**Bitkom AK Software Engineering & Software Architektur**
**Datum:** 15. September 2025 | **Ort:** Webkonferenz

# Erfolgsfaktoren für die Modernisierung von Software in Produkten und IT

Software altert – und mit ihr wachsen technische Schulden, Wartungsaufwände und Abhängigkeiten von veralteten Technologien. Unternehmen stehen vor der Herausforderung, bestehende Produkte und Systeme zu modernisieren, ohne Kunden und Nutzer zu verlieren.

Zusammengefasst hier sind Erfahrungen, Erkenntnisse und Best Practices rund um die Modernisierung von Software – von Architektur- und Technologieentscheidungen über den Umgang mit Domänenwissen bis hin zu pragmatischen Ansätzen für Modernisierungsprojekte.

## 1. Beyond Code: Strategien und Best Practices bei der Softwaremodernisierung

**Jan Hansmann, Systemum GmbH & Co. KG**

### Einleitung:

Historisch gewachsene Software birgt hohen Wert, ist jedoch oft schwer modernisierbar. Eine vollständige Neuentwicklung ist meist unwirtschaftlich oder riskant. Stattdessen müssen Unternehmen Strategien finden, wie sie technische Schulden, veraltete Technologien und steigende Markt- sowie Regulierungsanforderungen adressieren können.

### Key Takeaways:

- Verteilung IT-Budget lt. Lündendonk-Studie:
  - 1/3 Erhalt Bestands-IT
  - 1/3 Erneuerung Bestands-IT
  - 1/3 Innovation

- Softwareerosion führt unweigerlich zu Modernisierungsbedarf.
- Treiber sind Alterung, fehlende Dokumentation, technologische Obsoleszenz, neue Marktanforderungen und strengere Regulatorik.
- Cybersecurity und regulatorische Anforderungen sind aktuell zentrale Treiber.
- Kurzfristige Kompromisse bei Qualität und Wartung erzeugen instabile und unübersichtliche Legacy Software.
- Kritisch: Wissen über Fachlichkeit ist oft nicht vorhanden bzw. dokumentiert
- Brownfield: schnelle Ergebnisse, aber Altlasten bleiben. Fokussierung auf das Modernisierungsziel ist wichtig.
- Greenfield: sauberer Neustart, jedoch Risiko von Zeit-/Kostenaufwand und unzureichendem Wissen.
- Modernisierungsprojekte brauchen eine abgestimmte Strategie mit festem Scope und business-orientiertem Ziel.

## 2. Refactor, Rearchitect, Rebuild – Was macht man wann?

**Richard Gross, MaibornWolff GmbH**

### Einleitung:

Historisch gewachsene Software birgt hohen Wert, ist jedoch oft schwer modernisierbar. Eine vollständige Neuentwicklung ist meist unwirtschaftlich oder riskant. Stattdessen müssen Unternehmen Strategien finden, wie sie technische Schulden, veraltete Technologien und steigende Markt- sowie Regulierungsanforderungen adressieren können.

### Key Takeaways:

- Jede Software hat einen inhärenten Mehrwert – sonst bräuchte es sie nicht.
- Haupttreiber der Modernisierung sind:
  - Senkung von Betriebs-/Wartungskosten
  - Schaffung neuer Handlungsoptionen
- Tiefergehende Modernisierung bedeutet höhere Anfangskosten, aber auch größere langfristige Flexibilität und Einsparungen.
- Wardley Map hilft bei der Einordnung von Kosten vs. Optionen.
- Transparenz in Systemlandschaften und Abhängigkeiten ist die Grundlage für die Entscheidung: Eigenentwicklung, Standardsoftware oder Outsourcing.
- Business Value steigt linear mit der »Systemhygiene« (Quasar-Ansatz).
- Modernisierung und Weiterentwicklung gehören ins gleiche Team – keine Trennung. 3. Evolutionäre Softwareentwicklung – So gelingt nachhaltige Softwaremodernisierung ohne Big-Bang-Risiko

## 3. Evolutionäre Softwareentwicklung – So gelingt nachhaltige Softwaremodernisierung ohne Big-Bang-Risiko

**Markus Schüring, Arvato Systems Digital GmbH**

### Einleitung:

Klassische »Alles-neu«-Ansätze scheitern häufig an langen Projektlaufzeiten und fehlender Anpassungsfähigkeit. Ein evolutiver, iterativer Modernisierungsprozess ermöglicht hingegen schnelle Ergebnisse und eine kontinuierliche Weiterentwicklung – ohne Big-Bang-Risiko.

### Key Takeaways:

- Schwächen in Schnittstellen, Code und Dokumentation wirken unmittelbar auf Personal, Kosten und Flexibilität.
- Unternehmen stehen vor der Wahl: Stillstand (Kosten steigen), risikoreiche Neuentwicklung oder evolutionäre Modernisierung.
- Langfristiger Fokus ist entscheidend, um Modernisierung nicht ins Stocken geraten zu lassen.
- Flexibel und agil: evolutionäre Modernisierung ermöglicht schnelle Ergebnisse, birgt aber die Gefahr, dass Altlasten bestehen bleiben.
- Architekturentscheidungen sind der Grundstein: von Runtime und Prozessen über Cloud-Services bis zu Security- und Souveränitätsaspekten.

## Zusammenfassung

Die Modernisierung von Legacy-Systemen ist komplex und erfordert eine klare Strategie. Ob Refactoring, Rearchitecting oder ein vollständiger Rebuild – die Wahl hängt vom aktuellen Zustand des Systems und den zukünftigen Anforderungen ab. Praxisbeispiele zeigen, dass die größten Herausforderungen oft weniger in der Technik, sondern vielmehr in Organisation und Prozessen liegen. Moderne Werkzeuge, darunter auch KI-gestützte Ansätze, können helfen, den Transformationsprozess effizienter zu gestalten und Risiken zu minimieren.

## Fazit:

Eine erfolgreiche Modernisierung gelingt dann, wenn technische Maßnahmen mit einer ganzheitlichen Strategie kombiniert werden. Statt eines riskanten »Big Bang« führt ein evolutionärer Ansatz Schritt für Schritt zu nachhaltigen Ergebnissen.

**Felix Ansmann**
Referent Software &
IT-Services
T 030 27576-098
f.ansmann@bitkom.org