Software Engineering I

Prof. Dr. Martin Glinz

Kapitel 8

Realisierung

Universität Zürich
Institut für Informatik

Möglichkeiten der Realisierung

- Optionen
  - Entwerfen und Codieren
  - Entwerfen und Generieren
  - Konfigurieren vorhandener Komponenten
- Immer: integrieren und prüfen
- Wenn immer möglich: Nutzen vorhandener Komponenten

Detailentwurf und Codierung

Entwurf

- Entwurf von Algorithmen und Datenstrukturen
- Präzisierung des Lösungskonzepts

Codierung

- Umsetzen des Detailentwurfs in Programme

Codegenerierung

- Entwurf
  - Erstellung eines formalen oder teilformalen Entwurfsdokuments
  - Syntax muss den Anforderungen des Generators genügen
  - Dient als Grundlage für die Generierung
- Beispiele: Generierung von
  - Datenbankschemata
  - einfachen Anwendungen
  - Dialogen
  - Coderahmen

Systematisches Programmieren

- Geeignete Wahl von Algorithmen und Datenstrukturen
- Verwendung geschlossener Ablaufkonstrukte
- Saubere Gliederung der Software in Prozeduren und Module
- Verwendung symbolischer Konstanten
- Verwendung selbstdokumentierender Namen für Prozeduren, Variablen, Konstanten und Typen
- Dokumentation aller Definitionen durch Kommentare
  - Wirkung von Prozeduren (bzw. Methoden)
  - Bedeutung von Variablen und Konstanten
- Defensives Programmieren
- Vermeiden von Trickprogrammierung und Programmierung mit Nebenwirkungen

Beispiel 1: Auszug aus einem Java-Programm – 1

```
/* ---------------------------------------------------------------------
AUTOR:          Martin Arnold
PROJEKT:        Beispiel 8.1, Vorlesung Software Engineering I
JAVA:           Symantec Café, Projectmanager 8.2b3
COPYRIGHT:      Forschungsgruppe Requirements Engineering,
                Institut für Informatik, Universität Zürich
KLASSE:         MeasurementList
VERSION:        1.01 von M. Glinz am 18.12.1996
---------------------------------------------------------------------

ZWECK
Die Objekte der Klasse MeasurementList modellieren Messreihen.

VERANTWORTLICHKEITEN
Speicherung, Skalierung, Filterung und Abfrage von Messreihen.
*/

import java.awt.List;
```

Beispiel 1: Auszug aus einem Java-Programm – 2

```
public class MeasurementList extends List     //  MeasurementList wird Unterklasse von List
    {
/* --- KLASSEN-Variablen ---*/
    final private static double TOLERANZ = 0.30; // 30% zugelassene TOLERANZ nach
    unten

/* --- INSTANZ-Variablen ---*/
    double        wert;       // Listeneintrag: Messwert als reelle Zahl
    MeasurementList   naechstes;   // "Zeiger" auf nächstes Element der verketteten Liste

/*  Hinweis zur Dokumentation: Ein Objekt, auf das eine Methode (Operation) angewendet
    wird, wird aktuelles Objekt genannt.*/

/*  Hinweis zur Implementierung: Jede (auch die leere) Liste wird durch ein Objekt
    abgeschlossen, dessen Wert undefiniert ist und dessen Zeiger (naechstes) auf kein
    weiteres Objekt, sondern auf "null" zeigt.*/
```

Beispiel 1: Auszug aus einem Java-Programm – 3

```
void add(double element)
/*  Erste Version von Martin Arnold, 06.12.1996
    Letzte Aenderung von Martin Glinz, 18.12.1996
    PRE       –
    POST:     An die aktuelle Liste (d.h. diejenige, die mit dem aktuellen Objekt beginnt)
              wird ein Element mit der Zahl 'element' als Wert angehängt. Das aktuelle
              Objekt bleibt erstes Element der Liste.
*/
{
MeasurementList temp = this;    // Referenz auf aktuellen Listenbeginn (erstes Element)

while ( temp.naechstes != null ) // Suche nach letztem Element
   { temp = temp.naechstes; }

temp.naechstes = new MeasurementList();    // Zuweisung neuer Listenzeiger
temp.wert = element;  // Zuweisung des einzufügenden Wertes
}
...
```

Beispiel 2: Trickprogrammierung

Ein Unterprogramm in C zur Verkettung zweier Zeichenketten:

```
void strcat (char *s, char *t)
{
WHILE (*t++); FOR (t--;*t++=*s++;); /* t <- t concat s */
}
```

- Beide Schleifenrümpfe leer
  - Macht dieses Programm nichts?
  - Doch: ganze Funktionalität in Nebenwirkungen (side effects) versteckt
- Elegant und knapp zu schreiben
- Schwer zu verstehen und zu prüfen

Test und Integration

- Prüfung jeder Einzelkomponente
  - Syntaktische Richtigkeit (durch Compiler)
  - Inhaltliche Richtigkeit
    - Selbstinspektion durch Autorin/Autor
    - Formale Inspektion durch Dritte
    - Komponententest
- Schrittweise Integration der Komponenten
  - Aufwärts- oder Abwärtsintegration
  - Integrationstests
- Systemtest

Mehr zum Thema «Realisierung»

- In dieser Vorlesung nur summarisch behandelt

- Mehr zu Detailentwurf und Programmierung:
  - Kernvorlesung Software Engineering
  - auf Grundlage der Vorlesungen
    - Einführung in die Programmierung
    - Algorithmen und Datenstrukturen
- Mehr zu Test und Integration:
  - Kernvorlesung Software Engineering
  - teilweise Kapitel 9 dieser Vorlesung