Teil II: Evolution von Architekturen

Es ist bekannt, dass der größte Teil der IT-Aufwendungen auf die Wartung und Evolution bestehender Software-Systeme entfällt (zum Beispiel [ER03, S. 161]). Entsprechend hoch sind Umfang und Anzahl der Tätigkeiten von Software-Entwicklern in diesem Bereich. Wir verstehen hier unter Software-Evolution alle Tätigkeiten, die zur Anpassung von bestehender Software an neue oder geänderte Anforderung dienen. Die Evolution von Software-Architekturen umfasst also alle Tätigkeiten, die der Anpassung von Software-Architekturen an neue oder geänderte Anforderungen dienen. Diese Tätigkeiten gliedern wir in den folgenden Kapiteln:

Kapitel 8: Grundlagen der Evolution von Software-Architekturen

Dieses Kapitel geht auf die Evolution von Architekturen *im Kleinen* ein, nämlich auf sogenannte Refactorings, die eine Architektur anpassen, damit neue oder geänderte Anforderungen umgesetzt werden können. Das Refactoring selbst realisiert noch keine neuen oder geänderten Anforderungen, ändert also die Funktionalität des Systems nicht. Nach einer allgemeinen Einführung in das Thema Evolution werden Refactorings vorgestellt und am Beispiel erläutert. Dabei wird besonderer Wert auf die Validierung der funktionalen Äquivalenz des Systems vor und nach Durchführung des Refactorings durch Testen gelegt.

Kapitel 9: Reverse Engineering von Software-Architekturbeschreibungen

Dieses Kapitel behandelt das Problem, dass eine explizite Software-Architekturbeschreibung zwar vorteilhaft für die Evolution (und insbesondere wie im nächsten Kapitel gezeigt für die Migration) von Software ist, leider aber nur allzu häufig für bestehende Software-Systeme eine explizite Architekturbeschreibung fehlt oder unvollständig bzw. inkonsistent ist. Daher beschäftigt sich dieses Kapitel mit der Extraktion von Architekturinformation aus bestehendem Code. Dazu werden verschiedene Ansätze diskutiert, die mit dem Problem umgehen, dass eine Architektur keine reine Abstraktion vom Code ist (d. h. nur durch Weglassen von Informationen des Codes erhalten werden kann), sondern vielmehr enthält eine Architektur auch Informationen, die nicht im Code zu finden sind.

Kapitel 10: Migration von Altsystemen

Dieses Kapitel beschreibt ein Muster für das Vorgehen bei der schrittweisen Migration einer monolithischen Architektur zu einer dienstorientierten Schichtenarchitektur. Dieses Szenario ist im Bereich betrieblicher Informationssysteme häufig anzutreffen, zum einen, da sehr viele Altsysteme über keine klare Trennung zwischen Darstellung, Geschäftslogik und Datenhaltung verfügen (diese Trennung wird weder bei COBOL noch in 4GL-Sprachen unterstützt), zum anderen, da eine Schichtenarchitektur heute als sinnvoll für die meisten Client-Server-Systeme angesehen wird.