Modellbildung in der Entwicklung mit Schwerpunkt Architekturen Schnittstellen

Modellbildung in der Entwicklung
Prof. Dr. Dr. h.c. Manfred Broy
Gemeinsam mit Dr. Bernhard Schätz
Fakultät für Informatik, TU München
SS 2007

Überblick

- Motivation
- Syntaktische Schnittstellen
- Dynamische Schnittstellen

Schnittstelle

- Die Schnittstelle, auch Interface (englisch „Grenzfläche“, Begriff ursprünglich aus der Naturwissenschaft).
- Eine Schnittstelle bestimmt die Grenzen eines Systems zu seiner Umgebung (oder die Grenze zwischen zwei Systemen) und beschreibt die Wechselwirkung über die Systemgrenzen hinweg.
- Der Austausch von Informationen erfolgt in Form von physikalischen (z. B. Elektrische Spannung, Stromstärke) oder logischen Größen (Daten) und kann analog (z.B. Mikrofon an einer Soundkarte) oder digital (z.B. Parallelschnittstelle des PC) erfolgen.

Schnittstelle

- Die Schnittstellenspezifikation beschreibt den nach außen für seine Nutzung relevanten Teil eines Systems - wir sprechen von der Schnittstellensicht (auch Black Box Sicht oder beobachtbares Verhalten)
- Die Schnittstellensicht beschreibt die Wirkung eines (Teil-)Systems als Teil eines umfassenden System
- Das System wird zur Komponente des umfassenden Systems
- Ersetzt man ein Teilsystem eines Systems durch ein anderes, so sollte die Übereinstimmung der Schnittstellenbeschreibung der Teilsysteme ausreichen um sicherzustellen, dass das Systeme damit unverändert arbeitet (Schnittstellenkompatibilität)
- Man unterscheidet zwischen
  - Syntaktischer Schnittstelle (Struktur) und
  - Schnittstellenverhalten

Einsatzzweck von Schnittstellen (1)

- Arbeitsteilige Entwicklung: Wenn ein System von mehreren Teams gleichzeitig entwickelt werden soll,
  - so wird das System in Teilsysteme zerlegt,
  - die müssen in ihren Schnittstellen zusammen passen,
  - damit sie das gewünschte Systemverhalten sicherstellen und
  - die einzelnen Teile später wieder integriert werden können.
- Wiederverwendung von Systemen als Komponenten
  - in Form einer Komponentenbibliothek
  - Integration von zugelieferten Komponenten

Einsatzzweck von Schnittstellen (2)

- Als Teil der Komponenten-Spezifikation
  - Beschreibt eine bestimmte Sicht auf eine Komponente
- Reduzierung Komplexität
  - Aufteilung eines komplexen Systems in mehrere einfachere Subsysteme
  - „Devide and Conquer“
- Strukturierung des Systems
  - Welche Teile eines Systems sollen welche Aufgaben übernehmen

Einsatzzweck von Schnittstellen (3)

- Prüfung der Kompatibilität
  - Lassen sich mehrere unabhängig entwickelte Komponenten zu einem funktionierenden Gesamtsystem integrieren?

Schnittstelle, Komposition, Architektur

Es besteht eine enge Beziehung zwischen den Begriffen

Schnittstelle, Komposition, Architektur

- Eine Architektur entsteht durch Komposition einer Menge von Teilsystemen (die Komponenten der Architektur)
- Die Schnittstelle beschreibt (möglichst genau) die Information, die nötig ist um das (Schnittstellen-)Verhalten des Systems, das durch die Architektur geformt wird, vorherzusagen
- Das Architekturverhalten (Schnittstellenverhalten des Systems, das durch die Architektur geformt wird) ergibt sich formal durch die Komposition des Schnittstellenverhaltens der Komponenten

Schnittstelle, Komposition, Architektur

Daraus ergibt sich ein Konzept für das Zusammenspiel der drei Begriffe

1. Es wird ein Modell für Schnittstellen (Syntax und Verhalten benötigt); sei IF die Menge der Schnitstellen
2. Es wird eine Komposition benötigt, die aus Schnittstellen Schnittstellen durch Komposition bildet
   $\otimes : IF \times IF \rightarrow IF$
3. Ein Term bezeichnet für $c_1 \ldots c_n \in IF$
   $c_1 \otimes \ldots \otimes c_n$
   das Schnittstellenverhalten der Architektur aus den Teilsystemen mit Schnittstellenverhalten $c_1 \ldots c_n$

Prinzipien des Software Engineerings:

Information Hiding, Abstraktion, Modularität

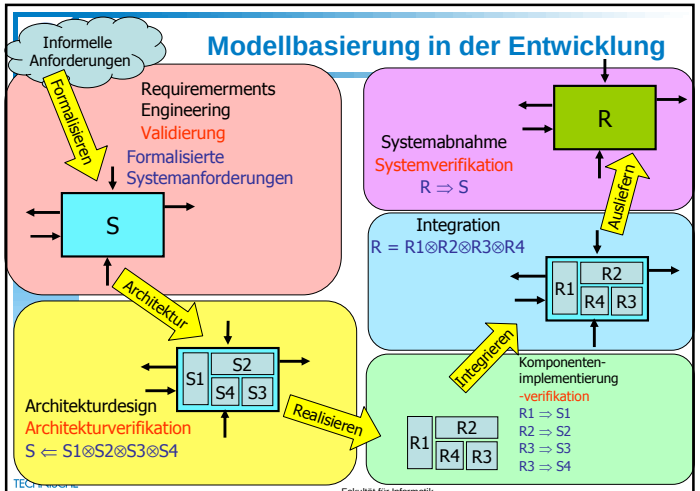
Prinzipien des Software Engineerings:

- *Information Hiding*: Implementierungsdetails der Teilsysteme verbergen
- *Abstraktion*: Schnittstellenabstraktion: Nur solche Information in der Schnittstellenspezifikation angeben, die für das Verhalten des Systems, das durch die Architektur gebildet wird, von Bedeutung sind
- *Modularität*: Das Schnittstellenverhalten der Architektur ergibt sich durch Komposition aus dem Schnittstellenverhalten der Teilsysteme
- *Hierarchie*: Die Komposition von Teilsystemen, spezifiziert durch ihre Schnittstellen, ergibt ein zusammengesetztes System,
  - dessen Schnittstellenverhalten sich aus dem Schnittstellenverhalten der Teilsysteme ermitteln lässt und
  - das selbst wieder als Komponente in der Komposition von Systemen verwendet werden kann.

Beispiele für Konzepte mit Schnittstellen

- Funktionale Programme
- Anweisungen
- Prozeduren
- Module
- Klassen
- Interaktive Systeme

Modellbasierung in der Entwicklung



Konstruktion eines Schnittstellenkonzepts

- Annahme: Gegeben eine Programmiersprache oder Modellierungssprache mit syntaktischen Kompositionsoperator, mit syntaktischen Schnittstellenbegriff und operationeller Semantik
- Gesucht: Kompositionale Semantik und Schnittstelle
- Beobachtung:
  - Ist eine beobachtbare Semantik (Verhalten) für Systeme definiert, so existiert ein „kanonischer" Begriff einer Schnittstelle
  - Es ex. dann ein minimales (voll abstraktes) Modell für die Schnittstellenbeschreibung
  - Eine explizite Darstellung für das Schnittstellenmodell zu finden ist oft schwierig

Bemerkung: Es wäre wohl richtiger statt über Schnittstelle über „Zusammensetzstelle" zu reden

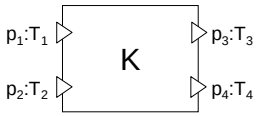
Aufgabe: Ein Schnitstellenkonzept für Petri-Netze

Ein Schnitstellenkonzep füt OO-Klassen

Syntaktische Schnittstelle

Definition:

Typisierte Ein-Ausgabe-Ports einer Komponente.



Syntaktische Schnittstelle

- Definition der Typen ähnlich wie bei funktionalen/operationalen Sprachen
- Literatur über Typsysteme:

  Luca Cardelli. **Type Systems**. In *The Computer Science and Engineering Handbook.* S. 2208-2236, CRC Press, 1997

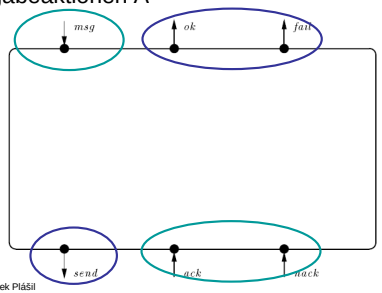
Schnittstellenverhalten

- Im Folgenden werden die „Interface Automata" von de Alfaro und Henzinger als Möglichkeit vorgestellt, das Verhalten von Schnittstellen zu beschreiben.
- Interface Automata beschreiben sowohl Verhalten der Komponente, wie auch erwartetes Verhalten der Umgebung.
- Literaturangabe:

  Luca de Alfaro and Thomas A. Henzinger. **Interface Automata**. In *Proceedings of the Ninth Annual Symposium on Foundations of Software Engineering* (FSE), ACM Press, 2001, pp. 109-120.

Interface Automata

- Die Schnittstelle einer Komponente besteht aus einer Menge von Eingabe Aktionen A^I und Ausgabeaktionen A^O

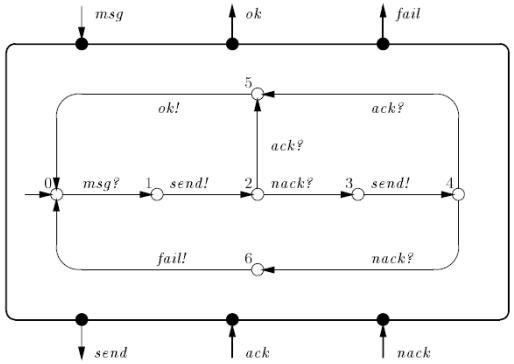


Basiert auf:
Jiří Adámek, František Plášil
Behavior specification and model checking of SW components

Interface Automaton

- A^I – Menge der Eingabeaktionen
- A^O – Menge der Ausgabeaktionen
- A^H – Menge der versteckten Aktionen
- $A = A^I \cup A^O \cup A^H$ – Menge aller Aktionen
- V – Menge der Zustände
- $V^{init} \subseteq V$ – Startzustand
- $\mathcal{T} \in Q \times A \times Q$ – Menge von Transitionen

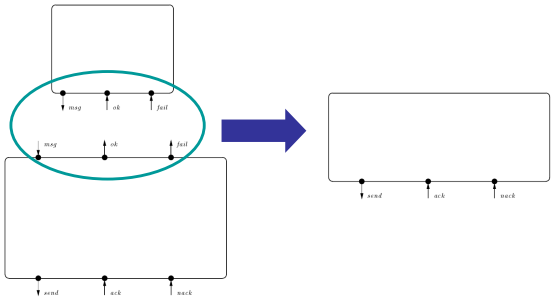
Beispiel



Komposition

- Komponenten werden Komponiert um aus einfachen Komponenten komplexere zu konstruieren.
- Dies lässt sich analog auch Schnittstellen übertragen.
- Die Aktionen, die vorher zwischen zwei Schnittstellen abgelaufen sind, sind jetzt versteckt, also von außen nicht mehr sichtbar.

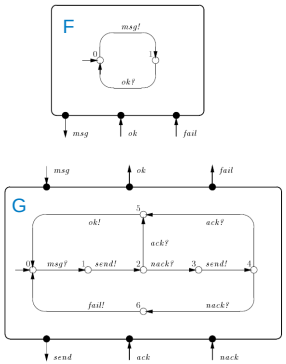
Komposition



Kompatibilität

- Fehler
  - Wenn ein Interface Automat eine Ausgabe-Aktion ausführt, die der andere in seinem gegenwärtigen Zustand nicht verarbeiten kann, wird dies als Fehler bezeichnet.
- Kompatibilität von Interface Automaten
  - Zwei Interface Automaten F, G sind kompatibel (geschrieben als F ~ G), wenn es eine Umgebung gibt, in der F und G fehlerfrei zusammenarbeiten können.
  - Eine solche Umgebung wird „helfende Umgebung“ genannt.
  - Im Umkehrschluss erhält man damit eine Aussage, in welche Umgebungen F und G gemeinsam eingesetzt werden können

Beispiel



- Automat F akzeptiert in Zustand 1 nur „ok“, aber Automat G kann aber in Zustand 6 „fail“ liefern.
- Daher können F und G nur gemeinsam in einer Umgebung eingesetzt werden, in der diese Situation nicht auftreten kann.

Komposition

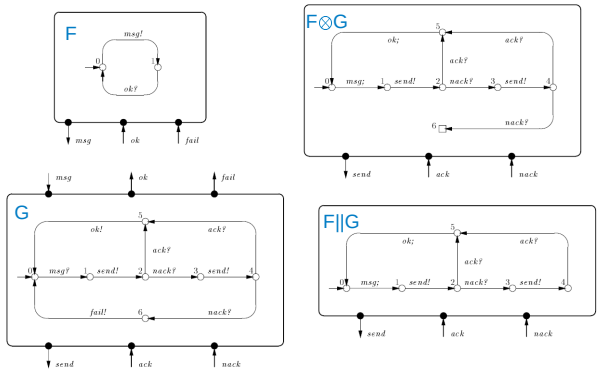
- Die Komposition von zwei Schnittstellenautomaten F und G wird geschrieben als F || G
- F || G beschreibt also das Verhalten von F komponiert mit G, sowie auch das einer helfenden Umgebung.
- F || G existiert, falls F und G kompatibel sind, also F ~ G

Prüfung von F ~ G und Konstruktion von F || G

1. Konstruktion des Produktautomaten F ⊗ G der beschreibt, wie F und G in beliebigen Umgebungen zusammenarbeiten würden.
2. Identifikation von Fehlerzuständen
3. Löschen aller Zustände aus F ⊗ G, aus denen ein Fehlerzustand autonom (also nur mit Ausgabe- oder versteckten Aktionen) erreichbar ist. Da keine Umgebung das erreichen dieser Zustände verhindern kann.

- Wenn dabei der Startzustand entfernt wurde sind F und G nicht kompatibel.
- Ansonsten sind F und G kompatibel und die verbleibenden erreichbaren Zustände definieren F || G.

Beispiel



Verfeinerung

- Ein Interface Automat G‘ verfeinert einen Automaten G, (geschrieben als G>G‘) wenn G‘ mehr Eingabe akzeptiert und weniger Ausgaben zulässt als G.
- Die Definition der Verfeinerung erfolgt über eine alternierende Simulation. Für Interessierte sei an dieser Stelle auf das Papier von de Alfaro und Henzinger verwiesen.
- Die Verfeinerungsrelation ist, wie zu erwarten, transitiv.

Beispiel - Verfeinerung