

Konzeption eines integrierten Spezifikations- und Entwicklungs-Prozesses für objekt-orientierte Systeme

Benjamin Braatz

Institut für Softwaretechnik und Theoretische Informatik
Technische Universität Berlin
Berlin, Germany

TFS-Forschungskolloquium, 28. April 2005

Übersicht

- 1 Spezifikation und Entwicklung
 - Entwicklungsprozess
 - Viewpoint-Spezifikationen

- 2 Formale Fundierung
 - Formale Syntax
 - Formale Semantik
 - Formale Entwicklungsrelationen

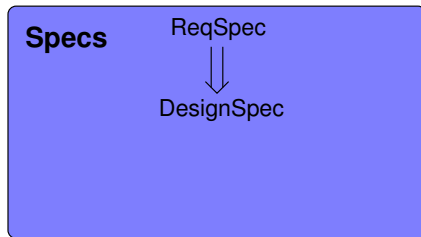
Schrittweise System-Entwicklung



Anforderungen

- funktional und nicht-funktional
- teilweise formalisiert
- grobe Beschreibung der Benutzer-Schnittstelle

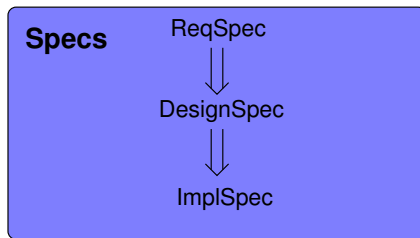
Schrittweise System-Entwicklung



Entwurf

- formale Beschreibung des Gesamt-Systems
- teilweise deskriptiv – teilweise konstruktiv
- plattform-unabhängig

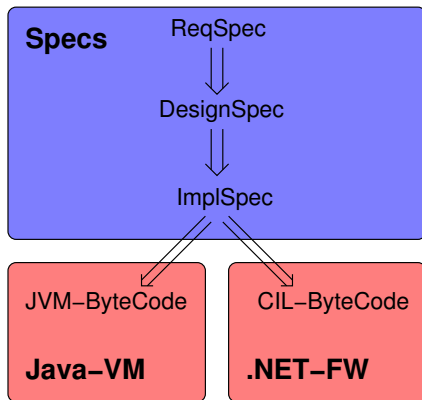
Schrittweise System-Entwicklung



Implementierung

- plattform-abhängig
- konstruktiv
- verschiedene Programmier- und Modellierungssprachen

Schrittweise System-Entwicklung



Code-Generierung
für unterschiedliche
Plattformen

Arten von Entwicklungsschritten

- Von den Anforderungen zum Entwurf:
Formalisierung und Präzisierung
- Vom Entwurf zur Implementierung:
Hinzufügen von plattform-abhängigen Strukturen
- Innerhalb des Entwurfs:
Erweitern von Determinismus/Konstruktivität
- Entwicklung des Entwurfs
⇒ Möglicherweise Entwicklung der Implementierung nötig

Modell-Transformationen

- Realisierung von Entwicklungsschritten
- Entwicklung durch Anwendung von Regeln
- Korrektheit der Regeln
⇒ Korrektheit der Entwicklung
- Mengen von Regeln für verschiedene Zwecke möglich

Viewpoint-Spezifikationen

Entwicklungs-Stufen mit verschiedenen, partiellen Techniken beschrieben

- Analyse

- Use-Case-Diagramme und textuelle Beschreibungen
- Einfache Klassendiagramme
- Use-Case-Szenarios in Sequenzdiagrammen
- Abstrakte OCL-Constraints

- Entwurf

- Komponenten-Architektur
- Komplexe Paket- und Klassenstrukturen
- Aktivitäts- und Zustandsdiagramme
- Constraints in OCL und anderen Logiken

- Implementierung

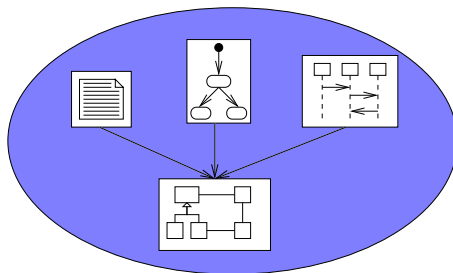
- Deployment-Diagramme
- Plattform-spezifische Klassenstrukturen
- Code-Fragmente in verschiedenen Programmiersprachen
- Konstruktive Modellierungstechniken

Viewpoint-Spezifikationen – Probleme

Trotz verschiedener Techniken:

- Sind die Anforderungen erfüllbar?
- Erfüllt der Entwurf die Anforderungen?
- Ist der Entwurf erfüllbar?
- Erfüllt die Implementierung den Entwurf?
- Gibt es Modell-Transformationen zwischen Anforderungen, Entwurf und Implementierung?
- Voraussetzung: Strukturbeziehungen gegeben

Struktur-Viewpoint



- Jede Spezifikation: Annahmen über Struktur
- Struktur in eigenem Viewpoint beschrieben
- Abbildung von allen Viewpoints in Struktur
- Syntaktische Konsistenz

Deskriptive und Konstruktive Techniken

- Zunächst nicht-deterministische Beschreibung von Eigenschaften mit deskriptiven Techniken
 - Szenarios (Sequenzdiagramme, Kollaborationen)
 - Logische Formeln (Gleichungen, First-Order)
- Dann schrittweise Konkretisierung durch konstruktive Techniken
 - Zustandsmaschinen
 - Konkrete Programmiersprachen
 - Deskriptive Techniken mit initialer Semantik
- Zusammenhang zu Komponenten: Deskriptive Techniken in Interfaces, Konstruktionen im Body

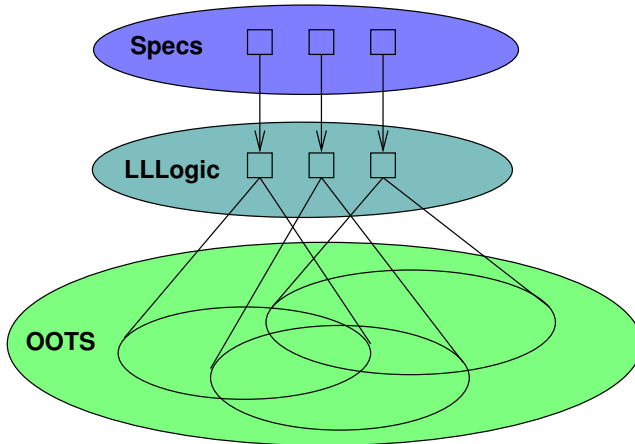
Motivation für Formalisierung

- Zusammenhang zwischen unterschiedlichen Techniken unvollständig
- Voll formal integrierte Techniken schwer anzuwenden
- Syntaktische Integration
⇒ Syntax-gesteuertes Editieren, Verfeinern
- Semantische Integration
⇒ Korrekte Entwicklungsregeln, Konsistenz
- Verträglichkeit und Beziehungen zu existierenden formalen Semantiken

Formale Syntax

- Sprache einer UML-Graphgrammatik (Dipl. FH, TM)
- Formale, abstrakte Syntax
- Weitere Transformationen für Entwurf, Implementierung, Refinement, Refactoring
- Momentan: Klassendiagramme, OCL/VOCL, Sequenzdiagramme, Zustandsmaschinen
- Mögliche zusätzliche Techniken: Weitere UML-Techniken, Verschiedene Programmiersprachen, Objekt-Graphgrammatiken
- Perspektive: Implementierung in Tiger oder ähnlichem Tool

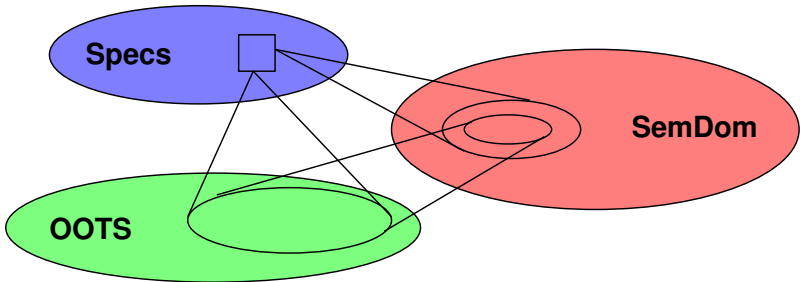
Referenz-Semantik



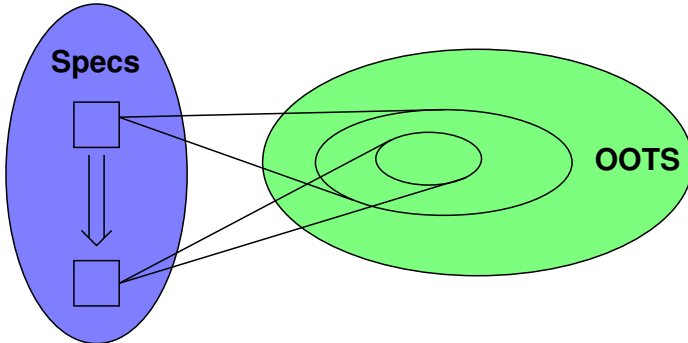
Objekt-Orientierte Transformationssysteme

- Diplomarbeit AK – Überarbeitung
- Kontrollgraph und Datenraum
- Datenraum streng objekt-orientiert – Alles ist ein Objekt!
- Zustände: Objektstrukturen mit Links zwischen Objekten
- Dynamik: Nachrichten zwischen Objekten mit Parameter-Objekten
- Weitere Struktur (Klassen, ...) durch spezielle Objekte
- Ein Datenraum für alle OOTSe

Beziehung zu üblicher Semantik

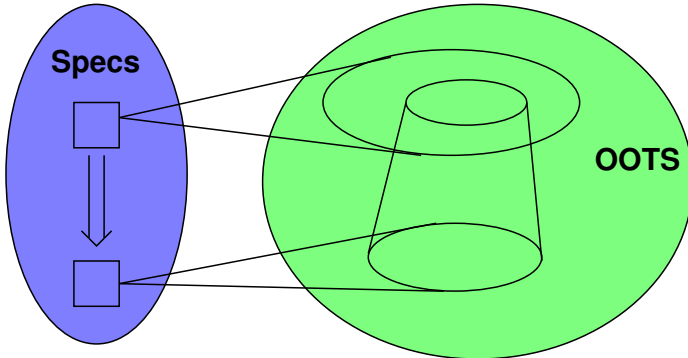


Entwicklungsrelationen und Modell-Transformationen



- Strukturbewahrende Transformation
- Zusätzliche Restriktion der Semantik
- Transformation erweitert spezifizierte Anforderungen

Entwicklungsrelationen und Modell-Transformationen



- Strukturverändernde Transformation
- Semantik mit anderer Struktur
- Übersetzung gemäß Strukturveränderung

Mögliche Entwicklungsrichtungen

Prozesstechniken		Algebraische Techniken	
Start mit			
Leeres Verhalten	Chaos	Term-Algebra	Finale Algebra
Verfeinerung durch			
Traces erweitern	Traces reduzieren	Gleichheit erweitern	Gleichheit reduzieren
Refusals reduzieren	Refusals erweitern	Ungleichheit reduzieren	Ungleichheit erweitern

Zusammenfassung und Arbeitsplan

- Integration von objekt-orientierten Viewpoint-Techniken
- Syntaktische Integration
 - ⇒ Graph-Grammatik für UML u. ä. (FH, TM)
- Semantische Integration
 - ⇒ Objekt-orientierten Transformationssysteme (AK)
- Entwicklungsrelationen
 - ⇒ Konsistenz-bewahrende Verfeinerung
- Validierung durch Fallstudie (TopicMaps, Entwicklungs-Umgebung)
- Langfristig: Tool-Unterstützung, Code-Generierung für JVM und .NET

Vielen Dank für Eure Aufmerksamkeit!