Ausgewählte Paradigmen der Softwareentwicklung

Prof. Dr. Holger Schlingloff

Institut für Informatik
und
Fraunhofer FIRST

Worum geht es?

- **Paradigma:** eigentlich „Beispiel“ (griechisch „nebenher Gezeigtes“; para=neben, halb; deiknynai=zeigen, begreiflich machen)

  heute: Paradigma = Denkmuster, übergeordnetes Prinzip

- **Programmieren:** Gestalten von Vorschriften oder Handlungsabläufen

  (Choreographie eines Tanzes, Komponieren eines Musikstücks, Zusammenstellung eine Gala-Diners, Schreiben eines Film-Drehbuchs, ...) kreative Tätigkeit!

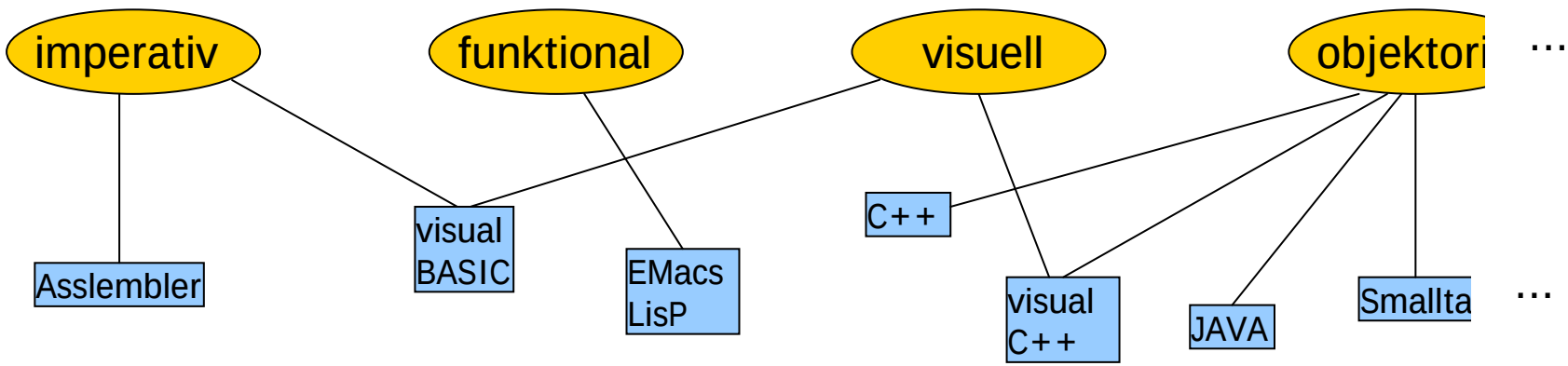
- **Programmierparadigma:** grundlegende Prinzipien, wie man Arbeitsvorschriften für Computer formulieren kann

Gliederung

- Paradigmen
- LOGO
- imperative, funktionale, objektorientierte, deklarative Konzepte
- Parallelismus
- Modellbasierung

Paradigmen und Sprachen

- Programmierparadigmen haben Einfluss auf die Gestaltung von Programmiersprachen
  - „visual C#.Net vereinigt Paradigmen der visuellen und objektorientierten Programmierung zur Programmierung von WebServices“
- Eine Programmiersprache realisiert ein bestimmtes Paradigma (oder auch mehrere)
  - „funktionale“ oder „objektorientierte“ Sprachen
- Programmiersprachen lassen sich auf Paradigmen zuschneiden



Entwurfsprinzipien

- Dilemma:
  - menschenorientiert oder maschinenorientiert?
  - leicht erlernbar oder effizient?
  - der Anschauung oder der Realität nachempfunden?

Konzeptionelle Klarheit wichtigstes Prinzip für Paradigmen!

Programme und Prozesse

- wie beschreibt man eine (Rechen-)Vorschrift?
  - durch Befehle und Datenstrukturen?
  - durch Angabe funktionaler Abhängigkeiten?
  - durch Klassen, Attribute und Methoden?
  - durch Regeln und Fakten?
- was ist ein (Rechen-)Prozess?
  - die Abarbeitung von Befehlen?
  - die Auswertung von Funktionen?
  - die Interaktion zwischen Agenten?
  - die Anwendung von Schlussregeln?

gängige Paradigmen

- imperativ
  - Programm als Folge elementarer Anweisungen
  - Programmausführung ist Abarbeitung der Anweisungen
  - Stichworte: Sprünge, Makros, Prozeduren, Variablen
- funktional (applikativ)
  - Programm als Funktion von Ein- in Ausgabewerte
  - Programmausführung ist Berechnung der Funktion
  - Stichworte: Lambda-Kalkül, LISP, SML, Haskell
- objektorientiert
  - Programm als Menge von Klassen und Objekten
  - Programmausführung ist Interaktion zwischen Objekten
  - Stichworte: ADT, Vererbung, Polymorphie, Agenten
- deklarativ (logisch)
  - Programm als Sammlung von Fakten und Regeln
  - Programmausführung ist Suche nach Antworten auf Anfragen
  - Stichworte: constraintbasiert, regelbasiert, prädikativ, relational
- parallel
- modellbasiert

Beispiel: LOGO

- nachfolgend als Beispiel dafür, wie sich verschiedene Paradigmen in einer Sprache bzw. Programmier-umgebung manifestieren lassen
- 1966 als Unterrichtssprache für den Schulunterricht entworfen (BBN/ArpaNet, ➡ Seymore Papert @MIT, Konstruktivismus) ➡
- ursprünglich nur 20 Kommandos und 14 Funktionen
- 1971: Schildkröte als Anschauungsobjekt (Dialogpartner / Personalisierung des Computers, *zuerst* reales Ausgabegerät, *dann* Bildschirmgrafik)

Turtle-Roboter
von 1948/49
(Smithsonian)

„Programmieren für Kinder“

- revolutionär in 1970, selbstverständlich in 2004 (Taschenrechner, Klingeltöne, Levelbuilder, ...)
- heutige Vorlesung: LOGO zur Demonstration elementarer Ideen („wie jedes Kind begreifen kann, ...“)
- Warnung vor Überheblichkeit
  - nichts ist „zu einfach“
  - LOGO bis zum Vordiplom
  - Entwicklung von kommerziellen Computerspielen mit LOGO
- Entwicklungen: LogoWriter, LegoLogo, MicroWorlds und StarLogo, MW Pro/EX
  - MWPro Entwicklung (deutsche Version) TZi Univ. Bremen, H. Schlingloff

Paradigma: Imperative Programmierung

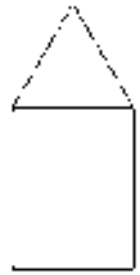
- Programm als Folge elementarer Anweisungen (mache erst dies, dann das, dann mach weiter mit jenem)
- Programmausführung ist Abarbeitung der Anweisungen der Reihe nach
- Wechselwirkung HW-Design / SW-Paradigma (von Neumann)
- Prototypisch: Assembler

```
mov $0x61, %al
inc %ah
jz  @loop
```

imperative Programmierung in LOGO

- Schildkrötenbefehle: `vorwärts, rückwärts, rechts, links, Stifthoch, Stiftabwärts, ...`
- Prozeduren als Makros:

```
Prozedur Viereck
  vw 100 re 90 vw 100 re 90 vw 100 re 90 vw 100 re 90
Ende
```



- erweiterte Befehle:

`Stiftfarbe!, Stiftstärke!, fülle, neuesBild, Startposition, Position!, versteckSchildkröte, ...`

Wiederholungen

- in Assembler meist durch Sprünge realisiert
  - „goto considered harmful“
- ursprüngliche Lösung: Endrekursion
  - schwer verständlich!
  - rein funktionales Paradigma

```
Prozedur Kreis
  vw 1 re 1
  Kreis
Ende
```

- spezielle Wiederholungsanweisungen:
  `immerwieder [ ... ]`

```
Prozedur Kreis
  immerwieder [vw 1 re 1]
Ende
```

Daten in LOGO

- endliche Wiederholungen:
  - `wiederhole 360 [vw 100 rw 100]`
  - `wiederholeIndex [i 10] [schreibeZeile :i * :i]`
  - `wiederholeListe ...`
- Daten sind
  - Wörter (Zeichenreihen)
  - Zahlen (spezielle Wörter)
  - Listen (in eckige Klammern eingeschlossen)
- Ein Programm ist eine Liste von Wörtern
  - Programme als Eingabedaten erlaubt
  - Interpreter statt Compiler

```
wiederhole 100
  [schreibe
    "|Ich darf im Unterricht nicht schwatzen. |]
```

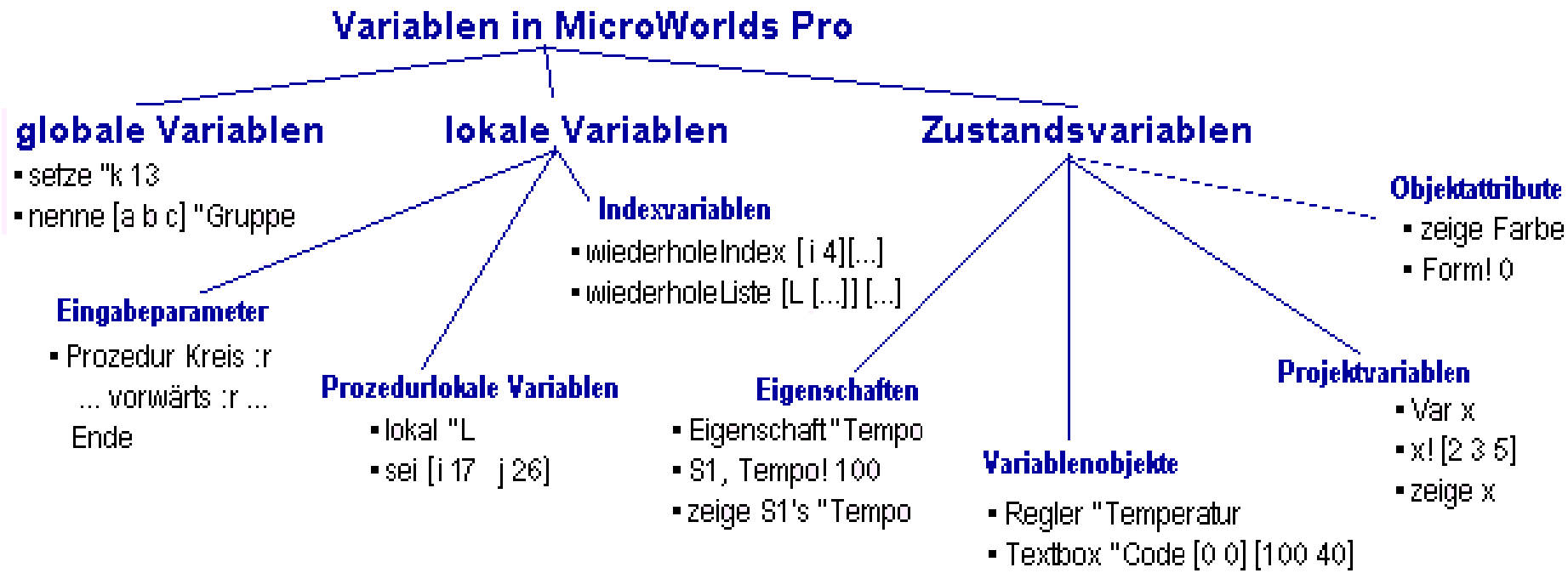
lokale und globale Variablen

- ein Original-LOGO-Programm von 1967
- Probleme der Gültigkeit und Lebensdauer von Variablen

```
to t1
5 print "function t1 entered"
10 make "A" "this is A from t1"
20 make "B" "this is B from t1"
30 print /A
40 print /B
50 do t2
55 print "just returned from t2"
60 print /A
70 print /B
```

```
to t2
10 print "function t2 entered"
20 print /A
30 print /B
40 make "A" "this is a new A from t2"
50 print /A
```

Variablen


Variablen in MicroWorlds Pro
globale Variablen
▪ setze "k 13
▪ nenne [a b c] "Gruppe
lokale Variablen
Zustandsvariablen
Indexvariablen
▪ wiederholeIndex [i 4][...]
▪ wiederholeListe [L [...]] [...]
Objektattribute
▪ zeige Farbe
▪ Form! 0
Eingabeparameter
▪ Prozedur Kreis :r
... vorwärts :r ...
Ende
Prozedurlokale Variablen
▪ lokal "L
▪ sei [i 17 j 26]
Eigenschaften
▪ Eigenschaft"Tempo
▪ S1, Tempo! 100
▪ zeige S1's "Tempo
Variablenobjekte
▪ Regler "Temperatur
▪ Textbox "Code [0 0] [100 40]
Projektvariablen
▪ Var x
▪ x! [2 3 5]
▪ zeige x

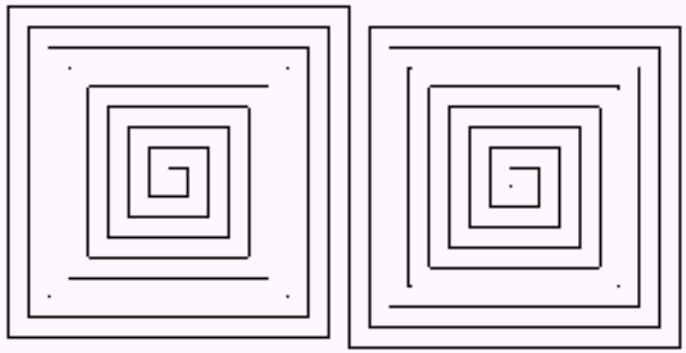
Paradigma: applikative Programmierung

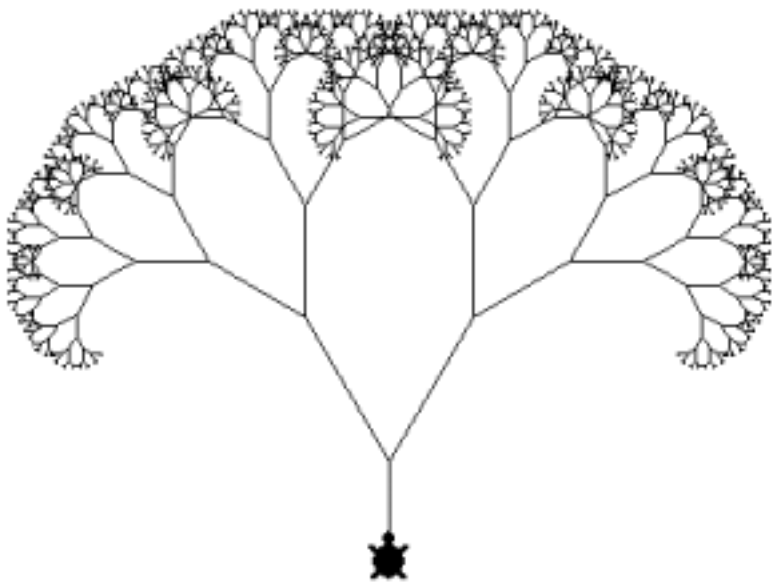
- Programm als Funktion von Ein- in Ausgabewerte, Programmausführung ist Berechnung der Funktion
- LOGO-Programme sind Anweisungen oder Ausdrücke
  - Anweisungen sind Folgen von Befehlen
  - Ausdrücke sind Aufrufe von Funktionen

- Beenden von Prozeduren mit `Rückkehr` (=`Stop`, bei Anweisungsprozeduren) oder `Rückgabe` (bei Funktionsprozeduren)
- Beliebige Rekursionen erlaubt, Endrekursion wird als Iteration gesondert behandelt

Beispiele

```
Prozedur Quadrale :Schritt
 wenn :Schritt > 170 [Rückkehr]
 vorwärts :Schritt
 rechts 90
 Quadrale :Schritt + 5
 vorwärts :Schritt
 links 90
Ende
```





```
Prozedur Binärbaum :Länge
 wenn :Länge < 1 [Rückkehr]
 vorwärts :Länge

 links 30 vorwärts :Länge
 Binärbaum :Länge * 2 / 3
 rückwärts :Länge rechts 30

 rechts 30 vorwärts :Länge
 Binärbaum :Länge * 2 / 3
 rückwärts :Länge links 30

 rückwärts :Länge
Ende
```

rekursive Funktionsprozeduren

```
Prozedur umgekehrt :w
 wenn leer? :w [Rückgabe "]
 Rückgabe
  Wort
   letztes :w
   umgekehrt ohneletztes :w
Ende
```

```
Prozedur Vokal? :z
  Rückgabe Element? :z [A E I O U Ä Ö Ü]
Ende
Prozedur ersetze-Vokal :x :y
  wenn Vokal? :x [Rückgabe :y]
  Rückgabe :x
Ende
Prozedur alles-mit :z :w
  wenn leer? :w [Rückgabe "]
  Rückgabe Wort
   ersetze-Vokal erstes :w :z
    alles-mit :z ohneerstes :w
Ende
```

```
zeige alles-mit "e "|Drei Chinesen mit dem Kontrabass|
Dree Chenesen met dem Kentrebess
zeige alles-mit "o "|Drei Chinesen mit dem Kontrabass|
Droo Chonoson mot dom Kontroboss
```

Insertion Sort

```
Prozedur sortiere :L
 Rückgabe sort :L []
Ende

Prozedur sort :L1 :L2
 wenn leer? :L1 [Rückgabe :L2]
 Rückgabe sort ohneerstes :L1 einsortieren erstes :L1 :L2
Ende

Prozedur einsortieren :El :Li
 wenn leer? :Li [Rückgabe miterstem :El :Li]
 wenn :El < erstes :Li [Rückgabe miterstem :El :Li]
 Rückgabe miterstem
  erstes :Li
  einsortieren :El ohneerstes :Li
Ende

zeige sortiere [5 3 7 2 4]
2 3 4 5 7
```

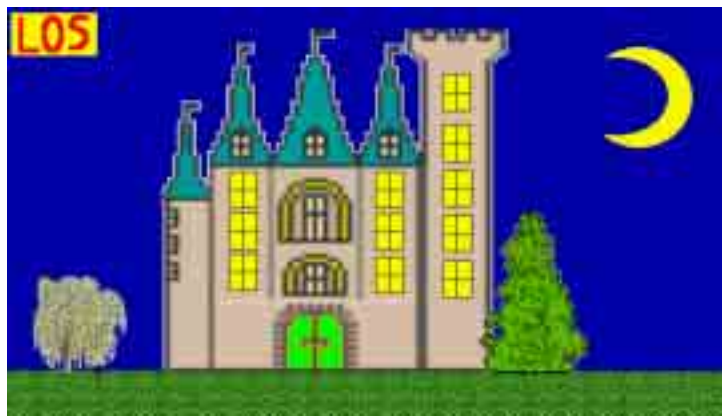
Paradigma: deklarative Programmierung

- Programm als Sammlung von Fakten
- Programmausführung als Suche nach Antworten auf Anfragen

- Logo-ähnliche Umgebung: Toon Talk (1995)
  - concurrent constraint programming
  - Klausel = Roboter
    Term = Schachtel
    Programmieren = Roboter trainieren
    Prozesse = Häuser

Paradigma: visuelle Programmierung

- Schildkröten können Formen annehmen
  - Möglichkeit der Animation durch Formwechsel
- Multimedia:
  - Neben Schildkröten gibt es
    - Textboxen
    - Knöpfe
    - Regler
    - Hyperlinks
    - Audio, Video, Excel-Tabellen



LOS

  - Aufruf eines Objektes durch Angabe des Namens

Paradigma: Parallelismus

- eigentlich kein eigenes Paradigma, sondern eine neue Dimension für die anderen Paradigmen
  - in applikativen oder deklarativen Sprachen parallele Auswertung von Funktionsargumenten oder Anfragen
  - in imperativen Sprachen gleichzeitige Ausführung mehrerer Befehle
  - in objektorientierten Sprachen simultane Interaktion zwischen Objekten

Parallelismus in LOGO

- mehrere Schildkröten gleichzeitig
  (in StarLogo mehrere hunderttausend!)
  - Jede Schildkröte hat einen eindeutigen Namen
  - Schildkröten tragen Instruktionen
- Prozessstart mit der Maus durch Anklicken
  - Interleaving (Zeitscheibenverfahren)
  - Befehl `wartebis` zum Warten auf Ereignisse
  - Farbwächter für einfache Ereignisse
  - Befehl `klickan` zum synchronen Start von Prozessen

```
Prozedur krabbeln
 iw [vw 1 re zz 3 li zz 3]
Ende

wartebis [( Abstand "S2 ) < 100]
krabbeln
```

Synchronisation

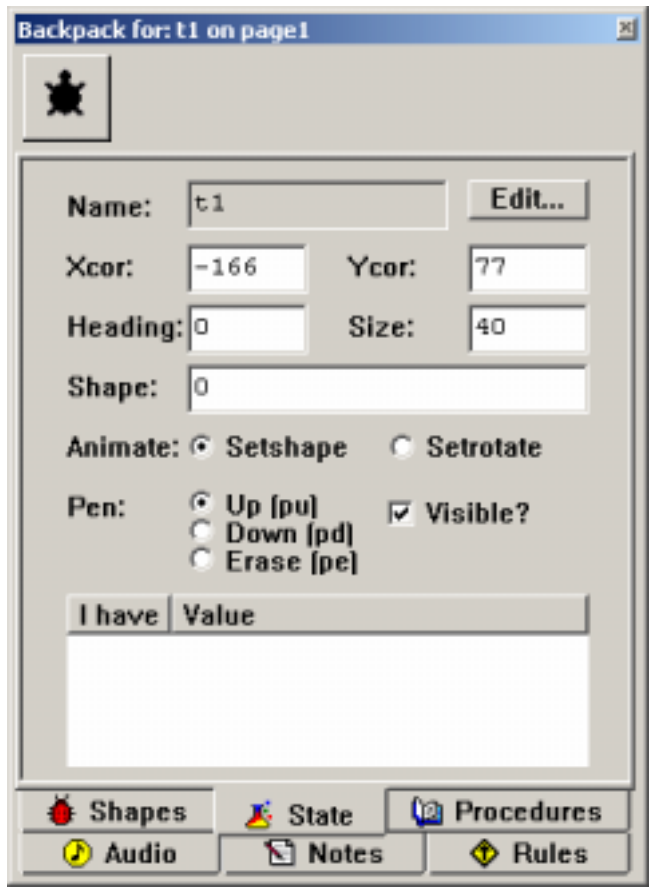
- Schildkrötenunabhängige Prozesse und Prozessfamilien
  - Befehl starte [...] zur Erzeugung eines Prozesses
- Synchronisation von Prozessen durch neue Prozesse
  - Befehl sobald [...][...] als Abkürzung:

```
sobald [yKoordinate > 100][rückwärts 100]
immerwieder [wartebis [yKoordinate > 100]
              rückwärts 100
              wartebis [nicht (yKoordinate > 100)]]
```

Paradigma: Objektorientierung

- Programm als Menge von Objekten
  Programmausführung ist Interaktion zwischen Objekten

- MW EX (2004): Schildkröten als „Objekte erster Klasse“
  - erweitertes Zustandskonzept
  - Rucksack mit eigenen Prozeduren
  - multiple Regeln für Ereignisse



Paradigma: modellbasiert

- Was hat das mit modellbasierter Entwicklung zu tun?

Noch nichts, aber:

- Prinzipiell bietet sich das Paradigma zur Umsetzung in solche Programmierumgebungen an
  (diskursive versus präsentationelle Kommunikation;
  Beispiel: I-Logix Rhapsody in C/C++ for UML 2.0)
- Ideen zur Einbringung vorhanden
- Forschungsbedarf!