Software-Engineering

Vorlesung 10 vom 20.12.2004

Sebastian Iwanowski
FH Wedel

Software-Engineering

Vorlesungsthemen:

1. Überblick über das Thema und die Vorlesung
2. Grundlegende Prinzipien
3. Softwareplanung
4. Systemanalyse
5. Softwareentwurf
6. CASE-Tools (UML und ARIS)
7. Aufwandsabschätzung
8. Qualitätsmanagement
9. Projektmanagement

Qualitätsmanagement

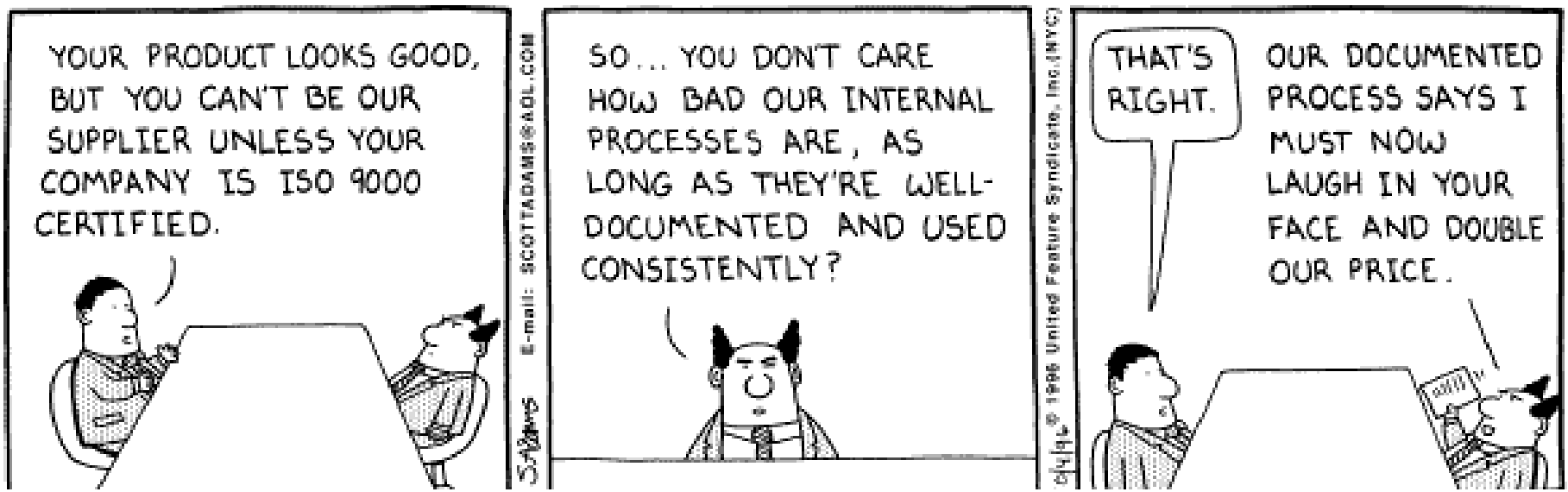
Richtlinien zur Herstellung von allgemeinen Produkten

- Die Qualität eines Produkts hängt maßgeblich von der Qualität seines Herstellungsprozesses ab.
- Insbesondere sollten alle Herstellungsvorgänge zurückverfolgt werden können.
- Für jeden Teilprozess muss es einen Verantwortlichen geben.
- Herstellungsprozess ist nach ISO 9000 und ISO 9001 (organisatorischer Rahmen) genormt.
- Die Einhaltung der Norm kann durch Zertifizierungsagentur bescheinigt werden.

Qualitätsmanagement

Übertragung der allgemeinen Richtlinien auf Software

- Für Software gibt es die Teilnorm ISO 9000-3.
- Zertifiziert wird der Herstellungsprozess, NICHT die Qualität des Endprodukts.
- Das hat dann z.B. folgende Auswirkungen:



Kriterien für SW-Qualität

Effizienz

- Effizienz ist bestimmt durch Verbrauch von Betriebsmitteln
- Unterschiedene Betriebsmittel bei SW: Speicherplatz, Laufzeit

Robustheit

- Robustheit wird bestimmt durch die Reaktion des Programms bei beliebiger Kommunikation mit der Umgebung
- insbesondere zu vermeiden: undefinierte Systemzustände und "Systemabstürze"
- wird beim Programmieren erreicht durch Abfangen aller möglichen Benutzereingaben
- Es reicht dir Beseitigung der Fehlersymptome, nicht der Ursachen

Kriterien für SW-Qualität

Verfügbarkeit

- Verfügbarkeit wird gemessen als Wahrscheinlichkeit, dass ein System zu einem gegebenen Zeitpunkt funktionsfähig ist.

Zuverlässigkeit

- Zuverlässigkeit ergibt sich aus Korrektheit, Robustheit und Verfügbarkeit.
- Zuverlässigkeit wird gemessen als Wahrscheinlichkeit, dass ein System seine Funktion während eines Zeitintervalls korrekt erfüllt.
- Zuverlässigkeit berücksichtigt Reparaturzeiten und Schwere der Fehler.
- Die geforderte Zuverlässigkeit wird üblicherweise in der Spezifikation festgelegt.

Kriterien für SW-Qualität

Sicherheit

- Sicherheit ist der Schutz gegen unerwünschte bzw. unerlaubte Verfälschung, Zerstörung oder Preisgabe von Daten.
- Sicherheit ist ein schwieriges Problem wegen der zunehmenden Dezentralisierung und Vernetzung.
- Sichrheit bezieht sich auch auf unbeabsichtigte Datenverluste, z.B. durch Stromausfall und Systemabsturz.

Kriterien für SW-Qualität

Bedienungsfreundlichkeit

Geforderte Merkmale der Bedienungsfreundlichkeit
(nach DIN 66234, Teil 8 und DIN EN ISO 9241):

- Aufgabenangemessenheit
- Selbstbeschreibungsfähigkeit
- Steuerbarkeit
- Erwartungskomformität
- Fehlerrobustheit (bei der Eingabe)

➢ wird ausführlich behandelt in der Vorlesung SW-Ergonomie

Kriterien für SW-Qualität

Verständlichkeit

- Maß für den Aufwand, ein (fremdes) Software-Produkt zu verstehen
- Verständlichkeit ist kein Selbstzweck, sondern erleichtert die Prüfbarkeit und Änderbarkeit.

Prüfbarkeit

- Möglichkeiten zum Testen eines Programms hinsichtlich Korrektheit, Robustheit und Zuverlässigkeit.
- Prüfbarkeit ist wesentlich abhängig von Modularität und Strukturierung.

Änderbarkeit

- Möglichkeiten zur Anpassung von Software an veränderte Einsatzbedingungen und Anforderungen
- Die Änderbarkeit sollte bereits bei Software-Entwicklung berücksichtigt werden.
- Änderbarkeit ist wesentlich abhängig von Modularität und wird durch das objektorientierte Konzept der Vererbung besonders unterstützt.

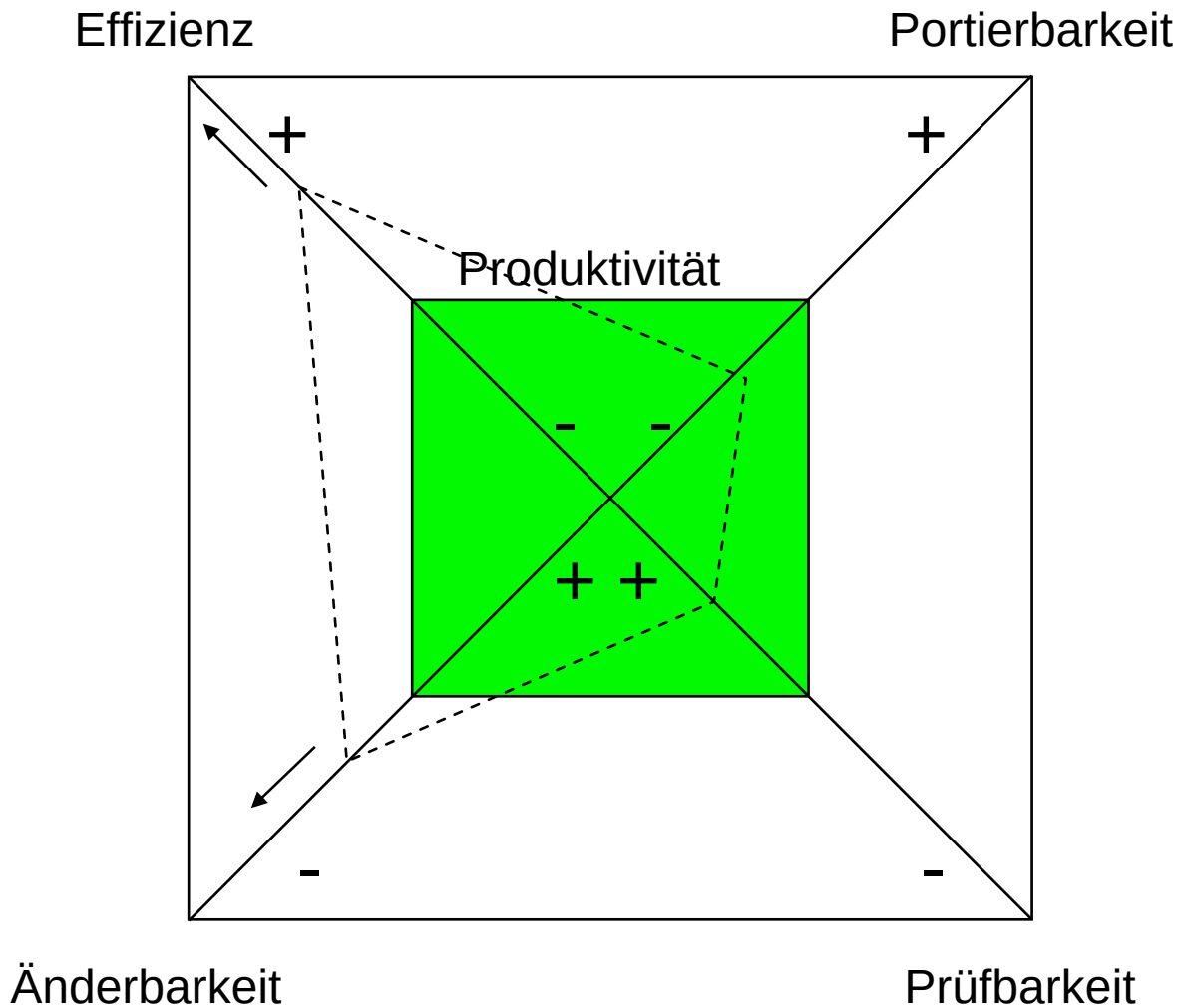
Kriterien für SW-Qualität

Wiederverwendbarkeit

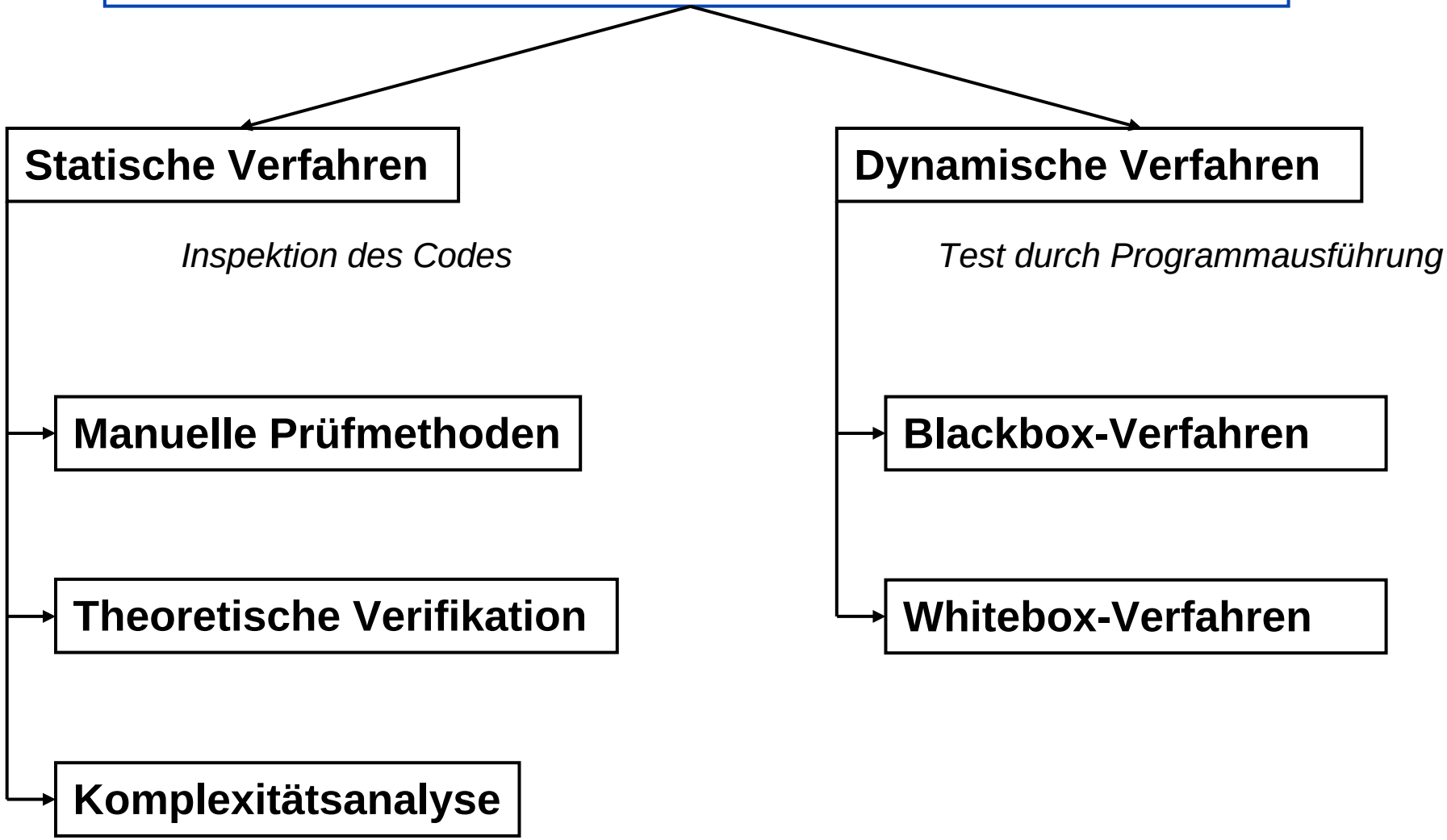
- Aufbau von Software-Bibliotheken und Nutzung objektorientierter Konzepte
- Wiederverwendbarkeit ist in erster Linie Qualitätskriterium für den Softwareentwickler
- Ziel: Senkung von Entwicklungskosten
- Wiederverwendbare Software verlängert die Lebensdauer derselben.
- Dadurch wird die Software besser gepflegt ➔ auch Anwendernutzen

Kriterien für SW-Qualität

Manche Qualitätskriterien sind gegensätzlich



Maßnahmen zur Qualitätssicherung



Die folgenden Details werden am Beispiel des Qualitätskriteriums Korrektheit ausgeführt

Manuelle Prüfmethoden

Bsp.: Reviews

- Durchführung in einem Team von ca. 4 Personen:
  - Programmersteller
  - Moderator
  - Programmspezialist
  - Testspezialisten
- Durchführung:
  - Programmierer erläutert die Programmlogik
    Anweisung für Anweisung
  - Teammitglieder stellen Fragen u. identifizieren Fehler

Formale Form des Reviews: Inspektion

Informelle Form des Reviews: Walkthrough

Dynamische Verfahren

Ausführung der Software mit Testdaten

- Auswahl von Testfällen
- Stichprobenartige Auswahl von Testdaten
- Möglichst realistische Testumgebung

Unterscheide:

Testfall:

- eine aus der Spezifikation oder dem Programm abgeleitete Menge von Eingabedaten zusammen mit den zugehörigen erwarteten Ergebnissen

Testdaten:

- Teilmenge der Eingabedaten der Testfälle, mit denen das Programm tatsächlich ausgeführt wird

Dynamische Verfahren

Grundsätzlicher Verfahrensunterschied:

Blackbox-Test

- Ableitung der Testfälle aus Spezifikation des Systems
- Innere Struktur des Programms wird nicht beachtet

Whitebox-Test

- Ableitung der Testfälle aus der Struktur des Programms
- Es wird angestrebt, möglichst alle Programmstrukturen zu durchlaufen (kontrollflussbezogenes Verfahren).
- Es wird angestrebt, möglichst alle Datenklassen zu testen, zwischen denen das Programm unterschiedet (datenflussbezogenes Verfahren).

und jetzt testen wir ein kleines Programm:

- **Berechne zu einem gegebenem Tag, in welchem Jahr er auf Weihnachten fällt !**

  Zu einfach !

- **Berechne zu einem gegebenem Tag, in welchem Jahr er auf Ostern fällt !**

  Außerdem:
  Berücksichtige auch die anderen von Ostern abhängigen Tage:
  Rosenmontag, Fasching, Aschermittwoch, Karfreitag, Ostermontag, Himmelfahrt, Pfingsten, Pfingstmontag, Fronleichnam

Blackbox-Test

Bestimmung der Testfälle:

- Bildung "funktionaler Äquivalenzklassen":
  Eingabe- und Ausgabemengen, die jeweils zu einer (Teil-) Funktionalität gehören
- Alle Werte einer Äquivalenzklasse verursachen ein identisches funktionales Verhalten eines Programms

Auswahl von Testdaten aus den Äquivalenzklassen:

- Zufällig
- Test spezieller Werte
- Grenzwertanalyse (primär Grenzbereiche der Eingabemengen als Testdaten)

Whitebox-Test

Kontrollflussbezogene Verfahren

Darstellung der Programmteile im Kontrollflussgraphen:

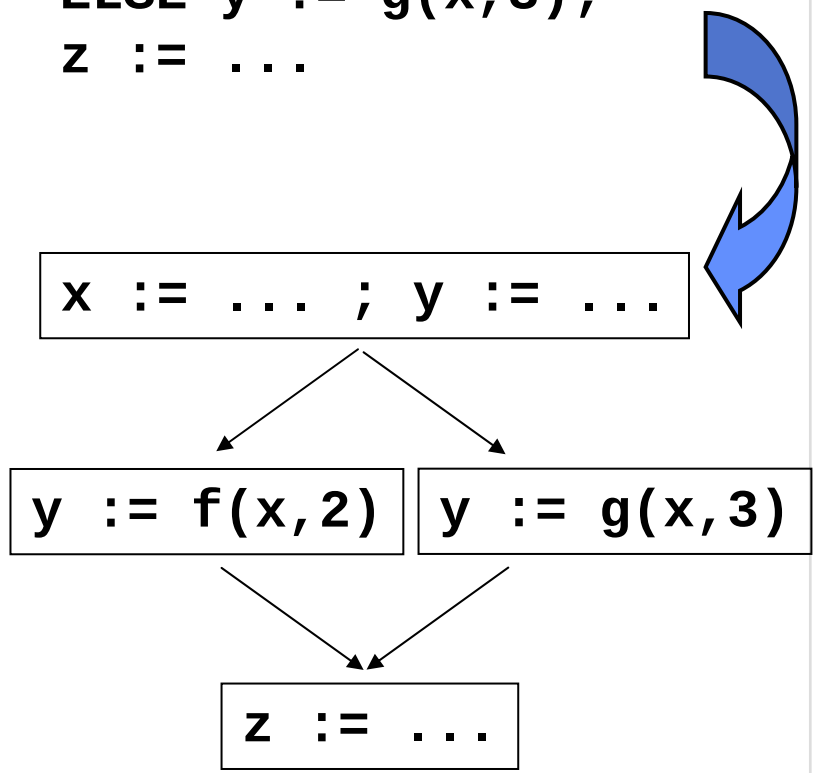
- Anweisungen (Anweisungsblöcke) als Knoten des Graphen
- Kontrollfluss als gerichtete Kanten zwischen Knoten

Überdeckung aller Kontrollstrukturen durch Testfälle:

- Gezieltes Durchlaufen (von Teilen) der Kontrollstrukturen durch geeignete Gestaltung der Testfälle
- Angestrebten/erreichten Überdeckungsgrad in Prozent angegeben

Anm.: 100 % sind unrealistisch !

```
x := ... ;
y := ... ;
IF (x > 5) AND (y < 2)
THEN y := f(x,2)
ELSE y := g(x,3);
z := ...
```



Whitebox-Test

Kontrollflussbezogene Verfahren

Anweisungsüberdeckung:

- Alle Anweisungen werden mindestens einmal ausgeführt.

Zweigüberdeckung:

- Alle Verzeigungen im Kontrollfluss werden mindestens einmal verfolgt.

Bedingungsüberdeckung:

- Alle booleschen Wertekonstellationen von (Teil-) Bedingungen werden einmal berücksichtigt.

Pfadüberdeckung:

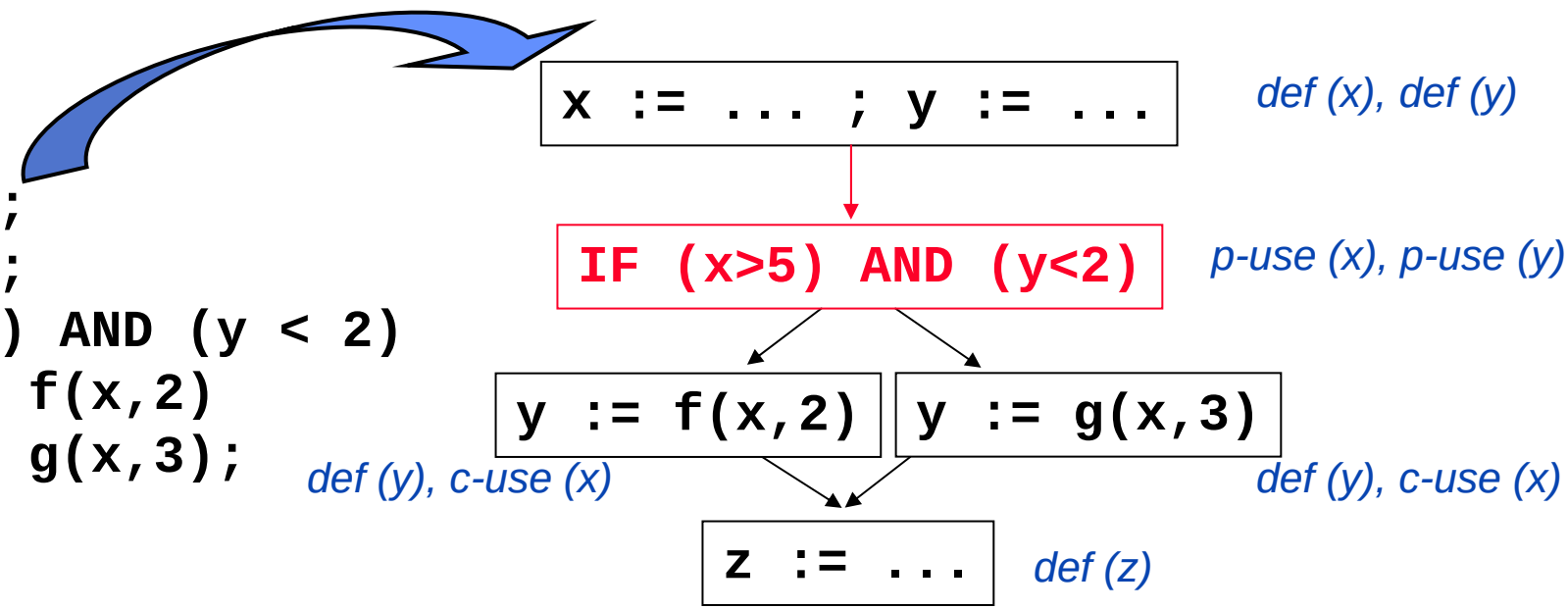
- Durchlaufen aller Pfade von Start- zum Endknoten:
  Das ist außer bei sehr kleinen Programmen nur theoretisch möglich.

Whitebox-Test

Datenflussbezogene Verfahren (Bsp.: defs/uses-Verfahren)

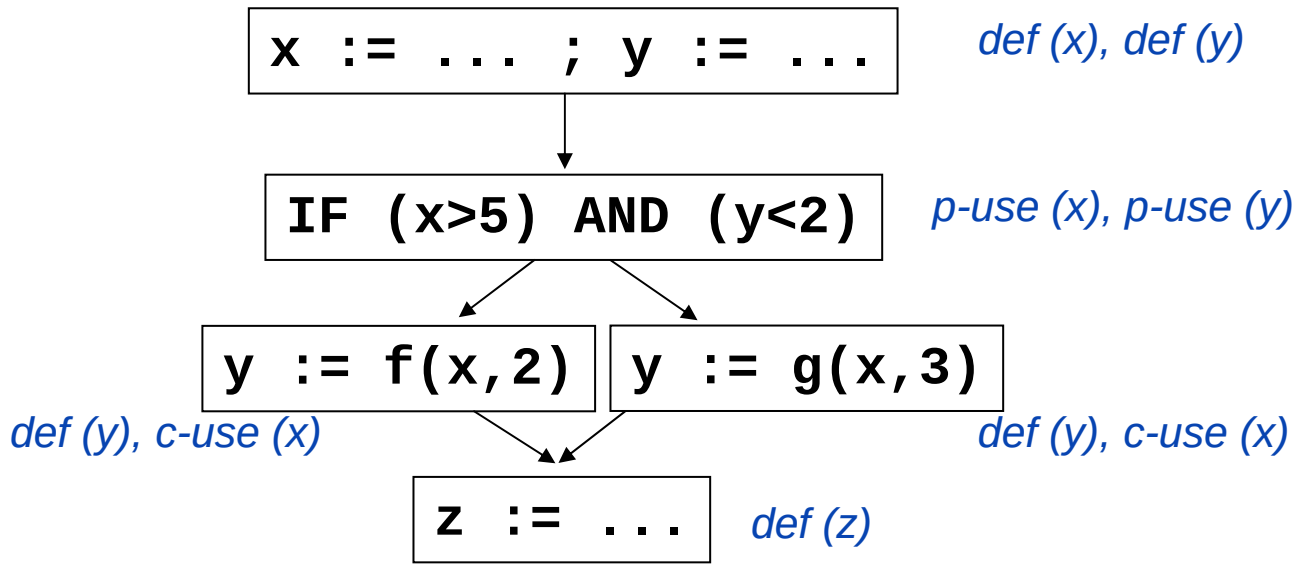
- Erweiterung der Kontrollflussgraphen um Datenflüsse
- Unterscheidung verschiedener Typen von Variablenzugriffen:
  - in Zuweisungen für die zugewiesene Variable (def)
  - in Wertausdrücken für ausgewertete Variable (c-use)
  - in Bedingungen für ausgewertete Variable (p-use)

```
x := ... ;
y := ... ;
IF (x > 5) AND (y < 2)
THEN y := f(x,2)
ELSE y := g(x,3);
z := ...
```



Whitebox-Test

Datenflussbezogene Verfahren (Bsp.: defs/uses-Verfahren)



- Betrachte „definitionsfreie Pfade“: Bis wohin wirkt sich die Zuweisung einer Variablen aus ?
- Unterscheidung verschiedener Typen von Datenflussüberdeckungen:
  - all defs: Jede Definition wird mindestens einmal benutzt
  - all p-uses: Jede Kombination Definition / p-use wird getestet
  - all c-use: Jede Kombination Definition / c-use wird getestet

Whiteboxtest von größeren Programmen

- Die vorgestellten Verfahren sind nur bei kleinen Programmen durchführbar.
- Größere Programme sind ohnehin modular aufgebaut

  => sie bestehen aus vielen kleinen Programmen.
- Teste die Module einzeln !
  (wird meistens bei der Implementierung gemacht)
- Außerdem müssen die **Modulverbindungen** getestet werden:
  - Von übergeordneten Programmen aus:
    Simulation der untergeordneten Module durch so genannte **Stub-Module**
  - Von untergeordneten Modulen aus:
    Simulation der Einbettung durch so genannte **Treiberprogramme**

Beim nächsten Mal:
Qualitätsmanagement (Abschluss)
Produktmanagement