Kapitel 10, 11, 12 gehören zusammen

Ein einfacher Softwareentwicklungsprozess von der Anforderungsanalyse über den Entwurf zur Programmierung und zum Test

Java-Beispielsammlung: Seiten 40 - 48

Sofort mit dem Programmieren zu beginnen, würde scheitern: 6 Dateien mit 378 Zeilen

10. Softwareentwicklung: Anforderungsanalyse und Problemdefinition

Java-Beispiele:
- Maze.java
- Mouse.java
- MouseMaze.java
- Maze.Test.java
- Easel.java
- SoftFrame.java

Das nicht-triviale Java Beispiel 'Maus im Labyrinth'

Ein einfacher Softwareentwicklungsprozess von der Anforderungsanalyse über den Entwurf zur Programmierung und zum Test

Kapitel 10, 11, 12

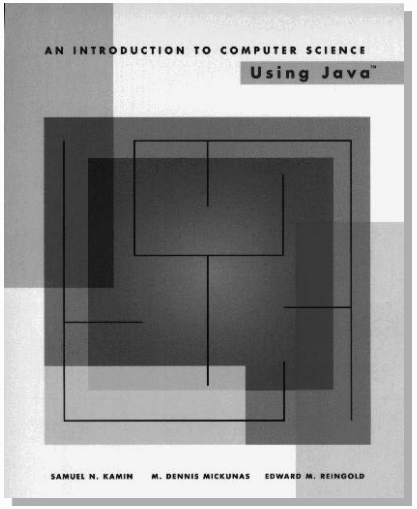
Neue Qualität der Aufgabenstellung

Bisher:
- kleine Programme
- Anforderungen / Aufgabenstellung vorgegeben
- Stuktur und Klassen immer vorgegeben

Jetzt:
- Gesamtprozess einer Softwareentwicklung
- auch: Anforderungen diskutieren
- auch: Softwarearchitektur ermitteln, d.h. wie findet man Klassen zum Problem
- auch: systematischer Test

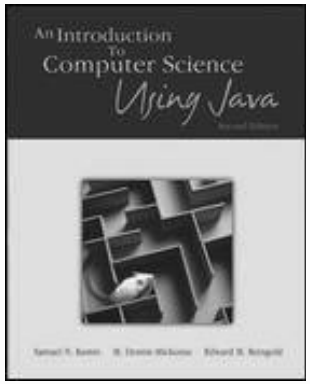
Quellen

Basierend auf einer Idee von S.N. Kamin, M.D. Mickunas, E.M. Reingold: „An introduction to computer science – Using Java“, McGraw-Hill, 1998



Quellen

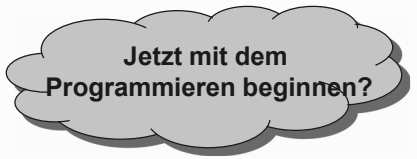
S.N. Kamin, M.D. Mickunas, E.M. Reingold: „An introduction to computer science – Using Java“, McGraw-Hill, 2nd ed., 2002



Aufgabenstellung: ‚Maus im Labyrinth‘

Aufgabe:

Entwickeln Sie ein Programm, das die Bewegung einer Maus durch ein Labyrinth (Irrgarten) vom Eingang zum Ausgang simuliert!



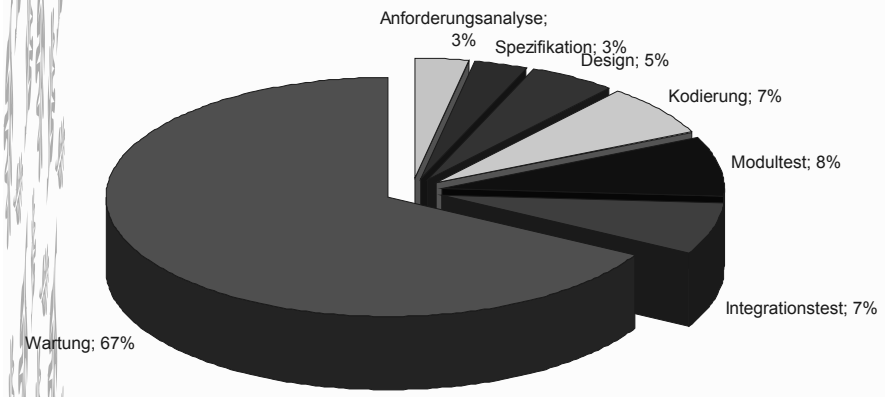
Fakten zu den Softwareentwicklungsphasen

Die meisten Fehler in Softwaresystemen gehen auf Fehler im Verständnis des Problems zurück (Anforderungsspezifikation).

Die Softwareentwicklung besteht nur zu einem geringen Teil aus der Programmierung:

- Anforderungsanalyse: 20 %
- Design: 15 %
- Implementation: 20 %
- Test: 45 %

Kostenverteilung im Software Life-Cycle



Quelle: Klösch, Gall, 1995, S.3

Lernziele

- Dieses Beispiel soll die Wichtigkeit einer vollständigen und korrekten Anforderungsanalyse für den Gesamterfolg des Projekts illustrieren.
- Bevor mit der Implementation begonnen wird, muss neben der Anforderungsspezifikation das Design des Programms geklärt werden: Welche Komponenten gehören zur Architektur?
- Ein zu zeitiger Implemetationsbeginn kann zum Scheitern des Projekts führen.

Entwicklungsprozess des Programms ‚Maus im Labyrinth'

→ Anforderungsanalyse

Design

Implementation und Test

Offene Fragen?

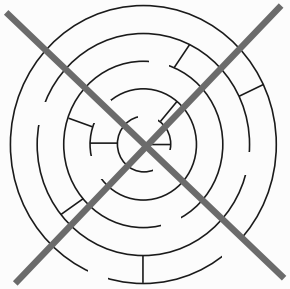
Entwickeln Sie ein Programm, das die Bewegung einer Maus durch ein Labyrinth (Irrgarten) vom Eingang zum Ausgang simuliert!

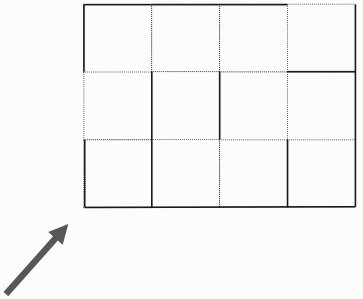
- Wie sieht das Labyrinth aus?
- Welche Fähigkeiten besitzt die Maus (z. B. welche Bewegungen)?
- Was ist die Ausgangsposition der Maus?
- Was, wenn es keinen Weg vom Eingang zum Ausgang gibt?
- In welcher Form soll die Lösung ausgegeben werden?

Wie sieht das Labyrinth aus?

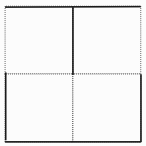
Glendurgan Garden, Cornwall, England

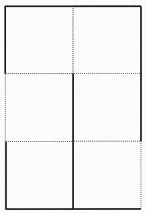
Wie sieht das Labyrinth aus?

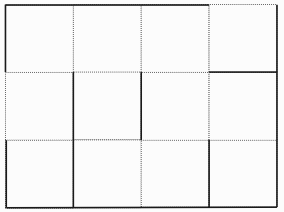


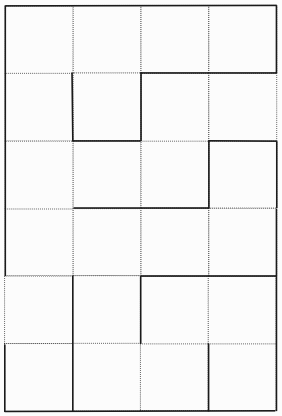


Mögliche Formen des Labyrinths









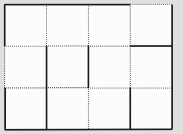
Anforderungsspezifikation (1)

Entwickeln Sie ein Programm, das die Bewegung einer Maus durch ein Labyrinth (Irrgarten) vom Eingang zum Ausgang simuliert!

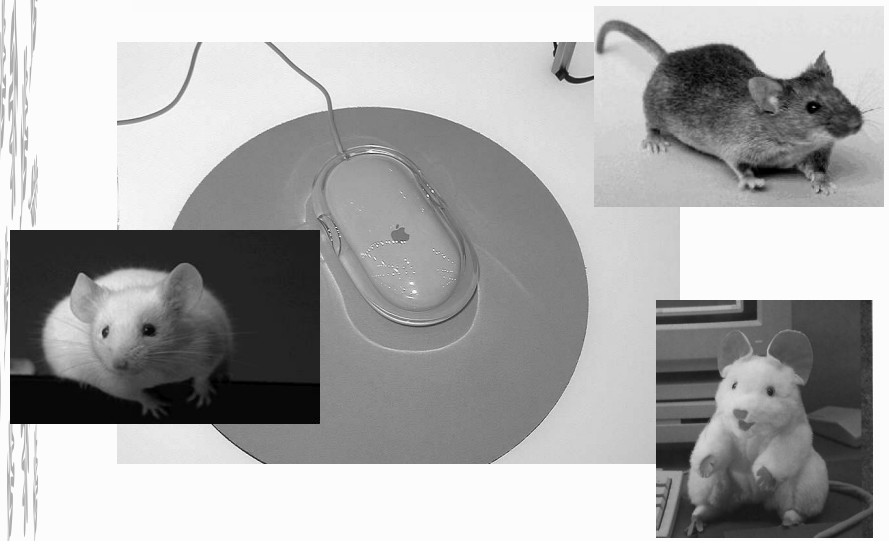
1. Das Labyrinth

Ein Labyrinth ist eine rechteckige Anordnung quadratischer Räume. Zwischen zwei benachbarten Räumen befindet sich entweder eine Wand oder eine Öffnung. Das Labyrinth wird durch eine zusammenhängende Wand umschlossen, die an einer oder zwei Stellen (ein Eingang und evtl. ein Ausgang) durchbrochen wird. Die Größe des Labyrinths (d. h. Länge und Breite) ist variabel.

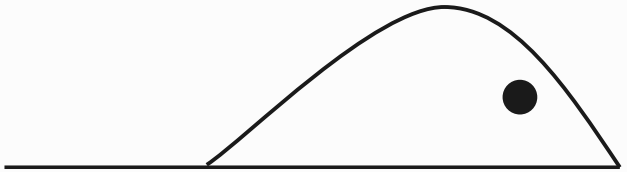
Beispiele:



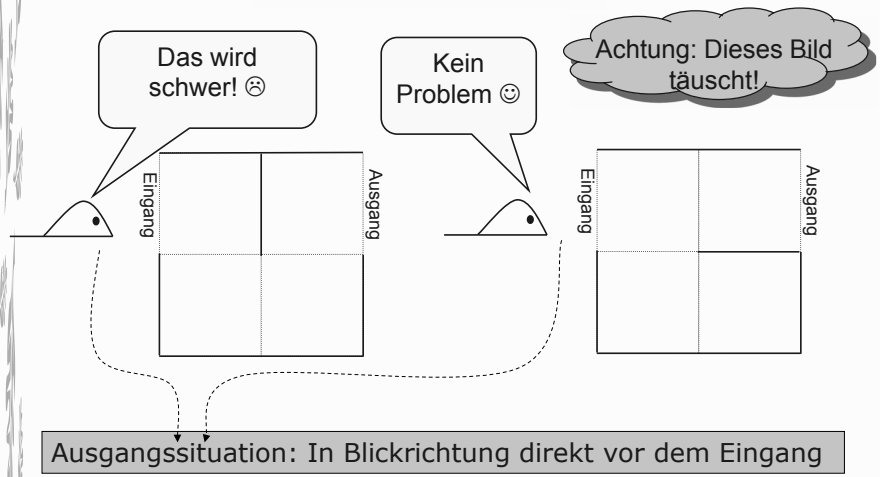
Wie sieht die Maus aus?



Unsere Maus



Maus mit verschieden Labyrinthen konfrontiert

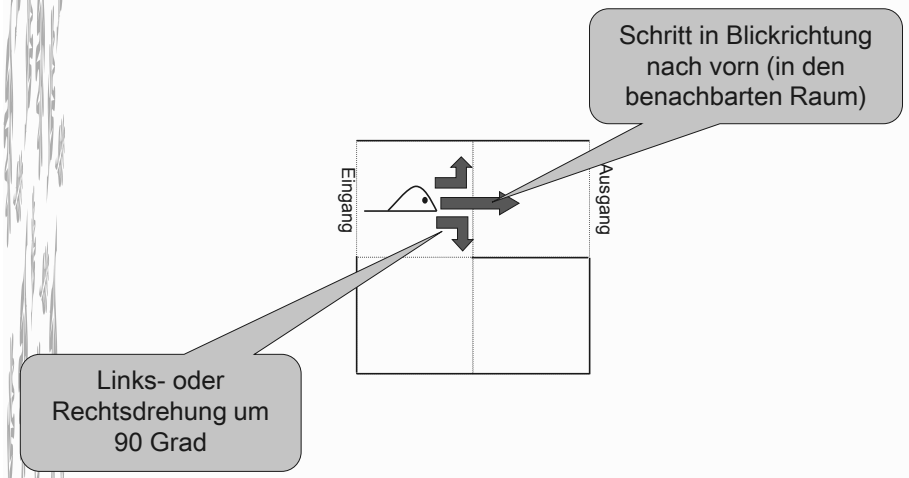


Ausgangssituation: In Blickrichtung direkt vor dem Eingang

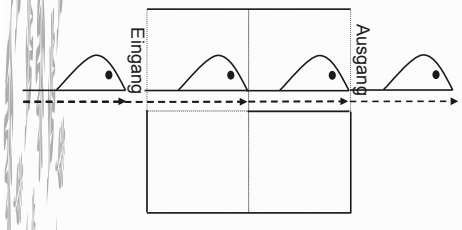
Welche Bewegungsmöglichkeiten hat die Maus?

- Einen Schritt vorwärts gehen
- Drehung nach links
- Drehung nach rechts

Bewegungsmöglichkeiten der Maus



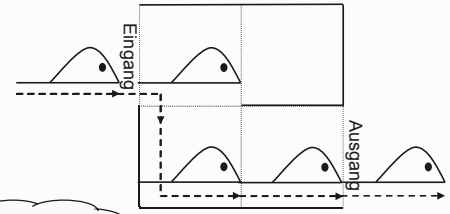
Zwei programmierte Lösungen



Was ist bei diesen Lösungen verdächtig?

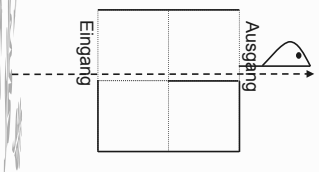
- Richtige Lösung für das falsche Problem.
- Jeweils den kürzesten Pfad gefunden.

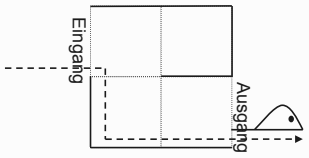
→ Anforderungsspezifikation: unvollständig



Was könnte fehlen?

Anforderungsspezifikation ist unvollständig

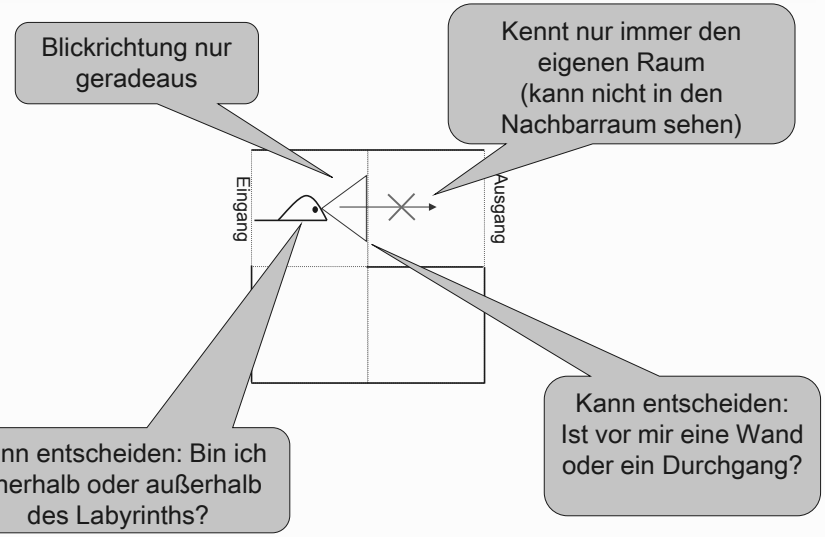




- In beiden Fällen: Der kürzeste Pfad wurde gefunden

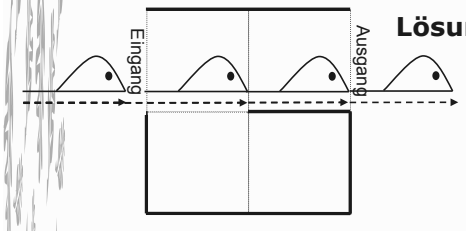
→ Nur möglich, wenn die Maus das gesamte Labyrinth kennt (Blick 'von oben').
→ Dies ist gerade nicht die typische Eigenschaft von Labyrinthproblemen.
→ **Wahrnehmungsmöglichkeiten** der Maus noch nicht betrachtet.

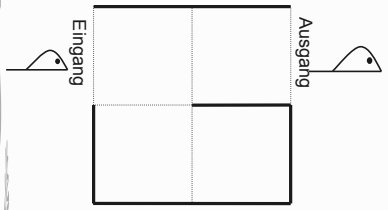
Wahrnehmungsmöglichkeiten der Maus



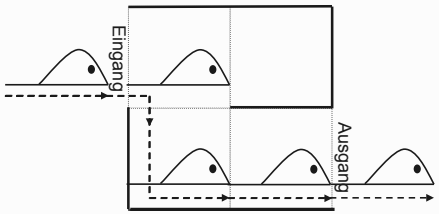
Falls Algorithmus die Lösung 2 ermittelt: Welche Lösung muss dann für das erste Labyrinth ermittelt werden?

Lösung 1: ?

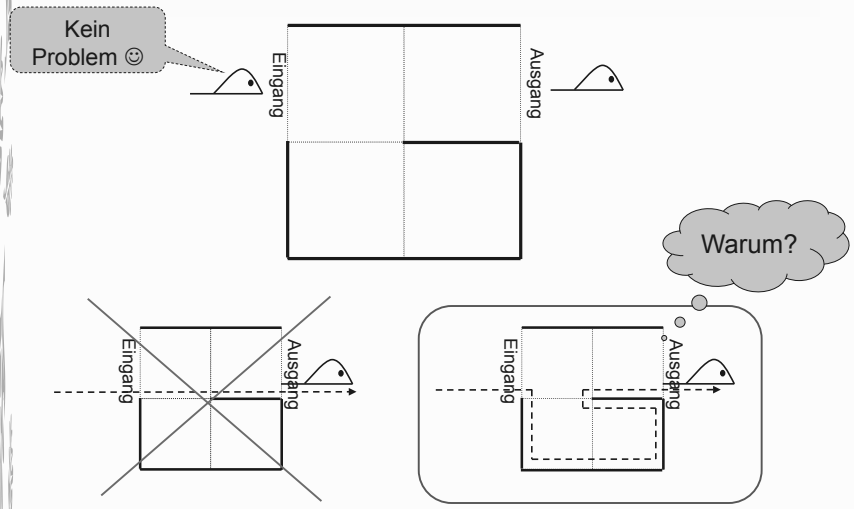




Lösung 2: ok



Falls Lösung 2 ermittelt wird (s.o.): Welche Lösung dann für 1 ermittelt?



Anforderungsspezifikation (2)

2. Die Maus:

Die Maus hat keine Gesamtübersicht des Labyrinths.

a) Die Maus kann sich folgendermaßen bewegen: Linksdrehung, Rechtsdrehung (jeweils um 90 Grad), Schritt vorwärts in den benachbarten Raum.

b) Die Maus befindet sich entweder in einem Raum innerhalb des Labyrinths oder direkt vor dem Eingang oder am Ausgang. Außerdem hat sie eine bestimmte Blickrichtung. Sie kann entscheiden, ob sie sich innerhalb des Labyrinths befindet.

c) Die Maus kann nur in die Blickrichtung sehen. Sie kann entscheiden, ob sich in dieser Richtung eine Wand vor ihr befindet oder nicht.

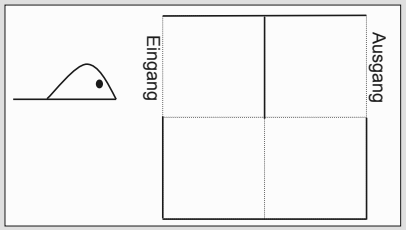
Anforderungspezifikation: offene Probleme?

- Beschreibung des Ausgangszustands der Maus
- Beschreibung des Endzustands der Maus
- Beschreibung der Aufgabenstellung
- Spezialfälle
- Nutzerinterface

Anforderungspezifikation (3)

3. Beschreibung des Ausgangszustands der Maus:
Die Maus befindet sich mit Blickrichtung genau vor dem Eingang (d. h. ein Schritt vorwärts führt bereits in das Labyrinth).

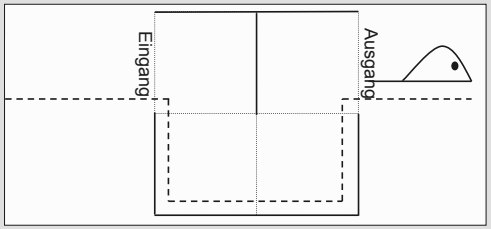
Beispiel:



Anforderungsspezifikation (4)

4. Beschreibung des Zielzustandes bzw. der Aufgabe:
Gesucht ist eine Folge von Bewegungen (drehe nach links, drehe nach rechts, Schritt vorwärts) der Maus, die sie vom Eingang zum Ausgang führt. Dabei sollen allein die Wahrnehmungseigenschaften der Maus im Labyrinth zur Auswahl des jeweils nächsten Schrittes führen.

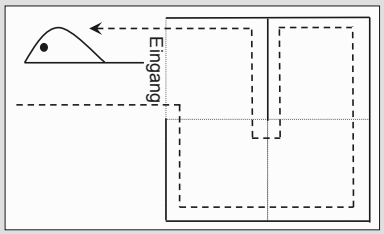
Beispiel:



Anforderungsspezifikation (5)

5. Spezialfall:
Falls es keinen Ausgang gibt oder vom Eingang zum Ausgang keinen Weg, so soll die Maus letztendlich wieder das Labyrinth durch den Eingang verlassen.

Beispiel:



Anforderungsspezifikation (6)

6. Nutzerinterface:
a) Textuelle Ausgabe der Lösung:
Beispiel:
step forward, turn right, … → Vorlesung

b) Graphische Oberfläche:
Die Folge von Bewegungen wird über eine graphische Nutzeroberfläche (GUI) angegeben, d.h. die Bewegungsschritte der Maus im Labyrinth können graphisch verfolgt werden.
Dabei kann der Nutzer interaktiv den jeweils nächsten Schritt auslösen. → Praktikum (früher)

Lösbarkeit des Problems

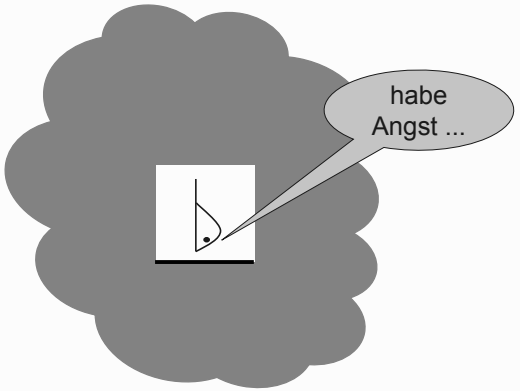
Gibt es einen Algorithmus, der bei den vorgegebenen Eigenschaften der Maus (Wahrnehmungen, Bewegungen) für beliebige Labyrinthe das Problem löst?

- Algorithmenfindung:
  normalerweise erst Teil der Implementation
- Falls Algorithmen entscheidend für Erfolg eines Projekts:
  bereits frühzeitig betrachten
  (Phase: Analyse & Definition)

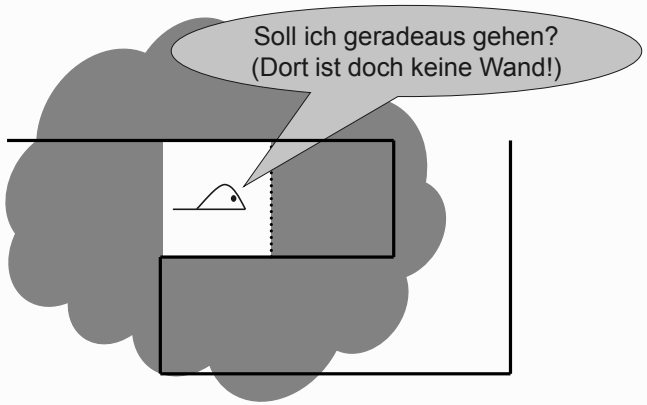
  (z.B. numerische Verfahren)

Betrachtungen zur Lösbarkeit des Problems

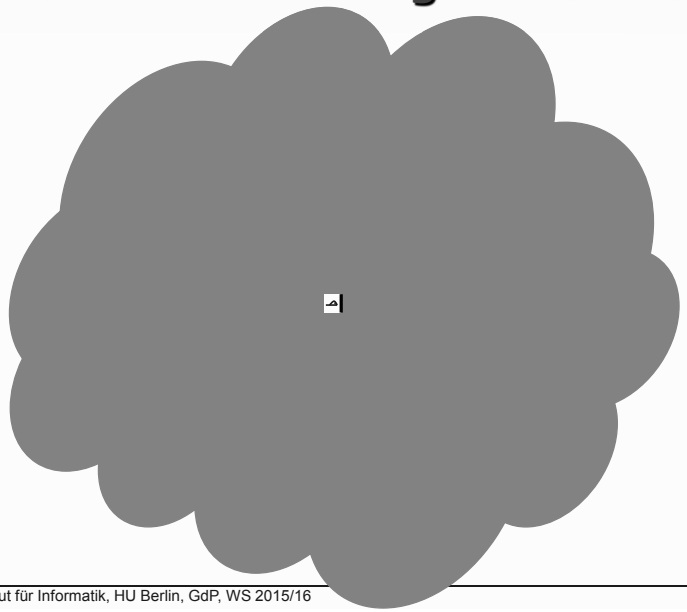
Gibt es einen Algorithmus, der der Maus den Weg weist?



Sichtweise der Maus



Das Problem kann groß sein! ☺



Algorithmus: Grundprinzip?

Prinzip:
Bewege die Maus immer so, dass sich an ihrer rechten Seite eine Wand befindet.

Prinzip schon Algorithmus?

Noch kein Algorithmus:

Kein Verfahren zur schrittweisen Berechnung der Folge von Bewegungen auf der Grundlage von Elementaroperationen.

Elementaroperationen:

Welche?

Algorithmus: Grundprinzip?

Prinzip:
Bewege die Maus immer so, dass sich an ihrer rechten Seite eine Wand befindet.

Prinzip schon Algorithmus?

Noch kein Algorithmus:

Kein Verfahren zur schrittweisen Berechnung der Folge von Bewegungen auf der Grundlage von Elementaroperationen.

Elementaroperationen:

- `step forward`
- `turn left`
- `turn right`
- `facing a wall?`
- `outside the Labyrinth?`

Algorithmus: als Pseudocodeprogramm

Nur ein Schritt – erster Ansatz:

```
falls die Maus noch innerhalb des Labyrinths:
  drehe nach rechts;
  falls (Wand in Blickrichtung): //also: rechts ist Wand
  dann
      drehe nach links;
      falls (Wand in Blickrichtung):
        dann
           drehe nach links und teste ... (Zyklus)
  sonst
      gehe Schritt vorwaerts
```

max. 3-mal

Strenger Pseudocode: ein Schritt

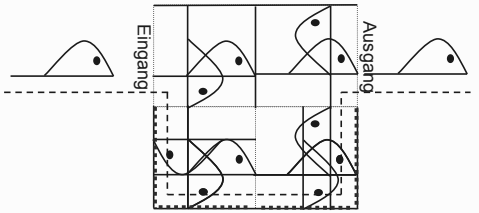
```
IF (NOT outside the Labyrinth?)
  BEGIN /*do next step*/
      turn right;
      WHILE (facing a wall?) DO
            turn left;
      ENDWHILE
      step forward;
  END
```

Unmöglich:

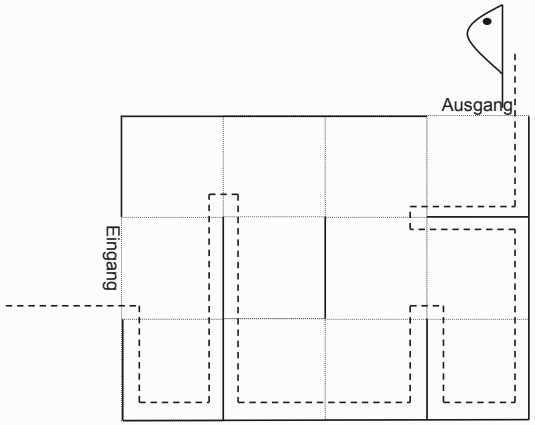
Strenger Pseudocode: gesamter Algorithmus

```
step forward; // vom Eingang
WHILE (NOT outside the Labyrinth?)
  BEGIN // do next step
      turn right;
      WHILE (facing a wall?) DO
            turn left;
      ENDWHILE
      step forward;
  END
ENDWHILE
```


Mausbewegungen dem Algorithmus folgend: erstes Beispiel im Detail

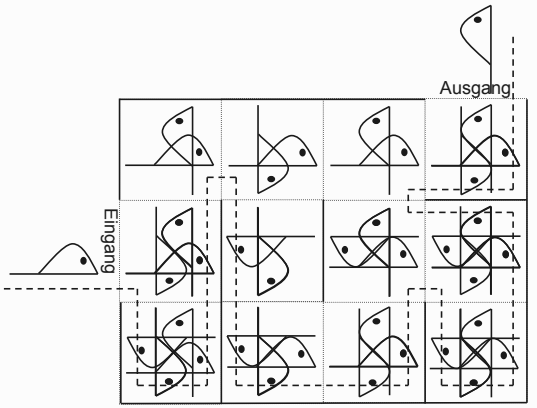


Mausbewegungen dem Algorithmus folgend: zweites Beispiel



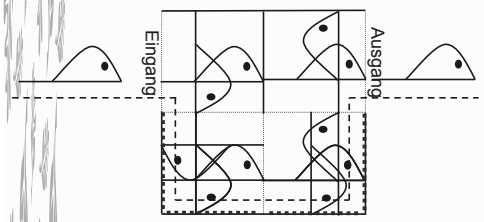
Kürzester Weg: 5 Räume
Algorithmus: alle 12 Räume (3 doppelt)

Mausbewegungen dem Algorithmus folgend: zweites Beispiel im Detail



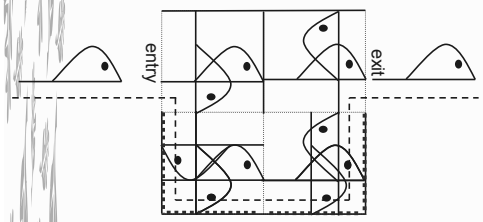
Kürzester Weg: 5 Räume
Algorithmus: alle 12 Räume (3 doppelt)

Mausbewegungen: graphische und textuelle Ausgabe




```
step forward
turn right
step forward
turn right
turn left
turn left
step forward
turn right
turn left
turn left
step forward
turn right
step forward
```

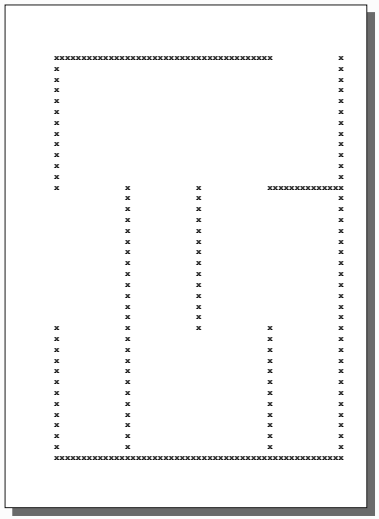

Mouse movements: graphical and textual output




```
step forward; /* enter the maze */
WHILE(NOT outside the maze?)  WHILE
   BEGIN /*do next step*/
         turn right;
         WHILE (facing a wall?) DO
                  turn left;
         ENDWHILE
         step forward;
   END
ENDWHILE
```

```
ENTER   step forward
WHILE   turn right
        step forward
WHILE   turn right
        turn left
        turn left
        step forward
WHILE   turn right
        turn left
        turn left
        step forward
WHILE   turn right
        step forward
```


Textuelle Ausgabe der Lösung



```
step forward
turn to the right
step forward
turn to the right
turn to the left
turn to the left
turn to the left
step forward
…
```

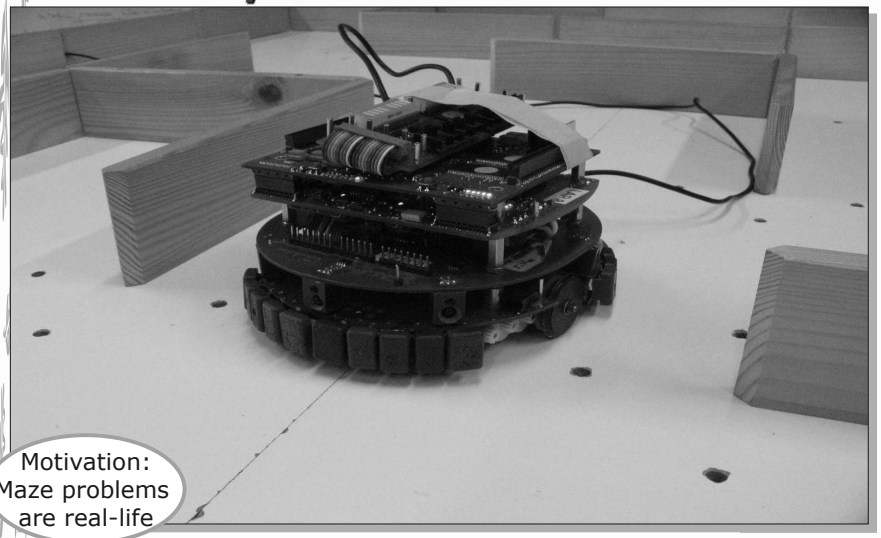

Mäuse im Labyrinth: nur ein nettes Spiel?

- Realistisches Programmierproblem oder nur ein 'Spielzeugprogramm'?

→ Beispielhaftes Problem des Findens von Algorithmen für autonome Roboter.
Roboter mit unterschiedlichen Aufgaben, z.B.

- Fußball spielen
- Roboterbewegung auf dem Mond
- Rettungsroboter

Model of a rescue robot: Mouse replaced by a technical device



Motivation: Maze problems are real-life

Source: Wikipedia

Umsetzung dieses konkreten Problems?

Data representation problem

Another task: Find the house

Algorithm does not work anymore

Glendurgan Garden, Cornwall, England

Irrgarten (Maze) = Labyrinth?

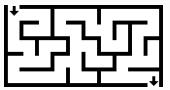
- **Irrgarten: Wege-Puzzle mit vielen Verzweigungen, durch das eine Route gefunden werden muss (Suchproblem)**

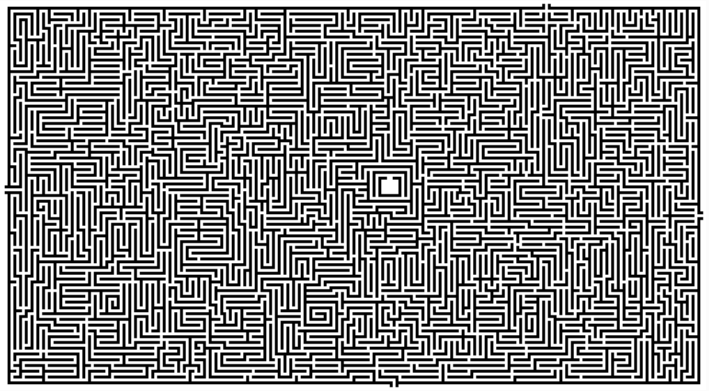
  Ein Irrgarten ist verschieden von einem Labyrinth.

- **Labyrinth: Hat einen eindeutigen Weg zum Zentrum und zurück.**

  Nicht entworfen, um ein Suchproblem zu lösen.

Maze examples (Irrgarten)





Source: Wikipedia

Labyrinth examples (1)

Roman representation of the Minotaur (as a man's head and torso joined to a bull's body) in a labyrinth engraved on an ancient gem

A Roman mosaic showing Theseus and the Minotaur. From Rhaetia, Switzerland

Source: Wikipedia

Labyrinth examples (2)

Labyrinth examples (5)

Classical labyrinth

Medieval labyrinth

Both images come from the 1st or 2nd edition of Nordisk familjebok (1904–1926). The copyrights for that book have expired and these images are in the public domain.

Source: Wikipedia

Resultate der Phase Analyse & Definition: die ganze Wahrheit

Labyrinth-Problem:
→ 6 Seiten Anforderungsspezifikation (ppt)

Praxis:
Vielzahl von Dokumenten entsteht in dieser Phase:

- Aufwandsabschätzung
- Projektplan
- Pflichtenheft (verbales Dokument)
  = unsere 6 Seiten (s.o.) + Testfälle + ...
- Produktmodell (formale Beschreibungskonzepte)
- Benutzeroberfläche
- Benutzerhandbuch

Resultate der Phase Analyse & Definition: die ganze Wahrheit (cont.)

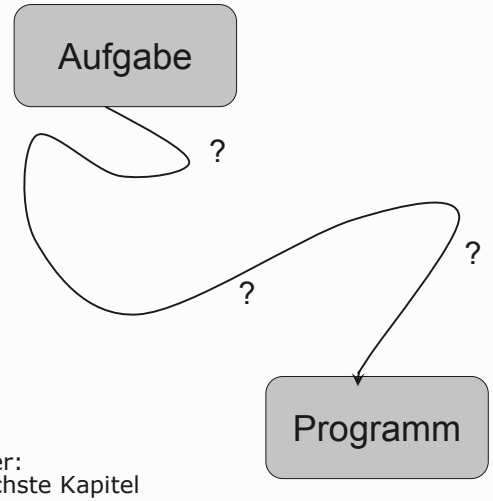
Pflichtenheft (verbales Dokument)
= unsere 6 Seiten (s.o.) + ...

Beispiele:

- Veranstaltungsverwaltungssystem → SemOrg
- Steuerung einer physikalischen Versuchsanlage → XCTL (Manuelle Justage)

3. Semester:
Software Engineering

Vom Problem zum Programm



→ weiter:
nächste Kapitel