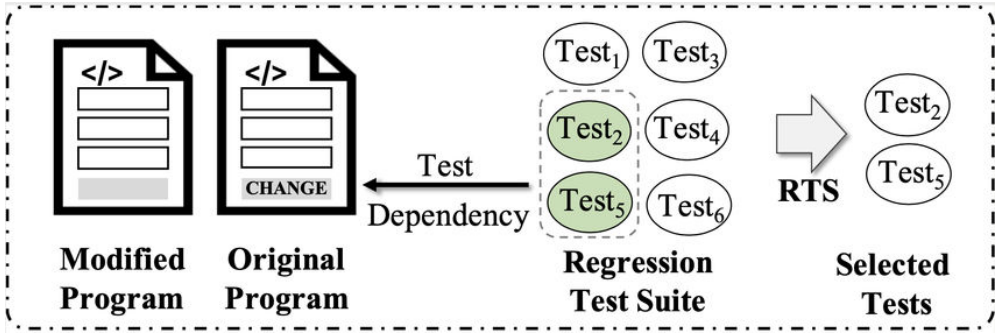


https://www.researchgate.net/figure/Workflow-of-test-based-automated-program-repair-generated-and-2-each-patch-validation_fig2_351656750

Unterstützung bei der Durchführung von Tests

Testauswahl und Priorisierung

- **Problem**:
  - Große, langsame und/oder teure Testsuite
- **Ansatz**:
  - Priorisierung von Tests nach Wert
  - Auswahl von Tests durch Kopplung der Testausführung mit geändertem Code
- **Ergebnis**:
  - Auswahl von auszuführenden Tests bei hoher Codeabdeckung für geänderten Code
  - Ressourcenersparnis
  - Reduzierung von "Flaky Tests"



https://www.researchgate.net/figure/Workflow-of-regression-test-selection_fig1_351656750

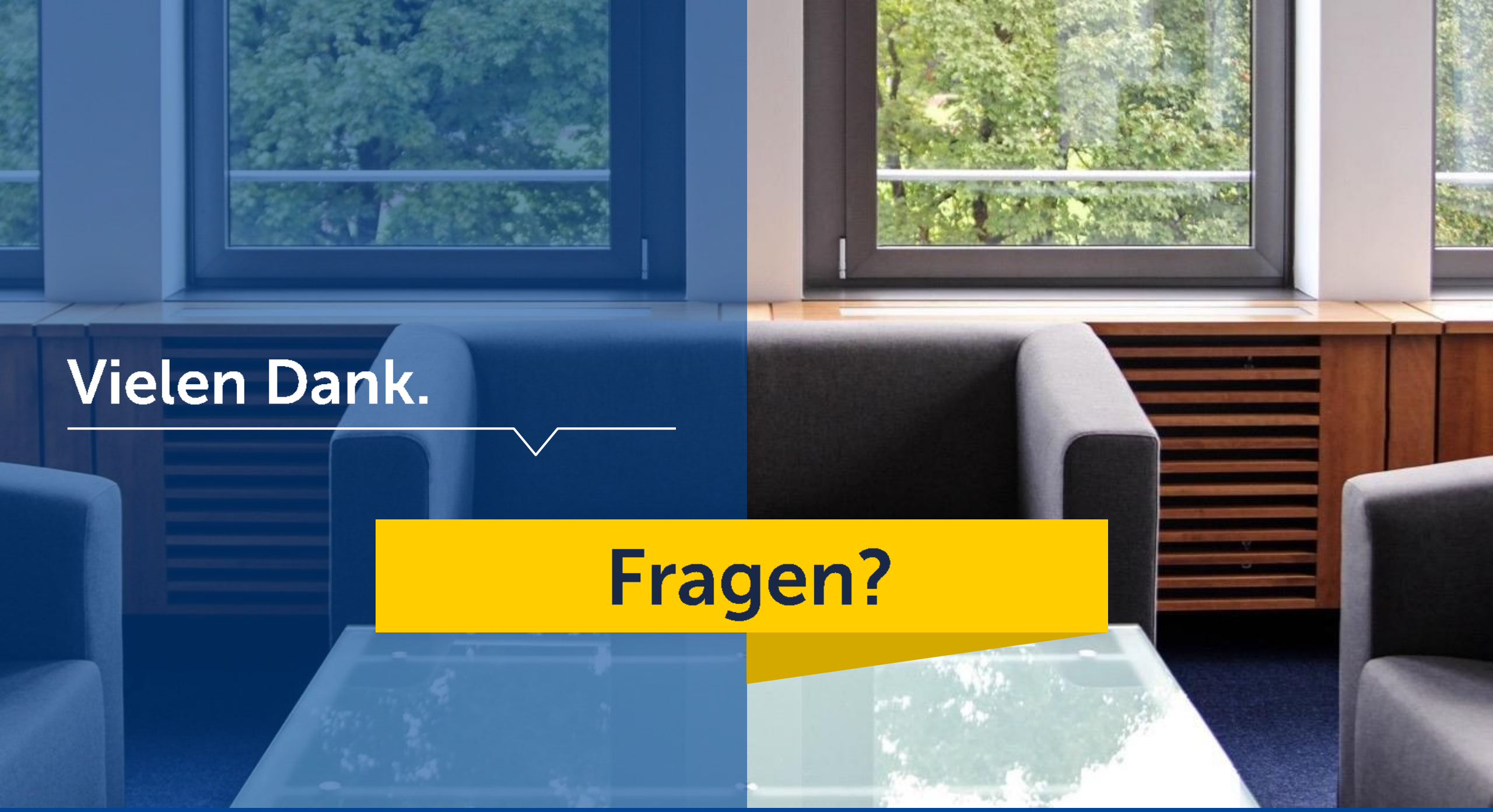
Wartung von Software proaktiv unterstützen

Fazit – Wichtigste Stakeholder sind Entwickler

- Clean Architecture unterstützt den Entwickler, gerade bei der Wartung langlebiger Software
- Clean Architecture befähigt weiterführende Maßnahmen zur Unterstützung der Entwickler

- Es gibt verschiedene Möglichkeiten die Wartbarkeit von Software zu unterstützen, viele davon sind aktiver Bestandteil der Forschung
- Fokus liegt auf der Unterstützung der Entwickler und Entwicklungsprozesse

- Qualität und Design der Software ist maßgeblich entscheidend für den Erfolg eines Produkts auf Entwicklungsseite und hat letztendlich damit auch Auswirkungen auf den Kunden haben


Vielen Dank.
Fragen?

Kontakt

Peter Bludau

fortiss GmbH
Guerickestraße 25
80805 München

cce.fortiss.org
bludau@fortiss.org

©2023