

Programmieren lernen – Spezifikation durch Vertrag in der Informatik-Grundausbildung

Karlheinz Hug

Folien zum Bewerbungsvortrag für die C3-Professur
„Informatik, Softwaretechnologie und Betriebssysteme“
im Fachbereich Informatik, Hochschule Reutlingen

24. November 2003

Programmieren lernen

- Dogmen und Antithesen
- Spezifikation durch Vertrag
- Vertragsmethode in Informatik 1

Dogmen und Antithesen

Dogma 1:

Informatik-Grundausbildung =
Prozedural implementieren lehren

- Programmieren =
Spezifizieren + Implementieren + Testen
- Software entwickeln =
Analysieren + Entwerfen + ... + Validieren
- Prozeduraler Programmierstil
 - Seit 20 Jahren veraltet
 - Behindert softwaretechnisches Denken

Dogmen und Antithesen

Dogma 2:

Spezifikation ist ein Thema
für fortgeschrittene Semester

- „Was?“ kommt vor „Wie?“
- Erst Speisekarte schreiben, dann kochen \Rightarrow
Erst spezifizieren, dann implementieren
- Ohne Spezifikation ist Testen sinnlos
- Spezifikation durch Vertrag
(SdV) ist leicht zu lernen

Dogmen und Antithesen

Dogma 3:

Methodisches Testen ist ein Thema
für fortgeschrittene Semester

- Wer implementiert, muss auch testen
- ...& zwar sofort & systematisch
- Spezifikation durch Vertrag
liefert Testmethodik gratis mit

Dogmen und Antithesen

Dogma 4:

OO & Komponenten sind Themen für fortgeschrittene Semester

- Moderne imperative Programmiersprachen sind objekt- & komponentenorientiert
- Sie für prozeduralen Programmierstil zu missbrauchen ist kontraproduktiv
- Studierende mit skalierbaren Methoden an oo & ko Denken gewöhnen
- SdV skaliert: Modul - Klasse - Komponente

Dogmen und Antithesen

Dogma 5:

Entwurfsmuster sind ein Thema
für fortgeschrittene Semester

- „Guter“ oo Programmierstil
ohne Entwurfsmuster ~
„guter“ prozeduraler Stil
ohne strukturierte algorithmische Muster
- Studierende früh an
Problem- & Lösungsmuster heranzuführen
- Entwurfsmuster mit SdV erst richtig schön

Dogmen und Antithesen

Dogma 6:

Softwarequalität ist ein Thema
für fortgeschrittene Semester

- Was Hänschen nicht lernt,
lernt Hans nimmermehr
- Qualität gibt's nicht gratis
- Man muss sie bewusst & systematisch
in Software hinein konstruieren

Dogmen und Antithesen

Dogma 7:

GUIs & Dialogboxen sind Themen
für fortgeschrittene Semester

Erstsemester lernen
stromorientierte E/A
im Fernschreiberstil

- Wo bleibt der Praxisbezug?
- Dialoge mit SdV erst richtig schön

Dogmen und Antithesen

Fazit:

Alle wesentlichen Erkenntnisse
der Softwaretechnologie
der letzten 20 Jahre sind Themen
für fortgeschrittene Semester

- Das darf doch wohl nicht wahr sein!

Spezifikation durch Vertrag

SdV

- Grundkonzepte:
Invarianten,
Vor- & Nachbedingungen
- Verantwortlichkeiten
- Weitere Aspekte
- Leitlinien

Spezifikation durch Vertrag

Invarianten, Vor- & Nachbedingungen

INTERFACE Natural

QUERIES

n : INTEGER *-- Abfrage*

INVARIANTS

n >= 0 *-- Invariante*

ACTIONS

Set (IN newN : INTEGER) *-- Aktion*

PRE

newN >= 0 *-- Vorbedingung*

POST

n = newN *-- Nachbedingung*

END

END Natural

Spezifikation durch Vertrag

Verantwortlichkeiten

	Kunde	Lieferant
Pflicht	Die Vorbedingungen einhalten	Anweisungen ausführen, die die Nachbedingungen herstellen und die Invarianten erhalten
Nutzen	Ergebnisse/Wirkungen nicht prüfen, da sie durch die Nachbedingungen garantiert sind	Aufrufe, die die Vorbedingungen verletzen, ignorieren (Die Vorbedingungen nicht prüfen)

Spezifikation durch Vertrag

Ergebnisse in Nachbedingungen

QUERIES

Factorial (IN n : INTEGER) : INTEGER

PRE

$0 \leq n; n \leq nMax$

POST

$(n \leq 1) \text{ IMPLIES } (\mathbf{result} = 1);$

$(n > 1) \text{ IMPLIES } (\mathbf{result} = n * \text{Factorial } (n - 1))$

END

result ist Standardname für Ergebnis

Spezifikation durch Vertrag

Vorzustände in Nachbedingungen

INTERFACE Set [Element]

QUERIES

Count : INTEGER

Has (IN x : Element) : BOOLEAN

POST

result IMPLIES (Count > 0)

END

ACTIONS

Put (IN x : Element)

POST

Has (x);

OLD (Has (x)) IMPLIES (Count = **OLD** (Count));

NOT **OLD** (Has (x)) IMPLIES (Count = **OLD** (Count) + 1)

END



OLD (a)
liefert Wert
des Ausdrucks
a vor dem
Aufruf

Spezifikation durch Vertrag

Quantoren

```
INTERFACE SortableList [Element EXTENDS Comparable]
```

```
  QUERIES
```

```
    Count : INTEGER
```

```
    Item (IN i : INTEGER) : Element
```

```
      PRE 1 <= i; i <= Count  END
```

```
    Number (IN x : Element) : INTEGER
```

```
      POST
```

```
        0 <= result; result <= Count;
```

```
        (result > 0) IMPLIES
```

```
          (EXISTS i IN {1 .. Count} IT_HOLDS x = Item (i))
```

```
      END
```

```
    Sorted : BOOLEAN
```

```
      POST
```

```
        result = FOR_ALL i, k IN {1 .. Count} IT_HOLDS
```

```
          (i <= k) IMPLIES (Item (i) <= Item (k))
```

```
      END
```

Spezifikation durch Vertrag

Leitlinien

**SdV ist nicht nur eine Notation,
sondern eine Methode!**

Wende folgende Leitlinien iterativ an.

- (1) Trenne **Abfragen** und **Aktionen**.
- (2) Trenne **elementare** und **abgeleitete Abfragen**.
Spezifiziere elementare Abfragen nicht vertraglich, sondern nutze sie als Spezifikatoren.
Spezifiziere abgeleitete Abfragen in Termen elementarer Abfragen.
- (3) Formuliere **Invarianten** als allgemeine,...

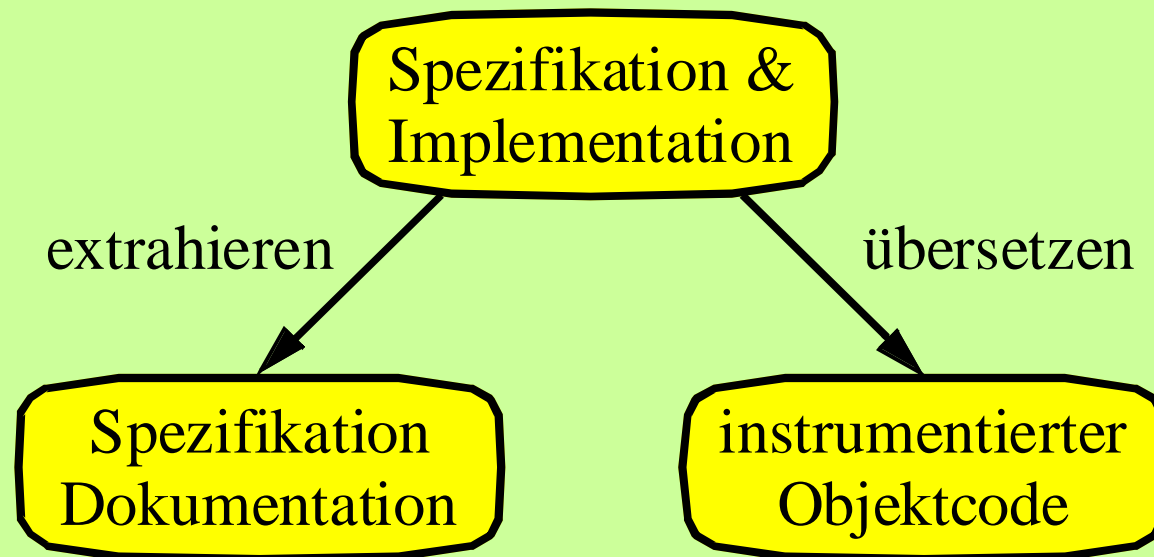
Vertragsmethode in der Lehre

- Von der Spezifikation zur Implementation
 - Eine Sprache
 - Zwei Sprachen
- Component Pascal
- Cleo
- Von der Spezifikation zur Dialogbox

Vertragsmethode in der Lehre

Von der Spezifikation zur Implementation

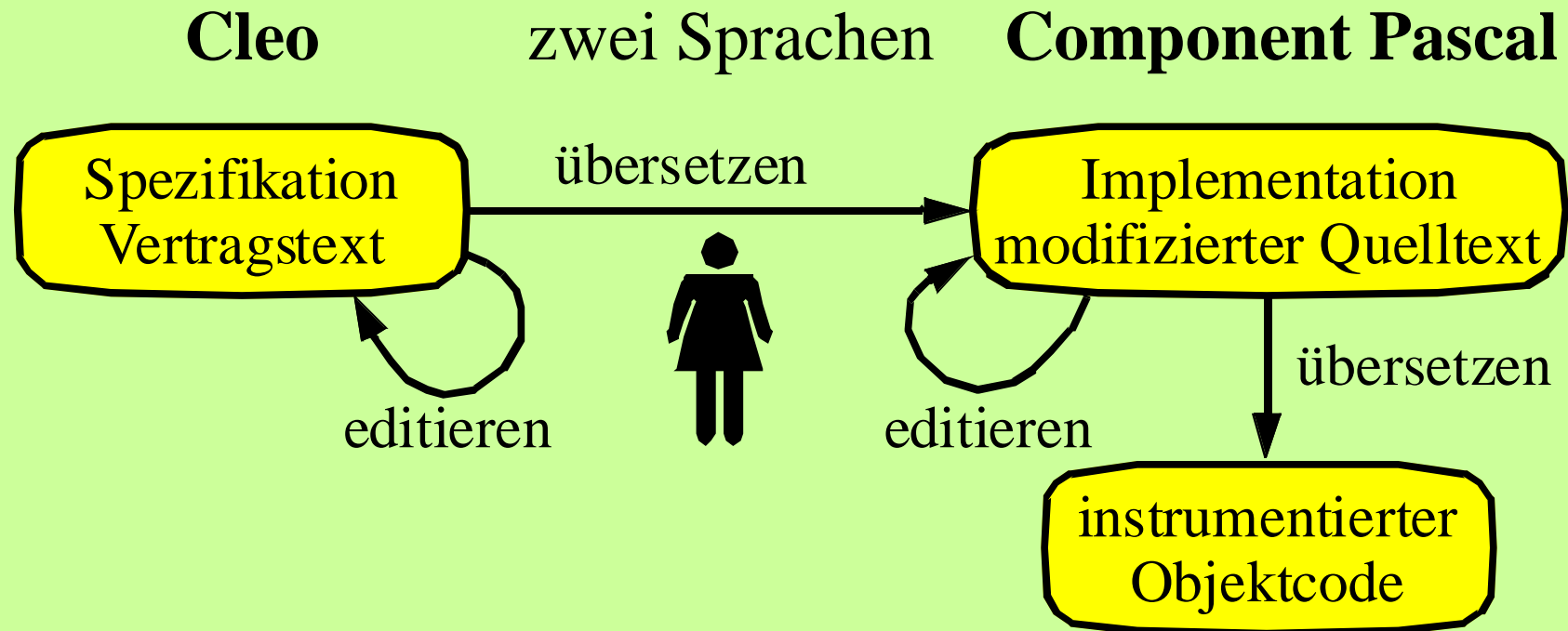
Eine Sprache: **Design by Contract = Specs'n'Checks**



- ☺ **Selbstdokumentation**
- ☺ **Einziges Quelldokument**
- ☺ **Nahtlose, reversible, werkzeuggestützte Entwicklung**

Vertragsmethode in der Lehre

Von der Spezifikation zur Implementation



- ☹ *Nicht* einziges Quelldokument
- ☹ *Keine* nahtlose, *keine* reversible Entwicklung
- ☺ **In der Lehre akzeptabel, ja vorteilhaft**

Vertragsmethode in der Lehre

Component Pascal

- Eine der ersten komponentenorientierten Sprachen
- Allgemeine Zusicherungen \Rightarrow minimale Unterstützung für SdV
- BlackBox Component Builder vertraglich spezifiziert
- SdV mit Component Pascal
siehe Hug: *Module, Klassen, Verträge*

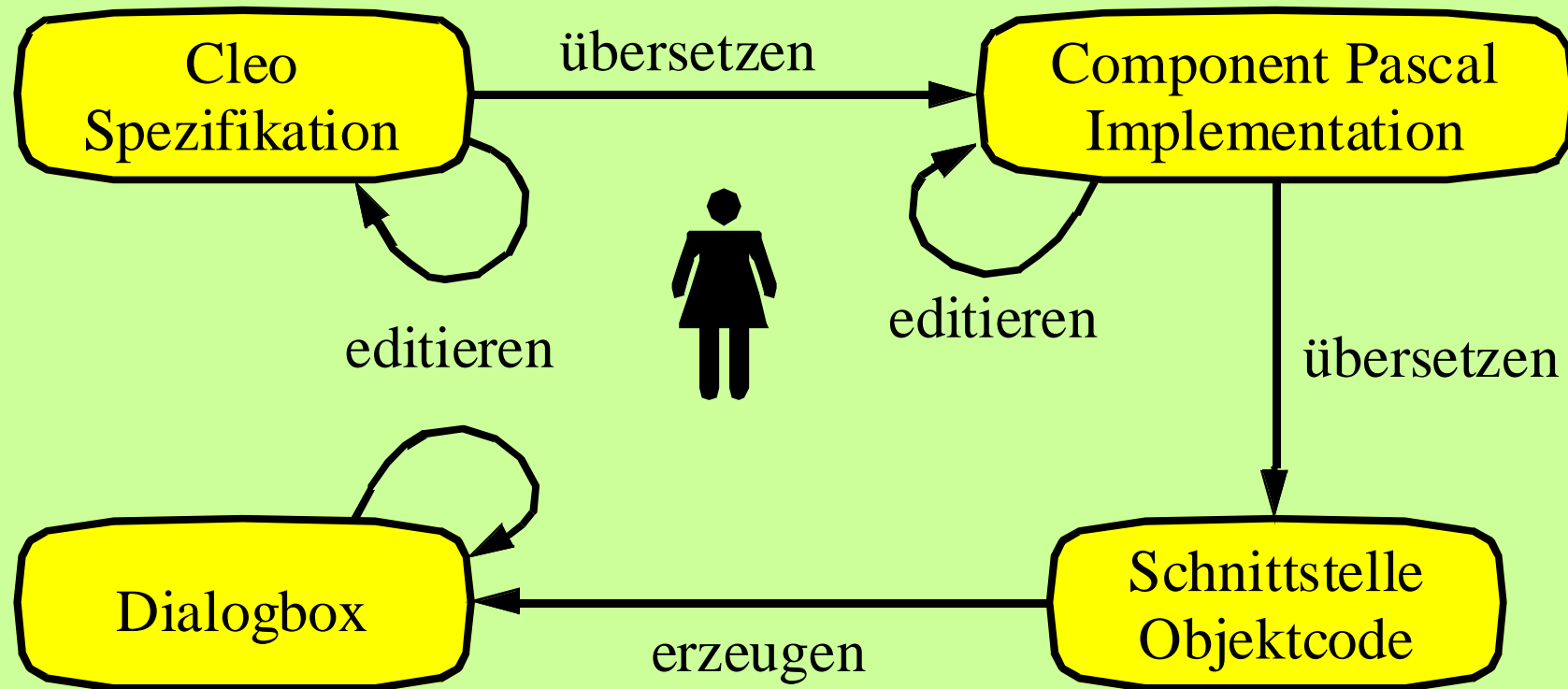
Vertragsmethode in der Lehre

Cleo

- **C**ontract **S**pecification **L**anguage
based on **E**iffel and **O**beron
- Konzepte von Eiffel,
Syntax von Oberon
- Klein, nur für Lehrzwecke
- Übersetzer
nach Component Pascal

Vertragsmethode in der Lehre

Von der Spezifikation zur Dialogbox



<C:\Hug\Artikel\C3\Spezifikation Dialogbox.pdf>

Vertragsmethode in der Lehre

Von der Spezifikation zur Dialogbox

- ☺ Mit Cleo: Funktionalität von Dialogen spezifizieren
- ☺ Mit CP: auf kreativen Teil der Implementation konzentrieren
- ☺ Dialoge fast automatisch erzeugen
- ☺ Mit Form-Editor: Dialog-Layout gestalten
- ☺ Exemplarischer werkzeuggestützter Software-Entwicklungsweg

Programmieren lernen

Danke für Ihre Aufmerksamkeit!