Towards a Generic Framework for the Evaluation of Component-Based Software Architectures

Steffen Becker, Viktoria Firus, Simon Giesecke, Willi Hasselbring, Sven Overhage, *Ralf Reussner*

Universität Oldenburg, Department für Informatik
Universität Augsburg, Lehrstuhl für Wirtschaftsinformatik

Agenda

Evaluation von Software-Architekturen

Methodische Voraussetzung Komponentenbasierung

Sichten auf Software-Architekturen

Ein generisches Vorgehensmodell

Klassifikation existierender Methoden

Beispiel für die Evaluation von Software-Architekturen

Ausblick

Evaluation von Software-Architekturen

- Ermöglicht die Vorhersage von Anwendungseigenschaften
  - vor der Implementierung / Beschaffung von Komponenten
  - bereits während des Entwurfs (ingenieurmäßiges Vorgehen)
  - → sichert die Konsistenz zwischen Entwurf und Anforderungen
- Entwicklung eines generischen Frameworks
  - Beschreibung des Evaluationsprozesses
  - Vergleich und Klassifikation existierender Methoden
  - → Ziel: Methodik für die Evaluation von Software-Architekturen

Klassifikation von Evaluationsverfahren

- Evaluation: Bewertung von Alternativen, keine „beweisbaren" Vorhersagen (z.B. Echtzeitschranken)
  - Einfluss der Implementierungsentscheidungen machen i.d.R. exakte Vorhersagen unmöglich
- Klassifikation von Evaluationsverfahren für Entwickler gemäß genutzter Architektursichten
  - welches Verfahren ist zu meinem Prozess kompatibel?
  - welche Änderungen muss ich an meinem Prozess vornehmen, um ein Verfahren einzusetzen?
  - welche Art von Entwurfsalternativen können vergleichend bewertet werden?

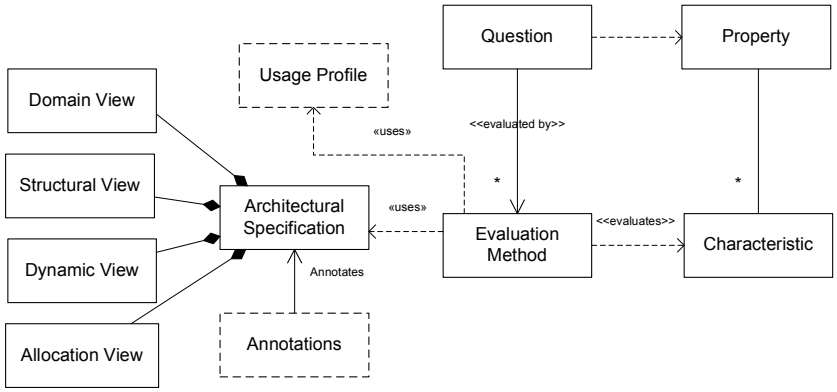
Komponentenbasierung als Methodische Voraussetzung

- „Divide and Conquer" als Erfolgsrezept zur Vorhersage von Eigenschaften komplexer Anwendungen
- Komponentenorientierte Architekturen
  - Komponenten als unabhängige, gekapselte Einheiten
  - Übersichtliche, auswertbare Systemstruktur (Architektur)
  - Einschränkung der Implementierungsfreiheit durch Wiederverwendung
    → bessere Vorhersagebarkeit
- Compositional/Modular Reasoning vs. Evaluation
  - Kompositionelles Schließen muss Abhängigkeit der Merkmale auf der Komponentenebene berücksichtigen
  - Evaluation als relaxiertes Verfahren zur Vorhersage

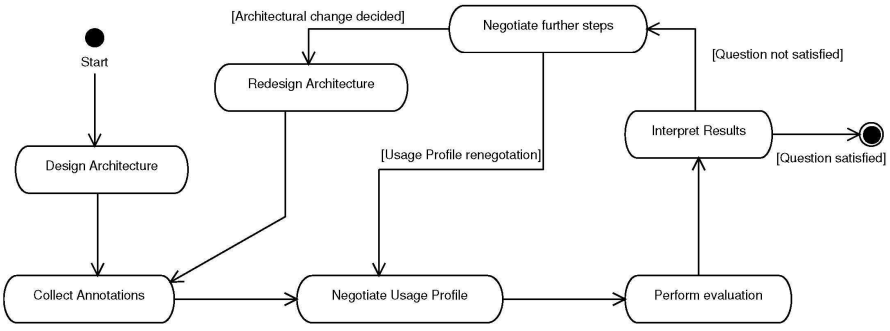
Sichten auf Software-Architekturen

- Beschreibungen der Software-Architektur liefern die Daten für die Vorhersage von Anwendungseigenschaften
  - Bauplan (Systemstruktur)
  - Eigenschaften der einzelnen Komponenten
- Architektur-Sichten
  - Struktursicht: Komponenten und Systemaufbau
  - Ressourcensicht: Zuordnung zu Rechnersystemen, Entwicklern
  - Dynamische Sicht: Systemverhalten (Ablauf)
  - Domänensicht: Anwendungsspezifische Informationen

Entitäten zur Evaluation



Vorgehensmodell für die Evaluation



Klassifikation existierender Methoden

Attribute	Method	Domain	Static	Dynamic	Ressource Mapping
Performance	(CB-)SPE		✓	✓	✓
	UML-PSI		✓	✓	✓
	LQN	✓	✓	✓	
	Cap. Planing	✓	✓	✓	
Reliability	Cheung		✓	✓	
	Wang et al.	✓	✓	✓	
	Ledoux	✓	✓	✓	
	Krishnamurthy et al.		✓	✓	
	Xie et al.			✓	

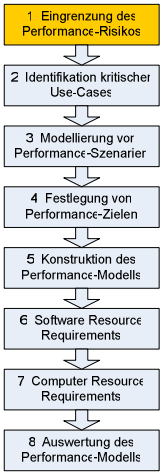
Klassifikation existierender Methoden

Attribute	Method	Domain	Static	Dynamic	Ressource Mapping
Interoperability	Zaremski		✓		
	Yellin		✓	✓	
Costs	ACSPM		✓		
	COCOTS		✓		
	COCOMO		✓		

Beispiele für die Evaluation: Übersicht

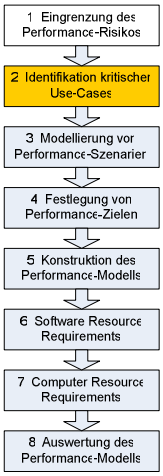
- Überprüfung der Interoperabilität von Komponenten
  - im Hinblick auf die Funktionalität
  - im Hinblick auf die Qualität
- Evaluation der Anwendungsqualität
  - Ermittlung der Performanz
  - Ermittlung der Zuverlässigkeit
- Evaluation der Anwendungsentwicklungskosten

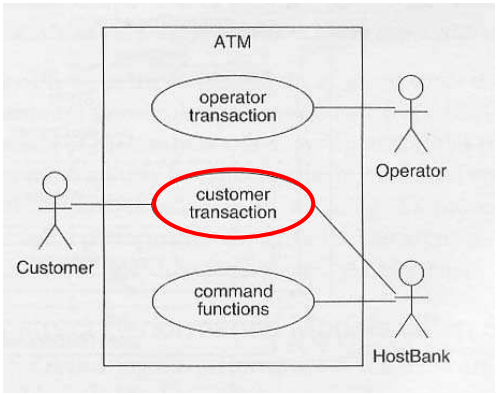
Beispiel: Performanz-Evaluation mit SPE



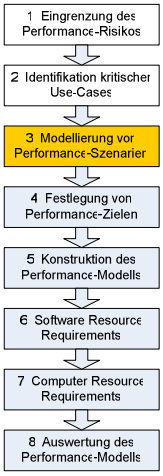
- Festlegung des Umfangs des SPE-Projekts je nach Signifikanz der Performance
- Erstellung eines detaillierteren Modells bei Performance-kritischeren Anwendungen
- Beispiel Geldautomat: geringes Performance-Risiko, einfaches Modell ausreichend

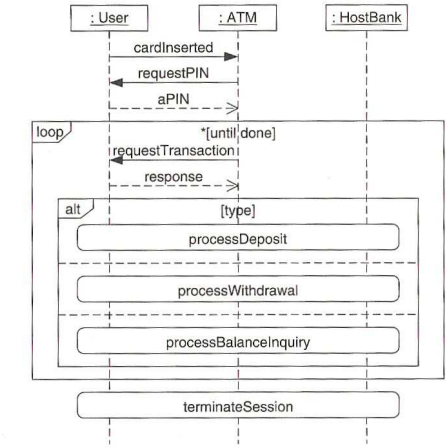
Beispiel: Performanz-Evaluation mit SPE



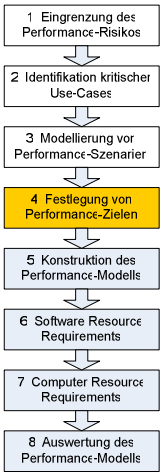


Beispiel: Performanz-Evaluation mit SPE



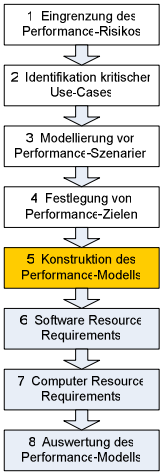


Beispiel: Performanz-Evaluation mit SPE

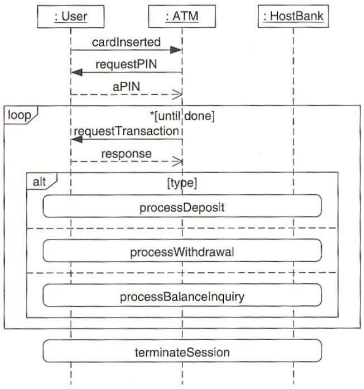


- Konkrete Vorgaben für z.B. Antwortszeiten, Durchsatz, Ressourcennutzung usw.
- Beispiel Geldautomat: „Die Bedienung darf nicht länger als 30 Sekunden dauern."

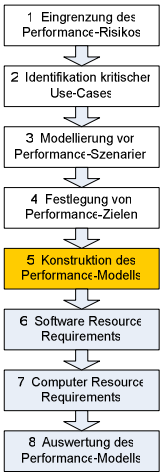
Beispiel: Performanz-Evaluation mit SPE



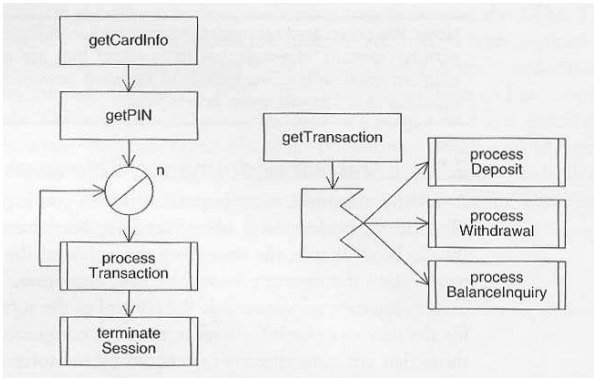
- Transformation der Sequenzdiagramme in Ausführungsgraphen



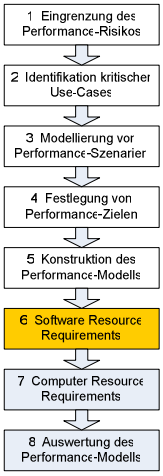
Beispiel: Performanz-Evaluation mit SPE



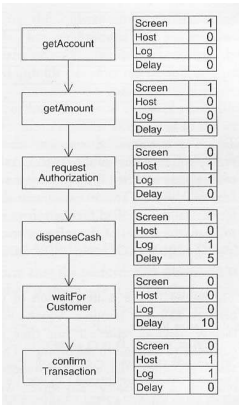
- Transformation der Sequenzdiagramme in Ausführungsgraphen



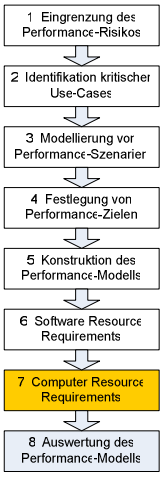
Beispiel: Performanz-Evaluation mit SPE



- Bestimmung der Anforderungen von Software-Elementen



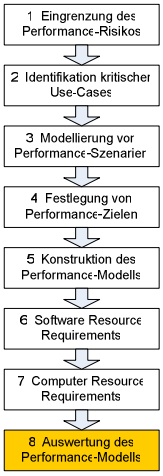
Beispiel: Performanz-Evaluation mit SPE



- Bestimmung der Anforderungen von Hardware-Elementen

Devices	CPU	Disk	Display	Delay	Net
Quantity	1	1	1	1	1
Service Units	Sec.	Phys. I/O	Screens	Units	Msgs.
Screen	0.001		1		
Host	0.005			3	2
Log	0.001	1			
Delay				1	
Service Time	1	0.02	1	1	0.05

Beispiel: Performanz-Evaluation mit SPE



- Berechnung der Performance-Werte
- Beispiel Geldautomat: „Die Bedienung dauert 29 Sekunden und ist somit knapp innerhalb der Vorgaben"
- bei Nichteinhaltung der Vorgaben:
  - Modifikation der Architektur
  - Modifikation der Anforderungen / Usage profiles
  - Überarbeitung der Performance-Ziele

Zusammenfassung

- Software-Architektur-Evaluation
  - Evaluation von Alternativen, keine „beweisbaren" Vorhersagen (z.B. Echtzeitschranken)
  - Überprüfung der Interoperabilität von Komponenten
  - Evaluation der Anwendungsentwicklungskosten
  - Keine Vorhersagen garantierter Eigenschaften
- Klassifikation von Evaluationsverfahren für Entwickler gemäß genutzter Architektursichten
  - welches Verfahren ist zu meinem Prozess kompatibel?
  - welche Änderungen muss ich an meinem Prozess vornehmen, um eine Verfahren einzusetzen?
  - welche Art von Entwurfsalternativen können vergleichend bewertet werden?

Herausforderungen

- Abhängigkeiten zwischen Einzelkomponenten erschweren Vorhersagen / Evaluation
- Behandlung *möglicher* Antagonistische Zusammenhänge zwischen Qualitätseigenschaften
  - Performanz / Zuverlässigkeit
  - Wartbarkeit / Performanz
  - Kosten / Wartbarkeit
- Modellierung intrinsischer Zusammenhänge zwischen Qualitätseigenschaften
  - Sicherheit / Performanz
  - Zuverlässigkeit / Performanz

Vielen Dank für Ihre Aufmerksamkeit!

Diskussion…

- Heute
- Später: reussner@informatik.uni-oldenburg.de