Fraunhofer

Einrichtung
Experimentelles
Software Engineering

Tamagotchi-Spezifikation in UML

Christian Becker
Steffen Glomb
Michael Graf

Gliederung

Grundlagen

- Notation
- Werkzeug

Modellierung

Details der Spezifikation

Erfahrungen

- Beurteilung von Notation und Werkzeug
- Review
- Typische Fehler

Fazit

Grundlagen - UML als Notation

Notation: UML (Unified Modeling Language), Version 1.1

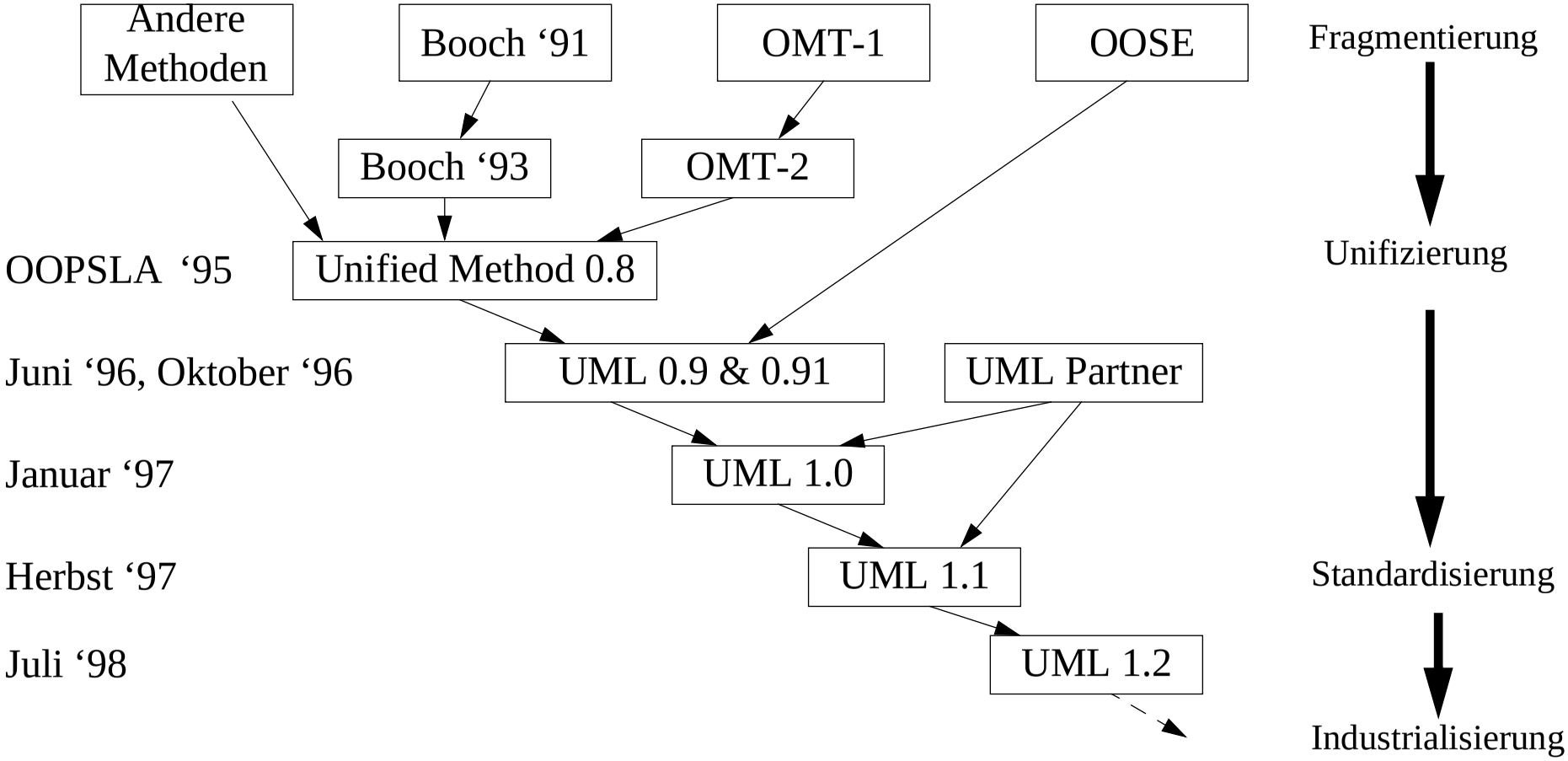
Autoren: James Rumbaugh, Grady Booch, Ivar Jacobson u.a.

Partner: Rational, IBM, Microsoft, HP, Oracle u.a.

Ziel:
- Vereinigung bestehender OO-Entwicklungstechniken
- Erleichterung des Entwurfs beliebiger, komplexer Systeme

Vorgehen:
- Nicht vorgeschrieben (UML ist nur Notation)
- Verwendung bekannter OO-Heuristiken

Grundlagen - Entwicklungsgeschichte der UML



Grundlagen - Konzepte der UML

Drei Dimensionen der Modellierung:

- Strukturelle Sicht
  - Klassendiagramm
  - Komponentendiagramm
- Funktionale Sicht
  - Anwendungsfalldiagramm
  - Aktivitätsdiagramm
- Verhaltenssicht
  - Zustandsdiagramm
  - Sequenzdiagramm
  - Kollaborationsdiagramm

Grundlagen - Werkzeug

Werkzeug: Rhapsody, Version 2.0.1

Hersteller: i-Logix

Basis: C++

Unterstützte UML - Diagramme:

- Anwendungsfalldiagramme („Use Case Diagrams“)
- Klassendiagramme („Object Model Diagrams“)
- Sequenzdiagramme („Sequence Diagrams“)
- Zustandsdiagramme („Statecharts“)

Grundlagen - Werkzeug (2)

Merkmale des Werkzeugs:

- Code-nahe Entwicklung
- Konsistenzcheck
- Codegenerierung
- Simulation mittels animierter Zustands- und Sequenzdiagramme

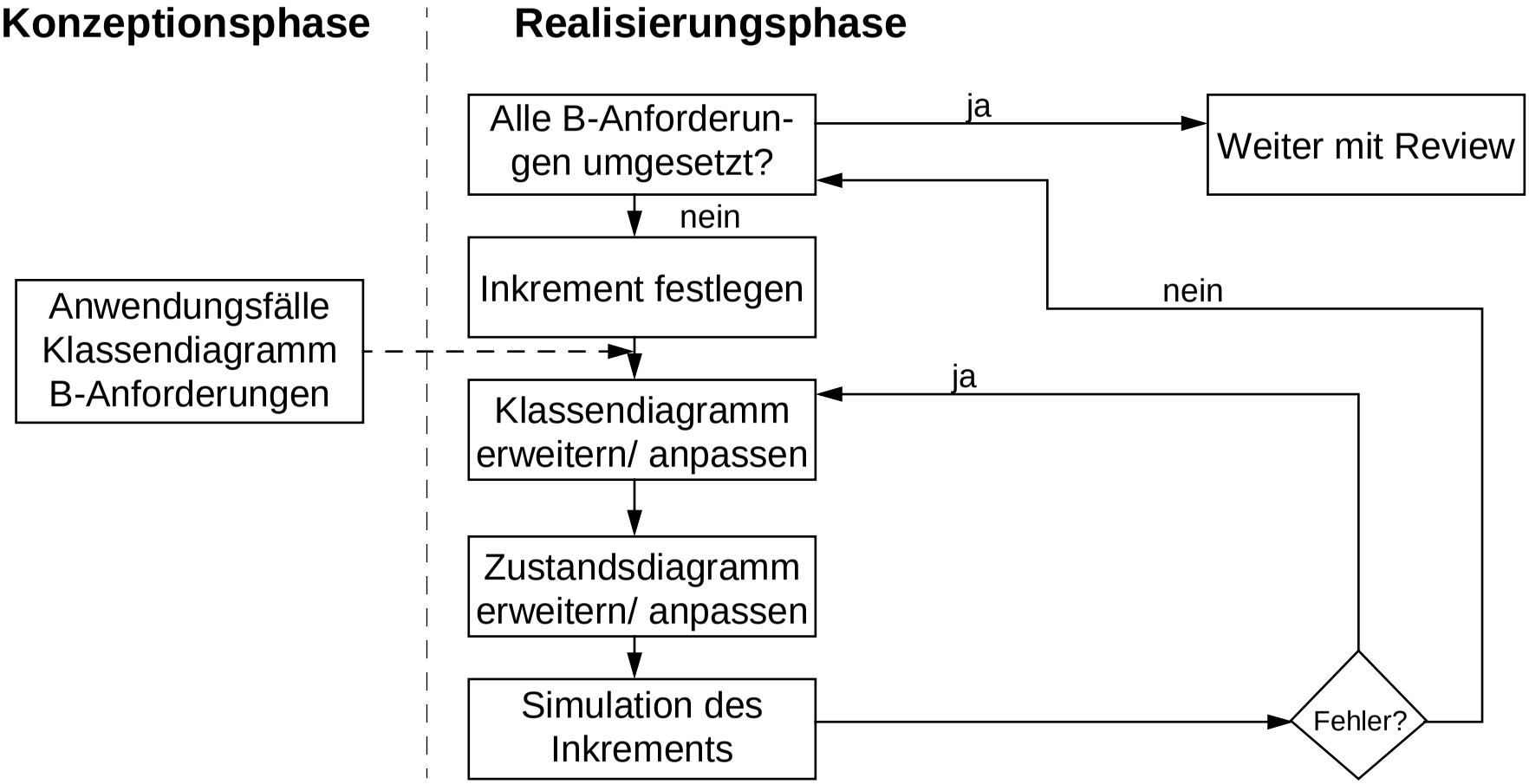
Modellierung - Vorgehensweise

Vorgehensweise: Unterteilung in Konzeptions- und Realisierungsphase

Konzeptionsphase (ohne Werkzeug):

- Brainstorming
- Anwendungsfälle identifiziert
- Klassendiagramm aufgestellt
- Sequenzdiagramme zur Klärung von Abläufen

Modellierung - Vorgehensweise (2)



Modellierung - Arbeitsteilung

Die Aufteilung der Anforderungen wird von Notation und Werkzeug unterstützt:

- Klassen als separate Verantwortungsbereiche
- Identifikation von Entwurfsinkrementen
- Versionsverwaltung mit Rhapsody möglich

Dennoch keine Aufteilung der Seminargruppe:

- Werkzeug nur auf einem Rechner installiert
- Nur ein Gruppenmitglied hat Schulung erhalten
- Fehlervermeidung durch direkte Rückkopplung in der Gruppe

Modellierung - Umfang der Spezifikation

Umsetzung der Spezifikation:

- Alle Anforderungen wurden umgesetzt
- Vereinfachte Benutzungsschnittstelle

Größe des Dokuments:

- 2 Klassendiagramme mit 20 Klassen
- 16 Zustandsdiagramme
- 5 Sequenzdiagramme
- 16.000 Zeilen Code (generiert)

Erfahrungen - Beurteilung der UML

Erfahrungen:

- Große Diagrammvielfalt
- Leicht erlernbare Notation
- Für schrittweise verfeinerten Entwurf geeignet
- Unterstützung bei der Vorgehensweise fehlt
- Schwierigkeit bei der Festlegung der Klassen

Erfahrungen - Beurteilung von Rhapsody

Stärken:

- Intuitive Bedienbarkeit
- Automatischer Konsistenzcheck
- Animierte Simulation
- Einfache Fehlerlokalisierung

Schwächen:

- Online-Hilfe und Dokumentation mangelhaft
- Keine „Undo“-Funktion
- Aktionen innerhalb von Zuständen nicht sichtbar

Erfahrungen - Beurteilung von Rhapsody (2)

Wünschenswerte Erweiterungen:

- Unterstützung weiterer UML-Diagrammtypen
- Simulator-Frontend
- Größere Compilerauswahl
- Verwendungsnachweis
- Verbessertes Drucken

Erfahrungen - Review

Vorgehensweise:

- Werkzeuggestütztes Review
- Bedienung des Werkzeugs durch jeweilige „Expertengruppe“
- Ohne Kenntnis der eigentlichen Notation
- Überdeckung der Spezifikation mit Benutzeranforderungen

Erfahrungen - Review (2)

Verständlichkeit:

- Rhapsody:
  - Animierte Simulation erleichtert Verfolgbarkeit des Systemverhaltens
  - Review auf Ausdruck kaum möglich
- SCR*:
  - Überblick aufgrund Zwang zu umständlicher Modellierung schwierig
  - Systemverhalten wegen Anzeigefehlern schlecht nachvollziehbar

Erfahrungen - Review (3)

Verfolgbarkeit:

- Rhapsody:
  - Einige Anforderungen auf mehrere Zustandsdiagramme abgebildet
  - Verwendungsnachweis wäre hilfreich
- SCR*:
  - Spezifikation schlecht überprüfbar, da Variablen manuell zu setzen
  - Variablenexplosion erschwert Verfolgbarkeit

Erfahrungen - Typische Fehler

Notation:

- Klasse „Tamagotchi“ anfangs zu groß entworfen
- Szenarien zu spät eingesetzt

Werkzeug:

- Multiplizität der Assoziationen vergessen
- Schwierigkeit bei der Reihenfolge der Konstruktoraufrufe
- Notwendige Includes vergessen

Erfahrungen - Aufwand

Einarbeitung:

- 40 Stunden Aufwand (zu gleichen Teilen für UML und Rhapsody)
- UML gut erlernbar, da auf bekanntem Konzept basierend
- Rhapsody schlecht dokumentiert, aber intuitiv bedienbar

Modellierung des Tamagotchi:

- 230 Stunden Gesamtaufwand (Realisierung, Simulation und Review)
- Mehraufwand wegen unzureichend dokumentiertem Werkzeug
- Größter zeitlicher Anteil: Realisierung

Fazit

Einschätzung von UML:

- Identifizierung von Objekten entspricht der menschlichen Denkweise
- Schrittweise Entwicklung wird unterstützt
- Nebenläufiges Verhalten läßt sich gut beschreiben

Einschätzung von Rhapsody:

- Übersichtliche Handhabung erleichtert Entwicklung
- Unterstützt Erstellung konsistenter Spezifikationen
- Code-nahes Arbeiten erfordert C++ -Kenntnisse
- Review auf ausgedruckten Diagrammen kaum möglich (fehlende Details)

Literaturhinweise

- „Objektorientierte Softwareentwicklung mit UML“, Bernd Oestereich, Verlag Oldenbourg
- „UML konzentriert“, Martin Fowler u. Kendall Scott, Verlag Addison-Wesley-Longman
- Referenzwebsite: www.rational.com
- „Adapting the Fusion Process to Support the UML“, Colin Atkinson, Object Magazine, Sigs Publications, Nov. ‘97