Modelle in der Sofware- und Systementwicklung

Modellbildung in der Entwicklung

Prof. Dr. Dr. h.c. Manfred Broy

Gemeinsam mit Dr. Bernhard Schätz

Fakultät für Informatik, TU München

SS 2007

Entwicklungsprobleme

F: Wie lang ist der Phaeton-Kabelbaum und was wiegt er?
A: 3.860m und 64 kg.

F: Wie stark stiegt die Anzahl der ECUs von der letzten zur aktuellen S-Klasse von Mercedes?
A: Um 64% von 45 ECUs zu bis zu 72 ECUs.

F: 1970 hatte die eingebettete Software eines Kfzs ca. 100 LOC. Wie groß ist der durchschnittliche Umfang heute?
A: 1 Million LOC; bei Premiumfahrzeugen bis zu 10 Millionen LOC.

F: Was passiert bei einem US-"New Beetle" von 2002 beim Ausfall eines der Bremslichter?
A: Die Automatik ist komplett blockiert.

F: Wenn alle Gewährleistungskosten eingebetteter Software eingespart werden können, wie würde das den EBIT (earnings before interest and taxes) beeinflussen?
A: Steigerung des EBIT um ca. 15-20%.

Ziele der Beherrschung softwareintens. Syst.

- Qualität
  - Aus Nutzungssicht: Funktionsumfang, Zuverlässigkeit, Sicherheit, Performanz
  - Aus Entwicklersicht: Wartbarkeit, Wiederverwendbarkeit
- Kosten
  - Entwicklungsaufwände
  - Vermarktbarkeit
  - Total Life Cycle Costs
- Entwicklungszeit
  - Time to Market
  - Reaktionszeiten bei Änderungen

Kernerfolgsfaktoren

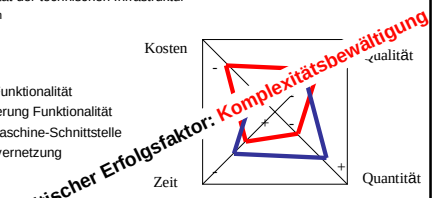
- Komplexitätsbeherrschung
  - Planbarkeit der Entwicklungsaufgabe
  - Arbeitsteiligkeit
  - Systematisierung
  - Einsatz bewährter Methoden
- Kostenreduktion und Bewältigung kurzer Entwicklungszeiten
  - Automatisierung
  - Wiederverwendung

Komplexität in softwareintensiven Systemen

Technische Komplexität
- Zunehmender Umfang Datenmodelle
- Zunehmende verteilte Implementierung
- Zunehmende Heterogenität Peripherie (Sensorik/Aktuatorik, MMI)
- Zunehmende Heterogenität der technischen Infrastruktur
  - Kommunikationsmedien
  - Betriebssysteme

Funktionale Komplexität
- Anwachsender Umfang Funktionalität
- Anwachsende Diversifizierung Funktionalität
- Anwachsende Mensch/Maschine-Schnittstelle
- Anwachsende Funktionsvernetzung

Entwicklungskomplexität
- Steigende Einflussnahme [illegible]ders
- Steigende Entwicklung in Zulieferketten
- Steigende Qualitätsanforderungen
- Steigender Innovationsdruck



Komplexität

Vielschichtigkeit, ineinander vieler Merkmale

Mittel der Komplexitätsbewältigung
- Abstraktion:
  Ausblenden unwesentlicher Teilaspekte
- Strukturierung:
  Gliederung und Aufteilung in in sich abgeschlossene Unterstrukturen
- Wissenschaftliche Durchdringung:
  Verwendung bekannter Erkenntnisse und Theorien
- Methodik und Systematik:
  Verwendung bewährter Verfahren und Lösungsmuster

Bewältigung der Komplexität

- **Modellbasierung: Modelle als Entwicklungsprodukte**
  - Vereinfachte Behandlung durch abstrahierte Darstellung
  - Angemessene Darstellung durch aufgabenorientierte Sichten
  - Unterstützte Entwicklung durch Analyse- und Syntheseverfahren
  - Integriertes Verständnis der Produkte durch Modelltheorie
- **Architekturzentrierung: Umfassender Architektureinsatz im Entwurf**
  - Systemstrukturierung durch Systemmodularisierung
  - Einsatz von Referenzarchitekturen
  - Abstraktionsebenen für Systemarchitekturen
  - Hierarchische Zerlegung von Systemen

Fehlerarten im Entwicklungsprozess

Quellphase	Fatal	Schwer	Mittel	Leicht	Total
Anforderungen	5,0	5,0	3,0	2,0	15,0
Design	3,0	22,0	10,0	5,0	40,0
Umsetzung	2,0	10,0	10,0	8,0	30,0
Dokumentation	0,0	1,0	2,0	2,0	5,0
Fehlereinf.	0,0	2,0	5,0	3,0	10,0
Gesamt	10,0	40,0	30,0	20,0	100,0

Quelle: Jones, 1991, Applied Software Measurement

- Wann werden welche Fehler gemacht:
  - Fatal: Anforderungsdefinition
  - Schwer: Entwurf
  - Mittel, leicht: Entwurf, Umsetzung

Qualitätssicherung im Entwicklungsprozess

Aktion	Analyse (% gef. Mängel)	Entwurf (% gef. Mängel)	Umsetz. (% gef. Mängel)
Strukt. Ansätze	50	25	10
Prototypen	40	35	35
Anford.Review	40	15	0
Entwurfsreview	15	55	0
Codereview	20	40	65
Teilsumme	75	85	73
Komp.Test	1	5	20
Funktionstest	10	15	30
Systemtest	10	15	35
Feldtest	20	20	25
Teilsumme	35	45	75
Gesamt	87	92	94

Quelle: Jones, 1991, Applied Software Measurement

- Effektivität:
  - Nur Testen: 1 von 4 Mängeln unentdeckt
  - Frühe Mängel benötigen frühe Maßnahmen
  - Hohe Qualität: Kombinierte Ansätze

Aufwände für Qualitätssicherung

Aktivität	Vorfeld (h/fp)	Ausführung (h/fp)	Behebung (h/f)
Strukt. Ansätze	0,15	0,25	1,00
Prototypen	0,25	1,00	1,00
Anford. Review	0,15	0,25	1,00
Entwurfsreview	0,15	0,50	1,50
Codereview	0,25	0,75	1,50
Komp. Test	0,50	0,25	2,50
Integr. Test	0,75	0,50	5,00
Systemtest	1,00	0,50	10,00
Feldtest	0,50	0,50	10,00

Quelle: Jones, 1991, Applied Software Measurement

- Aufwand:
  - Ausführung: wenig Unterschiede
  - Vorbereitung/Behebung: frühe Aktivitäten effizienter

Best Practices für Systementwurf

Best Design Practices for System Software:

1. Formal Specifications
2. Architecture Specifications
3. Specification Segmentation
4. Prototypes
5. Design Inspection
6. Performance Analysis
7. Reusable Specification

Source: Jones, 2000, Software Assessments, Benchmarks and Best Practices

Strukturierte, formalisierte Entwurfsmodelle

Softwareentwicklung braucht Modelleinsatz

- Software ist unanschaulich
  - ⇒ Modelle erlauben die begreifbare Behandlung von Software
- Software ist unübersichtlich
  - ⇒ Modelle erlauben Vereinfachung unwesentlicher Details
- Software ist facettenreich
  - ⇒ Modelle erlauben gezielte Darstellungen von Eigenschaften
- Software ist umfangreich
  - ⇒ Modelle erlauben automatisierte Analyse/Syntheseverfahren

```
int mult(a,b:int) {
   int c=0;
   while(a>0) do {
     c = c + b;
     a = a - 1; }
   return c;
}
```

Bessere Modelle = Höhere Effektivität/Effizienz

Prinzipielles zum Modelleinsatz

- Ein Modell ist eine vereinfachte Abbildung eines Produkts für einen bestimmten Zweck (Sicht) durch ein Darstellungsmittel (Notation)

Erfahrungen:

- Ein Modell reicht in der Regel nicht aus!
- Modellbasierung braucht anwenderspezifische Abstraktionen!
- Nicht die Notation, die Anwendung des Modells ist entscheidend!
- Anforderungsmodelle sind so wichtig wie Entwurfsmodelle!
- Die Validierung von Modellen ist so wichtig wie die Verifikation!
- Keine Modellbasierung ohne Werkzeugunterstützung!
- Ohne Modellintegration keine Durchgängigkeit!
- Modelle und ihr Einsatz müssen in den Prozess integriert werden!

Megatrend „Modellbasierung“, aber ...

- ... viele pragmatische Ansätze zur Modellbasierung haben unklare Interpretationen
  - ⇒ z.B. ungeeignet für die Kommunikation zwischen Unternehmen
- ... oft decken die Produkte zusammen nur Teilaspekte ab
  - ⇒ z.B. Systemspezifikationen ungeeignet für Testbereich
- ... werden die frühen Phasen nicht durch Modelle unterstützt
  - ⇒ z.B. fehlende Erfassung von Anwendungsszenarien
- ... existieren keine Überprüfungsverfahren für die Teilprodukte
  - ⇒ z.B. keine Sicherung zwischen Schnittstellenkonsistenz
- ... fehlen Verfahren zur Transformation/Generierung von Produkten
  - ⇒ z.B. keine Testfallgenerierung aus Verhaltensmodell

Über die Natur der Softwareentwicklung

Zwei extreme Sichten:

- Softwareentwicklung bedeutet die Konstruktion eines formalen, mathematischen, logischen Modells, das effizient ausführbar ist - ist also eine ziel- und zweckgerichtete Formalisierung
  - Software ist ein mathematisches Objekt, quantifizier-, spezifizier- und verifizierbar
- Softwareentwicklung ist Kunst und Handwerk; sie basiert auf Erfahrung, Augenmaß und Können
  - Software definiert sich durch die Ausführung auf einer Hardware/Softwareplattform, ist von Natur aus unzuverlässig, ad hoc, wird in Versuch und Irrtum erstellt

Inhalte

Vorlesungsinhalte:

1. Modellbegriff
2. Grundlagen
   1. Datenmodellierung
   2. Zustandsmaschinen
   3. Abläufe und Prozesse
   4. Architekturen
   5. Schnittstellen
   6. Modellintegration
3. Anwendung
   1. Taxonomien und Ontologien
   2. Modellierung in der Analyse
   3. Architekturentwicklung
   4. Qualitätssicherung
   5. Qualitätsmodelle
4. Modellgetriebene Entwicklung
5. Domänenspezifische Modellierung

Was ist ein Modell?

Ein Modell ist
- eine Abstraktion
- bezogen auf einen Betrachtungsgegenstand
- unter einem Gesichtspunkt für einem bestimmten Zweck.

Ein Modell erfordert eine Präsentationsform durch
- eine Metapher (Gedankenmodell) durch Text
- eine Sprache (Programmiersprache, Modellierungssprache) oder
- einen Gegenstand.

Ein Modell erlaubt bestimmte Aussagen über den Betrachtungsgegenstand.

Ein Modell dient
- der Kommunikation zwischen Menschen
- aber auch der Automatisierung

Daraus ergibt sich die Qualität eines Modells

Modellbildung und Modell

Modelle sind Bestandteile eines Prozesses, in welchem sie von einem Subjekt zu einem Original für einen bestimmten Zweck konstruiert und eingesetzt werden. Dieser Prozess wird mit Modellbildung bezeichnet.

Stachowiak, H.: Allgemeine Modelltheorie.

Modell: materielles oder ideelles Objekt, das stellvertretend für ein ihm ähnliches Original zum Gegenstand von Untersuchungen gemacht wird, die direkt am Original nicht möglich oder nicht zweckmäßig sind, deren Ergebnisse aber Rückschlüsse auf analoge Eigenschaften des Originals ermöglichen.

Digitales Wörterbuch der Deutschen Sprache.

Model: Muster, Entwurf
a) maßstabgetreue Verkleinerung eines geplanten oder bereits vorhandenen Bauwerks, eines (technischen) Gegenstands
b) (technischer) Gegenstand, der als Vorlage, Muster zum Anfertigen von weiteren gleichartigen Gegenständen dient.

Digitales Wörterbuch der Deutschen Sprache.

Modelle in der Informatik

Definition in der Informatik: Modell
- Kontext: Entwicklung von (Software-)Systemen
- Zweck: Beschreibung eines existierenden oder gedachten (Software-)Systems
  - Elimination aller unwesentlicher Eigenschaften
  - Erhaltung aller wesentlichen Eigenschaften
- Einsatzvariante: Deskriptives Modell
  - Modell als Abbild des Systems
  - Überprüfung von Systemeigenschaften
- Einsatzvariante: Präskriptives Modell
  - Modell as Vorbild des Systems
  - Vorgabe von Systemeigenschaften

Zwei essentielle Aspekte der Software Entwicklung

Software Entwicklung bedeutet immer:
+ Formalisierung: Software ist formales Artefakt
+ Modellierung: Software bedeutet immer Abstraktion von der Wirklichkeit (von dem Betrachtungsgegenstand) und damit Modellbildung

Jedes Softwaresystem enthält zwangsläufig Modelle und zwar
- ein Domänenmodell (fachliche Gesichtspunkte)
- ein Modellierung technischer Gesichtspunkt

Beispiele:
- Die „Blutgruppen der Komponenten einer Softwarearchitektur“ a la Siedersleben
- Das R3 System von SAP

Interessante Frage: Welche der Modelle sind wertvoller?

Modelle im Code

Im Code von Softwaresystemen (= Programme + Daten) sind

- das Domänenmodell (fachliche Gesichtspunkte)
- die Modellierung technischer Gesichtspunkt

„vergraben“ (sind implizit).

- Beim Entwickeln und Warten von Software müssen diese Modelle ständige genutzt (und gegebenenfalls wiederentdeckt werden)

Ein Beispiel: Das Problem der absoluten Mehrheit

totale Funktion f: IN → Partei
Anzahl n ∈ IN, mit n > 0 von Parteigängern.
#(p, k) = | {i ∈ IN: 1 ≤ i ≤ k ∧ f(i) = a} |
major(p, k) = 2 * #(p, k) > k
anarchic(k) = ∀ p ∈ Partei: 2 * #(p, k) ≤ k
k, e : Var Nat, r : Var Partei
k, r, e := 2, f(1), 1;
do k ≤ n **then**
if f(k) = r **then** k, e := k+1, e+1
[] f(k) ≠ r ∧ e = 0 **then** k, r, e := k+1, f(k), e+1
[] f(k) ≠ r ∧ e > 0 **then** k, e := k+1, e–1
fi od
{¬anarchic(n) ⇒ major(r, n)}

A, H : Var Set Nat
Wir wählen als Invariante
A anarchisch
H homogen: ∀ m ∈ H: f(m) = r
e = |H|

H, A := {1}, Ø

H := H ∪ {k}

H := {k}

H, A := H\{any(H)}, A∪{k, any(H)}

A anarchisch ∧ H homogen ∧ A∪H = {1, ... n}

Und noch ein Beispiel

SPEC SEQ =
{
based_on BOOL,
sort Seq α,
‹› : Seq α,
‹_› : α → Seq α,
Mixfix
_°_ : Seq α, Seq α → Seq α,
Infix
ises : Seq α → Bool,
first, last : Seq α → α,

Seq α **generated_by** ‹›, ‹_›, _°_,

ises(‹›) = true,
ises(‹a›) = false,
ises(x°y) = and(ises(x), ises(y)) ,
x°‹› = x = ‹›°x,
(x°y)°z = x°(y°z) ,

first(‹a›°x) = a,
last(x°‹a›) = a,
head(x°‹a›) = x,
rest(‹a›°x) = x
}

Hier haben wir alles perfekt: Klare Modelvorstellung, Syntax, logische Theorie

Formale Modelle

Ein formales Modell umfaßt:

- Syntax
  - Formale Sprache
  - Graphik
- Semantik - Festlegung der Bedeutung
  - Informell
  - Formal
    - Denotationell
    - Axiomatisch
    - Operationell
- Theorie
  - Theoreme
  - Umformungsregeln

“Formal Methoden” und “Software Engineering”

- Formal Methoden werden häufig als inadäquat, zu teuer, unrealistisch, nicht auf grosse Systeme unwendbar beurteilt
- Praktische Software Entwicklung wird often als “ad hoc”, “immature”, uncontrollable, und “not an engineering discipline” bewertet

Digitale Modellierung

Die Modelle der Informatik sind digital!

Essentielle Beiträge der Modellorientierung

- Software ist unanschaulich („immateriell“), bestimmt komplexe dynamische Vorgänge
  - Software wird durch Modelle begreifbar
- Die Ausführungsmodelle unserer Maschinen sind undurchsichtig
  - Software wird von den komplexen („low level“) Ausführungsmodellen unabhängig
- Software ist umfangreich und unübersichtlich
  - Modelle erlauben gezielte Darstellungen von Eigenschaften
  - Modelle geben Übersicht
- Software braucht unterschiedliche Darstellung verschiedener Aspekte
  - Modelle können auf Aspekte zugeschnitten werden - Sichtenmodellierung
- Software braucht präzise Analysen
  - Modelle können auf geeignete Theorien abgestützt werden
  - Modelle erlauben tiefe Automatisierung/Werkzeugunterstützung

Modellierung in softwareintensiver Systems

Was ist ein Modell:

- Eine vereinfachte („abstrakte“) Wiedergabe eines Produkts („Systems“) in einer bewährten Form (mit wohlverstanden Mitteln) für einen bestimmten Zweck

Was ist eine Beschreibung eines Modells:

- Text, Graphik, Tabellen die bestimmte Eigenschaften eines Produkts („Systems“) beschreiben

Wichtige Ziele:

- Verständlichkeit
- Präzision

Was ist eine Anforderung

- Geforderte Eigenschaft für ein System

Was ist eine (Anforderungs-)Spezifikation:

- Die genaue und vollständige Festlegung aller Eigenschaften eines Produkts („Systems“)

Vom Nutzen der Modelle

- Ein Modell erlaubt ein konkrete Darstellung bestimmter Aspekte oder einer Perspektive eines Systems
  - Erfassung und Erkenntnis
  - Dokumentation
  - Formalisierung
  - Analyse
  - Weiterverwendung
    - Modelltransformation
    - Codeerzeugung
    - Produktlinien
- Modelle
  - strukturieren die Sichten auf ein System
  - erfassen Anwender- und Entwicklerwissen
  - geben eine Richtschnur für den Entwicklungsprozess

Modellanwendung

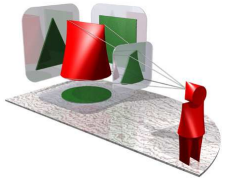
Präskriptiv:

- Zweck: Konstruieren von Systemen
- Fokus: Verständlichkeit
- Anwendung: Dokumentation, Kommunikation
- Bedeutung: (i.a.) informelle Semantik
- Beispiele:
  - Use Cases
  - Obektdiagramme
  - Sequenzdiagramme

Deskriptiv

- Zweck: Analysieren von Systemen
- Fokus: Analysierbarkeit
- Anwendung: Validierung, Verifikation
- Bedeutung: (i.a.) logische Formalisierung
- Beispiele:
  - Datenalgebren
  - Prozessalgebren
  - Kripkestrukturen

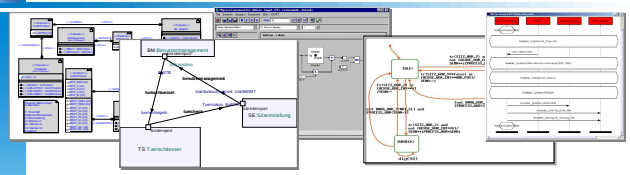
Modelleinsatz in der Entwicklung



Modelleinsatz: Komplexitätsreduktion, Automatisiertung und Qualitätssteigerung

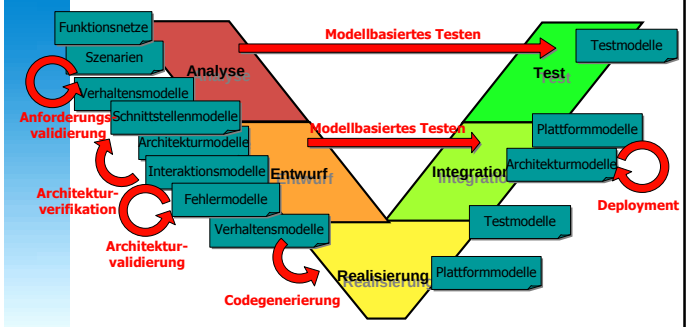
- Modularisierung durch Sichtenkonstruktion
  - Horizontal: Verschiedene Aspekte (z.B. Struktur, Verhalten)
  - Vertikal: Verschiedene Phasen (z.B. Analyse, Entwurf)
- Komposition durch Sichtenintegration
  - Analytisch: Prüfung Modelle (z.B. Vollständigkeit, Kompatibilität)
  - Synthetisch: Erzeugung Modelle (z.B. Code,Testfall,)

Modelle und Sichten



- Datensicht
  - Abstrakte Datentypen
  - E/R Diagramme
  - Klassendiagramme
- Strukturisicht
  - Blockdiagramme
  - Klassendiagramme
  - Paketdiagramme
- Verhaltenssicht
  - Prozessdiagramme
  - Zustandsdiagramme
- Ablaufsicht
  - Sequenzdiagramme
  - Kollaborationsdiagramme
  - Prozessdiagramme

Modelle im Entwicklungprozess



Modellbasierung ist mehr als Dokumentation

Modellbasierung unterstützt (im Idealfall):

- Unmissverständliche Beschreibung
- Wohldefinierte Sichten und Ebenen der Abstraktion
- Maschinenverarbeitbare Beschreibungen
- Rechnergestützte Analyse
  - Konsistenzsicherung
- Automatische Extraktion von Information für Nachfolgeschritte
- Frühe Qualitätssicherung
  - Verifikation

Weitere Modelle im Software & Systems Eng.

- Vorgehensmodelle
- Qualitätsmodelle
- Kostenmodelle
- Meta-Modelle
- Nutzermodelle

Zusammenfassung

- Modellorientierung
  - bildet die Grundlage ingenieurmäßiger Systementwicklung
  - mittels Kombination Komplexitätsredukution/Qualitätssteigerung
  - auf Basis von Modellen
    - mit abstrahierenden, aufgabenorientierten Sichten
    - integriert in eine einheitliche Modelltheorie
    - unterstützt durch analytische und generative Verfahren
- Modellorientierung ist nicht Stand der Technik in der Anwendung
  - Isolierte Nutzung von Modellen
  - Mangelnder Bezug zu Modelltheorien
  - Fehlende Integration von Modellsichten
  - Fehlende tiefe Werkzeugunterstützung