Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

Programmierung 1

Für die Studiengänge BI und BWI

WiSe 25/26

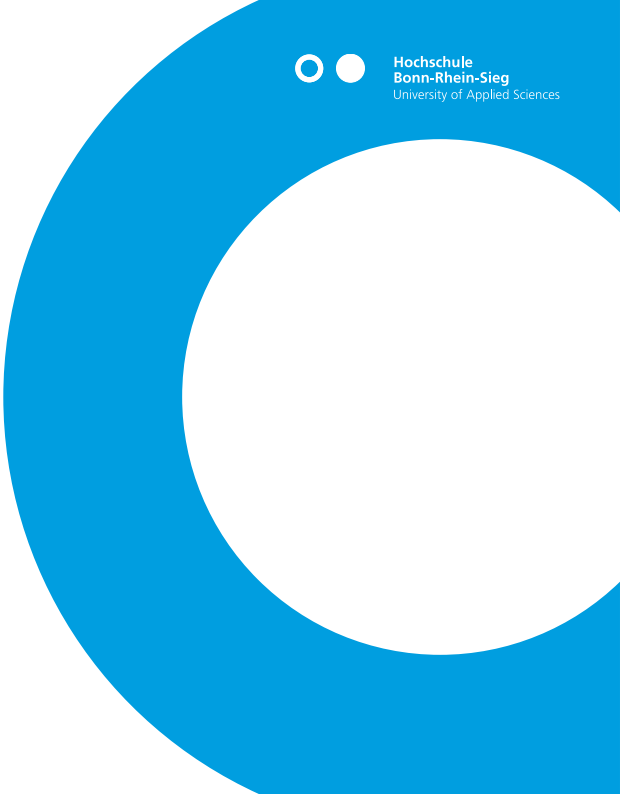
M. Sc. Moritz Balg

Inhaltsverzeichnis

Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

Einführung, Algorithmenbegriff 3

Einführung, Algorithmenbegriff



Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

Problemlösung

- Als Informatiker:in sind Sie Problemlöser
- Frage: wie kommt man (insbesondere bei komplexen Problemstellungen systematisch von einem Problem zu einer Lösung dieses Problems?
- Antwort für große Probleme: Phasen- und Prozessmodelle

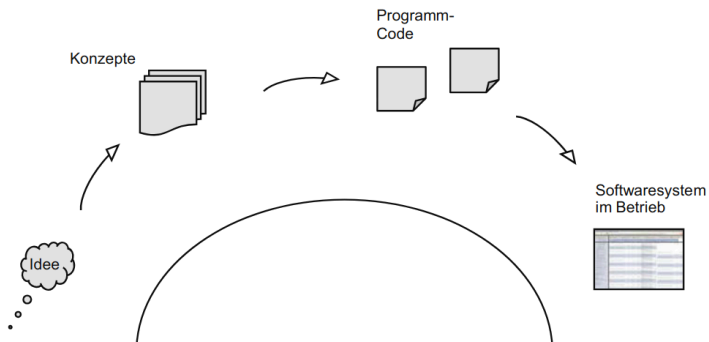


Abbildung 1: Phasen- und Prozessmodelle unterstützen bei der Lösungsfindung (Brandt-Pook & Kollmeier, 2016, S. 2)

Phasen- und Prozessmodelle

- Phasenmodelle teilen den Entwicklungsprozess in einzelne Phasen mit bestimmten Aktivitäten auf. Die Phasen werden in einer bestimmten Reihenfolge ausgeführt und liefern jeweils bestimmte Ergebnisse (Dokumente, Modelle, Software,...)
- In einem Prozessmodell wird u.a. die Reihenfolge der Phasen festgelegt
- Einfachste denkbare Reihenfolge: Phase $i + 1$ kann erst beginnen, wenn Phase i abgeschlossen ist (Wasserfallmodell)

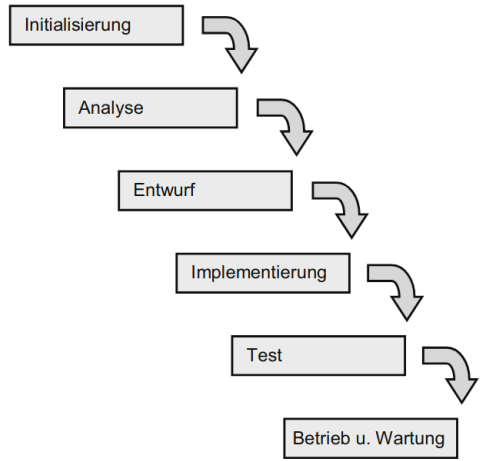


Abbildung 2: Phasen des Wasserfallmodells (Brandt-Pook & Kollmeier, 2016, S. 23)

Phasen der klassischen Software-Entwicklung

- **Analyse**
  - Fachliche Anforderungen ermitteln, Testfälle beschreiben
- **Entwurf**
  - Aus den Anforderungen wird ein (grobes) Software-Modell entworfen mit genau spezifizierten Schnittstellen zwischen einzelnen Teilen
- **Implementierung**
  - Basierend auf dem vorgegebenen Software-Modell werden Lösungen für Teilprobleme erstellt, die exakt die Schnittstellen des Software-Modells einhalten
- **Testen**
  - Unter Nutzung der Testfälle wird überprüft, ob die Software-Lösung das fachliche Problem löst
- **Betrieb und Wartung**
  - Durch veränderte Randbedingungen oder aufgetretene Fehler müssen Änderungen an der Software vorgenommen werden

Was ist ein Algorithmus?

Algorithmus - Definitionen

Definition: Algorithmus I

„Ein Algorithmus ist grob gesprochen eine wohldefinierte Rechenvorschrift, die eine Größe oder eine Menge von Größen als Eingabe verwendet und eine Größe oder eine Menge von Größen als Ausgabe erzeugt. Somit ist ein Algorithmus eine Folge von Rechenschritten, die die Eingabe in die Ausgabe umwandeln.“ (Cormen et al., 2013, S. 5)

Definition: Algorithmus II

„Ein Algorithmus ist eine Art von Black-Box, die mit Eingaben gefüllt wird und nach einer bestimmten Zeit eine Ausgabe produziert […]. Ein Algorithmus besteht dabei aus einer Menge von Aktionen“ (Dörn, 2019, S. 97)

Algorithmus Zusammenfassung

- Ein Algorithmus ist eine Arbeitsanweisung zur systematischen und schrittweisen Lösung einer Klasse von Problemen.
  - Beispiel: Sortieren von Matrikelnummern, Klausurnoten, Namen, ...
- Sie verarbeiten Ein- und Ausgaben durch entsprechende Zwischenschritte
- Algorithmen können auf unterschiedliche Weise formuliert sein (umgangssprachlich, als Programm, als Diagramm,...; gleich mehr dazu)

Darüber hinaus sollten Algorithmen folgende Eigenschaften erfüllen:

- ***Finitheit***: Das Verfahren muss in einem endlichen Text eindeutig beschreibbar sein.
- ***Ausführbarkeit***: Jeder Schritt des Verfahrens muss tatsächlich ausführbar sein.
- ***Dynamische Finitheit***: Das Verfahren darf nur endlich viel Speicherplatz benötigen.
- ***Terminierung***: Das Verfahren darf nur endlich viele Schritte benötigen.

Beispiel 1

- Problemstellung: Berechne den größten gemeinsamen Teiler zweier natürlicher Zahlen x, y, wobei nicht beide Zahlen gleich 0 sein dürfen
- Lösung seit > 2000 Jahren bekannt (Euklid):

```
ggT(x,y):
  Falls x = 0 ist, dann ist y das Ergebnis
  ansonsten
      wiederhole, solange y ≠ 0 gilt
          falls x > y ersetze x durch x – y
          ansonsten   ersetze y durch y - x
      x ist das Ergebnis
```

Listing 1: Euklidischer Algorithmus als Pseudocode

$$\mathrm{ggT}(x,y) := \begin{cases} y \text{ falls } x = 0 \\ x \text{ falls } y = 0 \\ \mathrm{ggT}(x-y, y) \text{ falls } x > y \\ \mathrm{ggT}(x, y-x) \text{ sonst} \end{cases}$$

Abbildung 4: Rekursive Definition zur Bestimmung des ggT

Beispiel: ggT

$$\mathrm{ggT}(16,10) = \mathrm{ggT}(6,10) = \mathrm{ggT}(6,4) = \mathrm{ggT}(2,4) = \mathrm{ggT}(2,2) = \mathrm{ggT}(2,0) = 2$$

Diskussion


```
ggT(x,y):
  Falls x = 0 ist, dann ist y das Ergebnis
  ansonsten
      wiederhole, solange y ≠ 0 gilt
          falls x > y ersetze x durch x – y
          ansonsten   ersetze y durch y - x
      x ist das Ergebnis
```

$$\mathrm{ggT}(x,y) := \begin{cases} y \text{ falls } x = 0 \\ x \text{ falls } y = 0 \\ \mathrm{ggT}(x-y,y) \text{ falls } x > y \\ \mathrm{ggT}(x,y-x) \text{ sonst} \end{cases}$$

- Nutzung von Variablen x und y, die Stellvertreter für konkrete Werte sind, mit denen der Algorithmus genutzt wird (vergleiche Funktionen in der Mathematik)
- Der Algorithmus enthält Anweisungen
  - zur Veränderung von Werten (Beispiel: ersetze x durch $x - y$)
  - mit denen die Reihenfolge der Ausführung dieser Einzelanweisungen beeinflusst wird (wiederhole, solange...)

Beispiel 2

- Problemstellung: Wie alt ist die älteste Person in diesem Raum?
- Eine Lösung:
  - Wähle eine beliebige Person aus und frage nach dem Alter. Dieses Alter ist das höchste Alter aller bisher befragten Personen.
  - Wiederhole, solange es noch ungefragte Personen in dem Raum gibt:
    - Frage eine Person nach dem Alter
    - Falls dieses Alter höher als das bisher bestimmte höchste Alter ist, so ist dieses Alter das neue höchste Alter.
- In unserer schönen heilen Algorithmuswelt ignorieren wir dabei an dieser Stelle, dass eine Person ihr Alter nicht nennen will, die mögliche zeitliche Varianz (Personen können aus dem Raum gehen / hereinkommen),...

Beispiel 3

Hochschule Bonn-Rhein-Sieg
University of Applied Sciences

- Problemstellung: Finde den kürzesten Weg vom Knoten **A** zum Knoten **E**
- Unterschiedliche Probleme lassen sich hiermit lösen:
  - 🗺️ Navigation & Routenplanung
  - 🖥️ Netzwerke & IT
  - ⚙️ Planung & Infrastruktur
  - 📊 Analyse & Forschung
- Mehr dazu in späteren Semestern

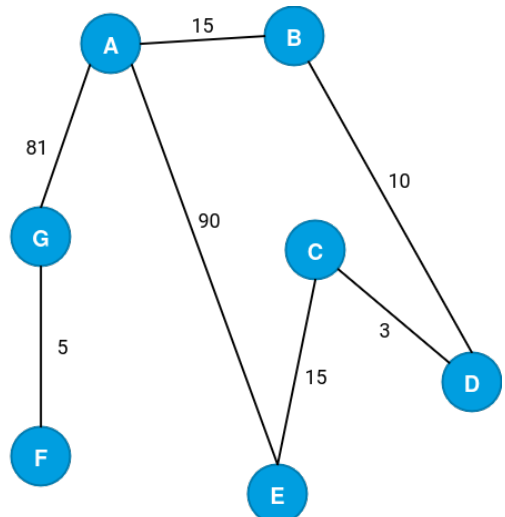


Abbildung 5: Ein einfacher ungerichteter Graph mit Kantengewichten

Algorithmische Grundbausteine

- In Mathematik, Maschinenbau, Medizin, Jura,... gibt es oft wiederkehrende fachspezifische Muster, die als Bausteine zur Lösung komplizierter Zusammenhänge genutzt werden
- Teilweise existieren sogar eigene Symbole für solche Muster
- Einfache Beispiele aus der Mathematik:
  - Summe von n Zahlen: Σ
  - Produkt von n Zahlen: Π
  - unbestimmtes Integral einer Funktion $f(x) : \int f(x)\ \mathrm{dx}$
- Beim Entwurf von Algorithmen gibt es ebenfalls solche Grundbausteine
- Alle gängigen Programmiersprachen unterstützen solche einfachen Entwurfsmuster direkt durch entsprechende Sprachkonstrukte
- Im Folgenden noch keine Java-Schreibweise!
- Zentraler Begriff in diesem Zusammenhang: **Anweisung**

Einzelanweisung

- Ein Einzelanweisung ist eine Anweisung

$$A$$

- Wir legen hier nicht fest, was genau eine Einzelanweisung sein kann. Das kann jede Anweisung sein (z.B. umgangssprachlich notiert), zu der allen Beteiligten die Bedeutung unzweifelsfrei klar ist.

Beispiele für Einzelanweisungen

- Ersetze x durch y (Andere Notationen dafür: x=y; x:=y; x $\sqcup$ y)
- Frage nach dem Alter
- x ist das Ergebnis
- Man nehme 200 g Mehl
- Morgens eine halbe Tablette nehmen

Sequenz

- Hat man beliebige Anweisungen A1bis An, so kann man diese zu einer Sequenz zusammenfassen

$$A_1; ...; A_n$$

- Eine Sequenz ist wieder eine Anweisung
- Die Bedeutung einer Sequenz ist, dass die einzelnen Anweisungen A1 bis An in genau dieser Reihenfolge hintereinander ausgeführt werden müssen

Beispiel für eine Sequenz

- Frage nach dem Alter; ersetze x durch den erfragten Wert

Selektion

- Hat man eine Bedingung B (ein logischer Ausdruck, der zu wahr oder falsch ausgerechnet werden kann) und eine Anweisung A, so kann man damit eine Selektion formulieren:

$$\text{if } B \text{ then } A$$

Auch eine Variante davon ist möglich:

$$\text{if } B \text{ then } A_1 \text{ else } A_2$$

- Eine Selektion ist wieder eine Anweisung
- Die Bedeutung einer Selektion ist, dass die Bedingung B ausgewertet wird. Anschließend wird die Anweisung A nur dann ausgeführt wird, wenn die Bedingung wahr war (Variante: A1 wird ausgeführt, wenn die Bedingung wahr ist, ansonsten wird A_2 ausgeführt)

Beispiele für eine Selektion

- if $x > y$ then ersetze x durch y

Mehrfachselektion

Falls man einen Wert $x \in \{x_1, ..., x_n\}$ hat und n Anweisungen $A_1, ..., A_n$, so wird eine *Mehrfachselektion* angegeben durch:

```
switch x: case x1:A1; case x2:A2; … case xn:An; end switch
```

- Eine Mehrfachselektion ist wieder eine Anweisung
- Die Bedeutung einer Mehrfachselektion ist wie folgt:
  - Zuerst wird x ausgewertet.
  - Falls $x = x_i$ für ein $i \in \{1, ..., n\}$, wird die Anweisung A_i ausgeführt, womit dann die gesamte Anweisung beendet ist.
  - Falls $x \neq x_i \forall i = 1, ..., n$, so ist die Anweisung beendet.

Mehrfachselektion (ii)

Beispiele für eine Mehrfachselektion

- ```
  switch Handyhersteller:
    case Apple:   nehme Batterietyp 1;
    case Samsung: nehme Batterietyp 2;
    case Xiaomi:  nehme Batterietyp 1;
  end switch
  ```
- ```
  switch Wochentag:
    case Montag:   x:= "Mo.";
    case Dienstag: x:= "Di.";
    case Mittwoch: x:= "Mi.";
    ...
  end switch
  ```

Iteration

- Hat man eine Anweisung A und eine Bedingung B, so kann man damit eine Iteration formulieren:

```
while B do A
```

- Eine Iteration ist wieder eine Anweisung
- Die Bedeutung einer Iteration ist wie folgt. Zuerst wird die Bedingung B ausgewertet. Ist der Wert von B falsch, so ist die Ausführung der Anweisung damit beendet. Ansonsten (B ist wahr) wird die Anweisung A ausgeführt und anschließend diese Ausführungsschritte wiederholt.

Iteration (ii)

Beispiel für eine Iteration

```
while ungefragte Personen im Raum sind do
  frage nach dem Alter
while y ≠ 0 do
  if x > y then ersetze x durch x – y
  else             ersetze y durch y - x
```

Struktur des Beispiels

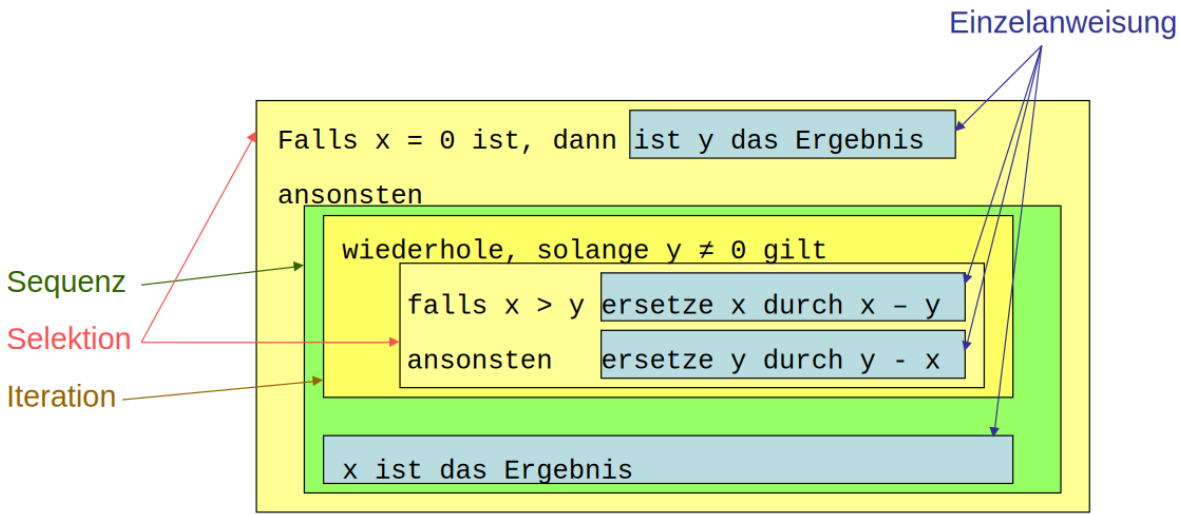


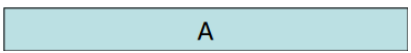
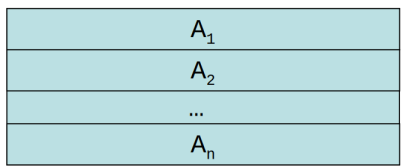
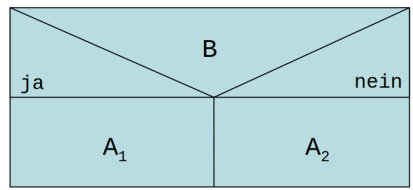
Abbildung 6: Struktogramm des Euklidischen Algorithmus

Darstellungsmöglichkeiten für Algorithmen

- Sehr, sehr kleine Problemlösung (Einzeiler; für uns weniger interessant):
  - Umgangssprachlich
  - Mathematische Formel
  - Einfache Skizze
  - ...
- Überschaubare Problemlösung (das ist unser Thema dieses Semester): Eine Reihe von Beschreibungsmöglichkeiten
  - UML Aktivitätsdiagramm
  - Struktogramm
  - Programm
- Große bis sehr große Problemlösung: andere Möglichkeiten wie zum Beispiel UML (Unified Modeling Language) (spätere Semester)

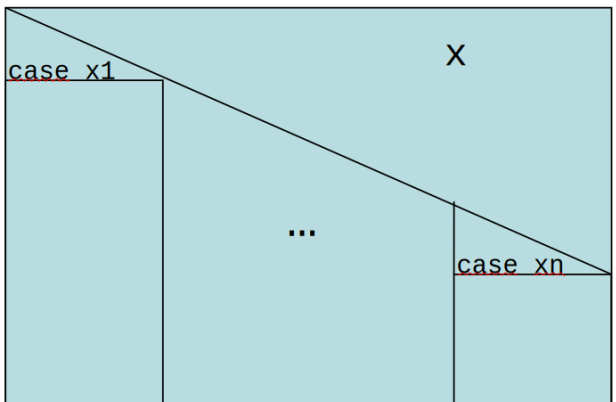
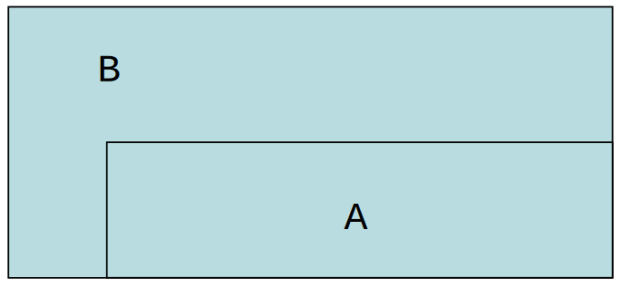
Struktogramm

- Die Elemente eines Struktogramms entsprechen den identifizierten Programmiermustern
- Struktogramme spielen in der Praxis keine Rolle, helfen aber sehr gut beim Einstieg in Programmiermuster / Programmstrukturen

Anweisung	Sequenz	Selektion
		
A	$A_1 ... A_n$	if B then A_1 else A_2

Struktogramm (ii)

Hochschule Bonn-Rhein-Sieg
University of Applied Sciences

Mehrfachselektion	Iteration
	
switch x : case x_1 : A_1; case x_2 : A_2; ... case x_n : A_n; end switch	while B do A

Struktogramm - Beispiel ggT

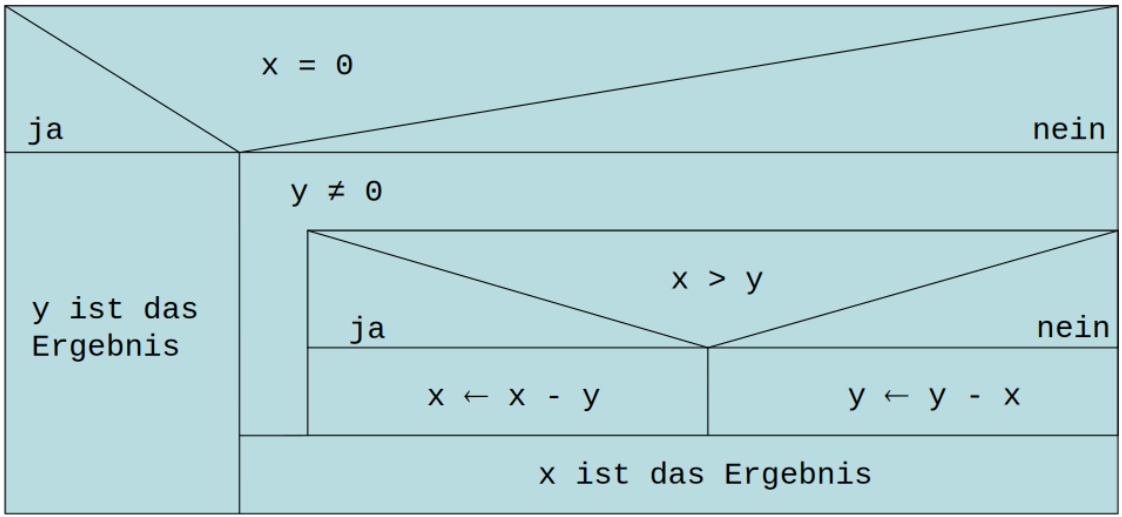


Abbildung 7: Der Euklidische Algorithmus als Struktogramm

UML Aktivitätsdiagramm

- UML: Unified Modeling Language (sehr umfangreich)
- Verschiedene Diagrammtypen zur Veranschaulichung von Zusammenhängen aus unterschiedlichen Sichten
- Hier interessant: Aktivitätsdiagramme der UML
- Grafische Darstellung (Zeichnung)
- Ein Aktivitätsdiagramm besteht aus Aktionselementen, die durch Pfeile miteinander verbunden werden können
- Pfeile geben mögliche Wege durch den Algorithmus an
- Eine Berechnung ist das Durchlaufen eines Aktivitätsdiagramms von einer Start- zu einer Stop-Markierung, wobei alle Aktionen auf dem durchlaufenden Weg ausgeführt werden
- Nur der hier relevante Teil von Aktivitätsdiagrammen wird beschrieben
- Nachteil von Aktivitätsdiagrammen:
  - schnell unübersichtlich
  - nicht alle bekannten Programmiermuster sind direkt darstellbar (Iteration)

Auswahl an grafischen Elementen

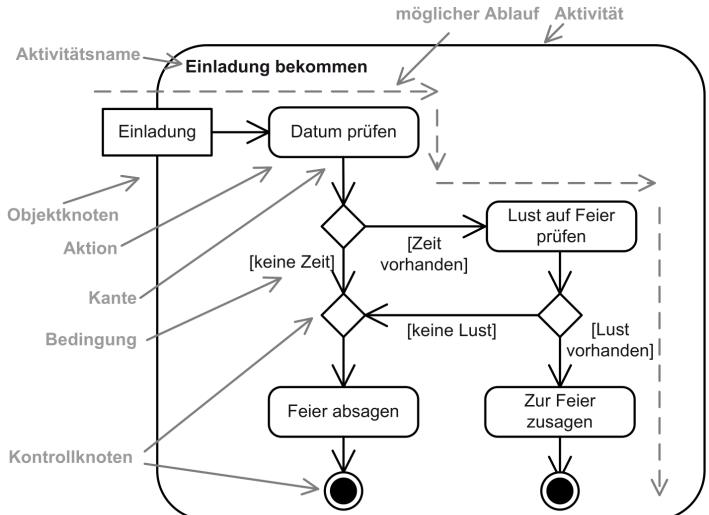


Abbildung 8: Einige Elemente des UML-Aktivitätsdiagramm (Rupp & Queins, 2012, S. 265)

Aktivitätsdiagramm - Beispiel ggT

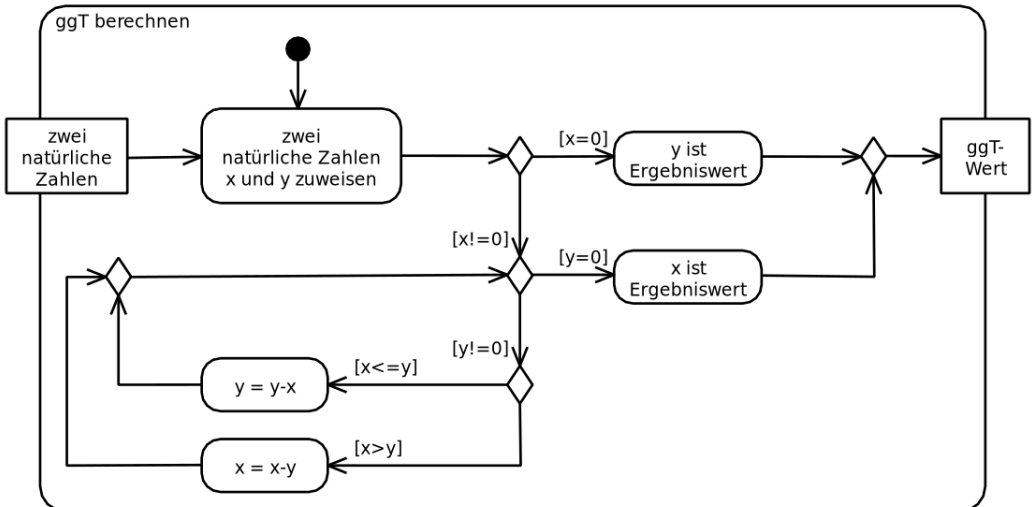


Abbildung 9: Der Euklidische Algorithmus als UML-Aktivitätsdiagramm

Programm

- Aktivitätsdiagramm und Struktogramm sind als Dokumentationswerkzeuge für überschaubare Algorithmen im Wesentlichen für den Menschen gedacht
- Ein Rechner könnte diese Darstellungsmöglichkeiten (es sind ja eigentlich Zeichnungen) nur sehr schwer verstehen
- Zur Kommunikation mit dem Rechner gibt es besser geeignete Formulierungsmöglichkeiten für Algorithmen (Programmiersprachen), mit denen auch sehr umfangreiche und sehr komplexe Algorithmen formuliert werden können
- Wer den letzten Punkt anzweifelt: stellen Sie die derzeit ca. 11 Millionen Zeilen Programmcode für den Linux-Kernel oder die ca. 50 Millionen Zeilen für Microsoft Windows als Struktogramm oder Programmablaufplan (PAP; Flussdiagramm) dar
- Alle Details zu Programmen und Programmiersprachen folgen später

Programm - Beispiel ggT

Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

```
public class GGT  {
    public static void main (String[] args)  {
        // Beispielwerte
        int x = 43158;
        int y = 26364;

        // gebe die Werte von x und y aus
        System.out.print ("Der ggT von " + x + " und " + y + " ist ");

        if(x == 0) {
            // gebe das Ergebnis / den ggT aus
            System.out.println (y);
        } else {
            while (y != 0)  {
                if (x > y) {
                    x = x - y;
                } else {
                    y = y - x;
                }
            }
            // das Verfahren brach ab und in x steht das Resultat
            System.out.println (x);
        }
    }
}
```

Zwischenstand

- Algorithmen lassen sich auf verschiedene Weise angeben
- Die UML (Unified Modelling Language) besitzt Aktivitätsdiagramme, in denen im Wesentlichen die Reihenfolge von durchzuführenden Aktivitäten grafisch dargestellt wird
- Ein Struktogramm eignet sich sehr gut, um die geschachtelte Struktur von Anweisungen darzustellen (Programmiermuster)
- Ein Programm ist nötig, um einen Algorithmus auf einem Rechner ausführen zu können

Zusammenfassung

- Das große Bild der Software-Entwicklung:
  - Phasen- und Prozessmodelle sind für größere Projekte unentbehrlich ($\rightarrow$ nachfolgende Veranstaltungen zum Software-Engineering)
  - In dieser Veranstaltung beschäftigen wir uns aber ausschließlich mit überschaubaren Problemstellungen, bei denen diese Instrumente zu komplex sind
- Algorithmusbegriff
- Häufig wiederkehrende Entwurfsmuster in Algorithmen
- Darstellungsmöglichkeiten: Aktivitätsdiagramm, Struktogramm, Programm

Literatur

Brandt-Pook, H., & Kollmeier, R. (2016). *Softwareentwicklung kompakt und verständlich*. Springer Vieweg.

Cormen, T. H., Leiserson, C. E., Rivest, R., & Stein, C. (2013). *Algorithmen - eine Einführung*. De Gruyter Oldenbourg.

Dörn, S. (2019). *Java lernen in abgeschlossenen Lerneinheiten*. Springer Vieweg.

Rupp, C., & Queins, S. (2012). *UML 2 glasklar*.