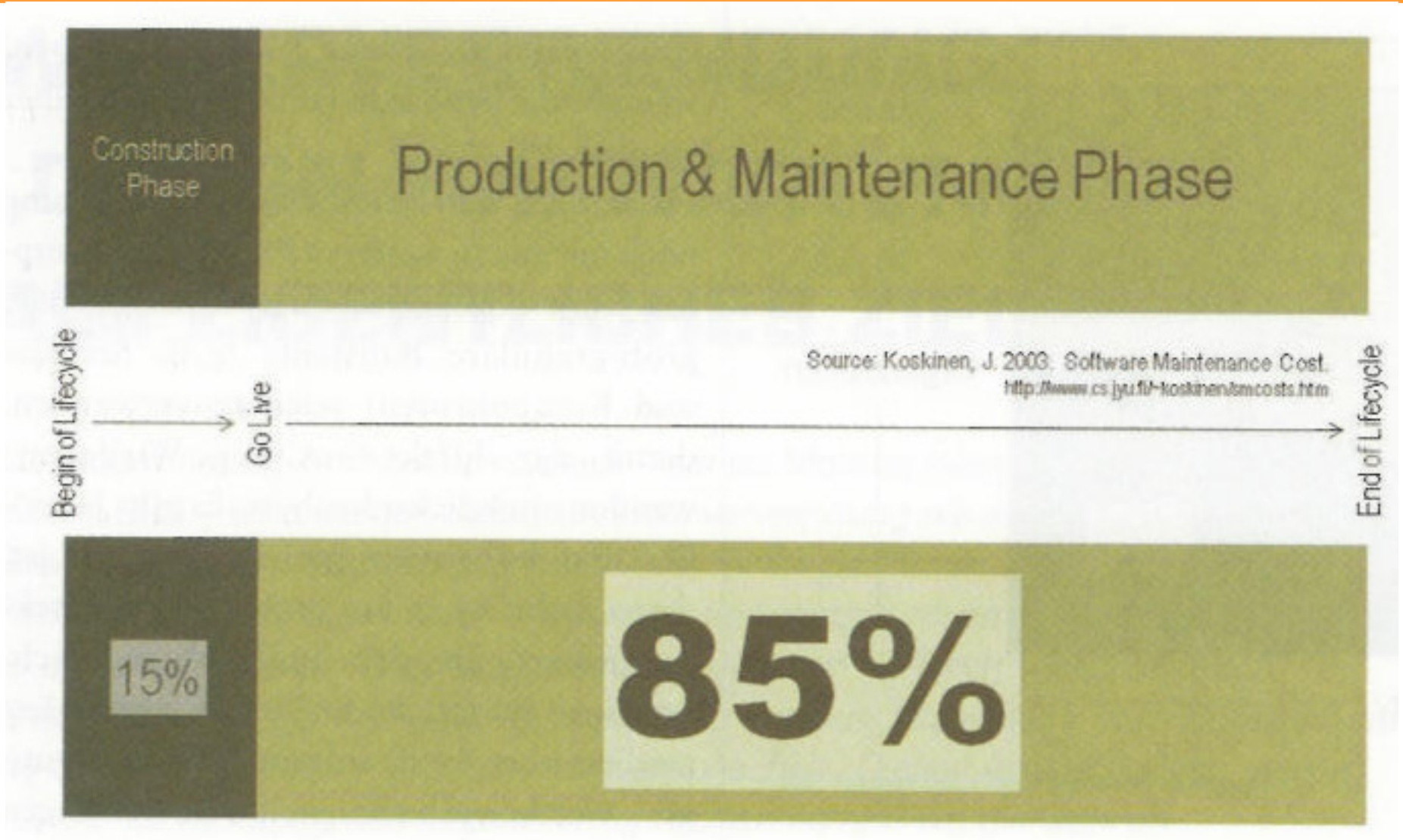
Die Kunst des Programmierens...

Wo die Kosten anfallen



Der Mythos Wiederverwendung: „Design für Wartung“ als eigentliches Ziel,
Objekt Spektrum 4/2009

„**software maintainers** sped 45 percent of their time seeking understanding of a change to be made, 35 percent of their time verifying the change once it is made, and only 20 percent of their time actually making the change [...] they **spend 80 percent of their time thinking**“

Fjelstad, R.K and Hamlen, W.T. Application program maintenance study report to our respondents. Proceedings GUIDE 48, Philadelphia, PA, 1979.

Nach: Robert Glass (2006). Software Creativity 2.0. developer.* books.

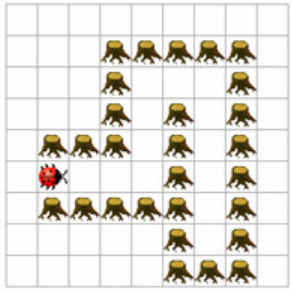
„for every **25 percent** increase in the complexity of a problem to be solved, there is a **100 percent** increase in the complexity of the software reuquired to solve it“

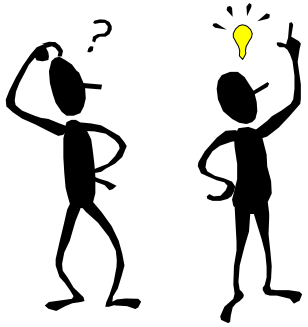
Scott N. Woodfield: An Experiment on Unit Increase in Problem Complexity. IEEE Trans. Software Eng. 5(2): 76-79 (1979)

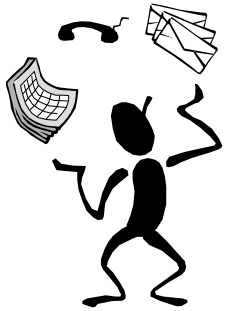
Nach: Robert Glass (2006). Software Creativity 2.0. developer.* books.

Die Kunst des Programmierens...

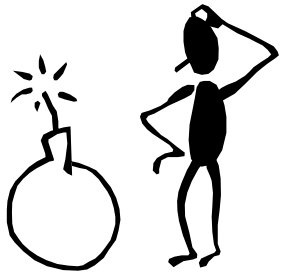
Die Aufgabe **analysieren** **Lösung planen** **und umsetzen!**







Sonst passiert halt...



...was nicht sehr produktiv ist!

The [software] life cycle as universal problem-solving algorithm

1. Identify a problem
2. Define the requirements of the problem.
3. Design a solution to the problem.
4. Implement the design.
5. Test the implementation.
6. Use the implemented product.

None of [these steps] is intrinsically software-specific.

Robert Glass (2006). Software Creativity 2.0. developer.* books.

„essence of creative design“

„People composing design plans performed these steps:

1. Build a mental model of a proposed solution to the problem.
2. Mentally execute the model to see if it does indeed solve the problem. Often this mental execution (also called a „simulation“) takes the form of providing sample input to the model to see if it produces correct sample output.
3. If the sample output is incorrect (as will often be the case in the early stages of design), the model is expanded to correct its deficiencies, then executed again.“

Robert Glass (2006). Software Creativity 2.0. developer.* books.

„4. When the sample output finally becomes correct, another sample input is selected, and steps two and three are repeated.
5. When sufficient sample inputs have passed the test in step four, the model is assumed to be a suitable design model and representation of the design begins.“

„the process is entirely cognitive – at mind speeds“

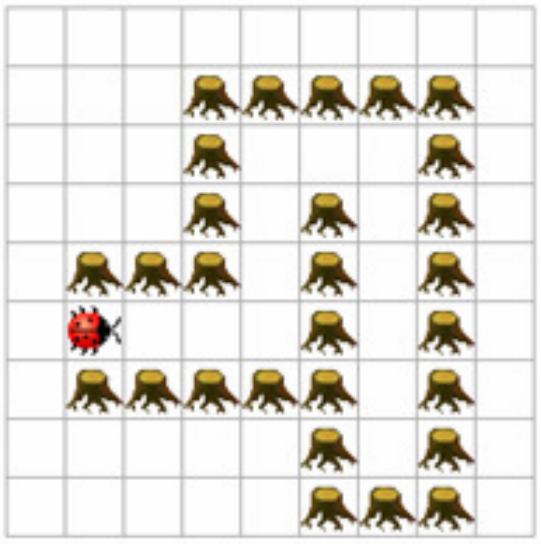
„the process is iterative, a trial-and-error process – heuristic“

Robert Glass (2006). Software Creativity 2.0. developer.* books.

Kara, der Tunnelwächter

Die Aufgabe:

Kara soll den „Tunnel“ in beiden Richtungen endlos ablaufen



Die Analyse:

Kara ist zu jedem Zeitpunkt in genau einer dieser fünf Situationen:

Nur vor ihm ist kein Baum

Nur links von ihm ist kein Baum

Nur rechts von ihm ist kein Baum

Links und rechts kein Baum

Links, rechts und vorne Bäume

⇒ braucht drei Sensoren:

Kara, der Tunnelwächter

Die Analyse:

Kara ist zu jedem Zeitpunkt in genau einer dieser 5 Situationen:

Nur vor ihm ist kein Baum 

Nur links von ihm ist kein Baum 

Nur rechts von ihm ist kein Baum 

Links, rechts und vorne Bäume 

Links und rechts kein Baum 

Das Programm:

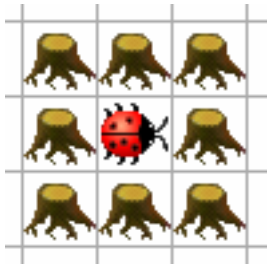
Kara muss nur diese 5 (von allen möglichen 8) Situationen berücksichtigen:

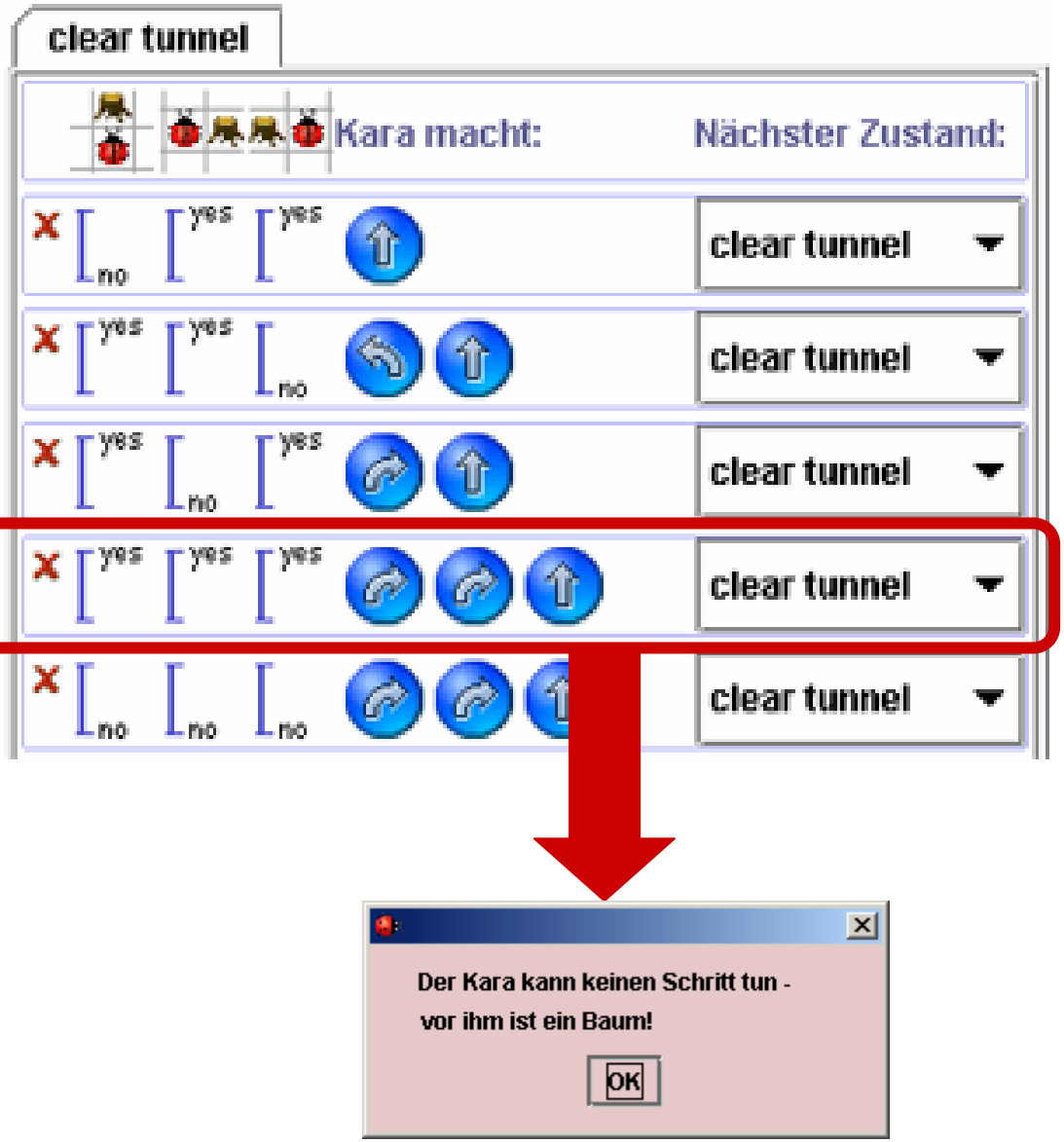
Dies ist nur eine von allen möglichen Lösungen!

Tunnelwächter - Programmtest

Wo ist das Problem mit dem Programm?

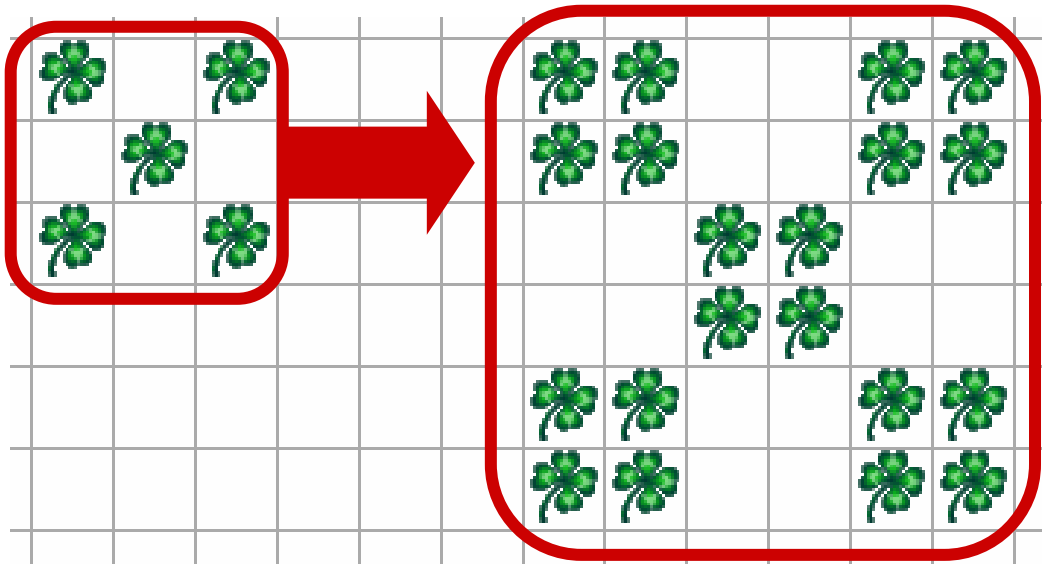
Ist das noch ein „Tunnel“?





Kara, der Bildbearbeiter

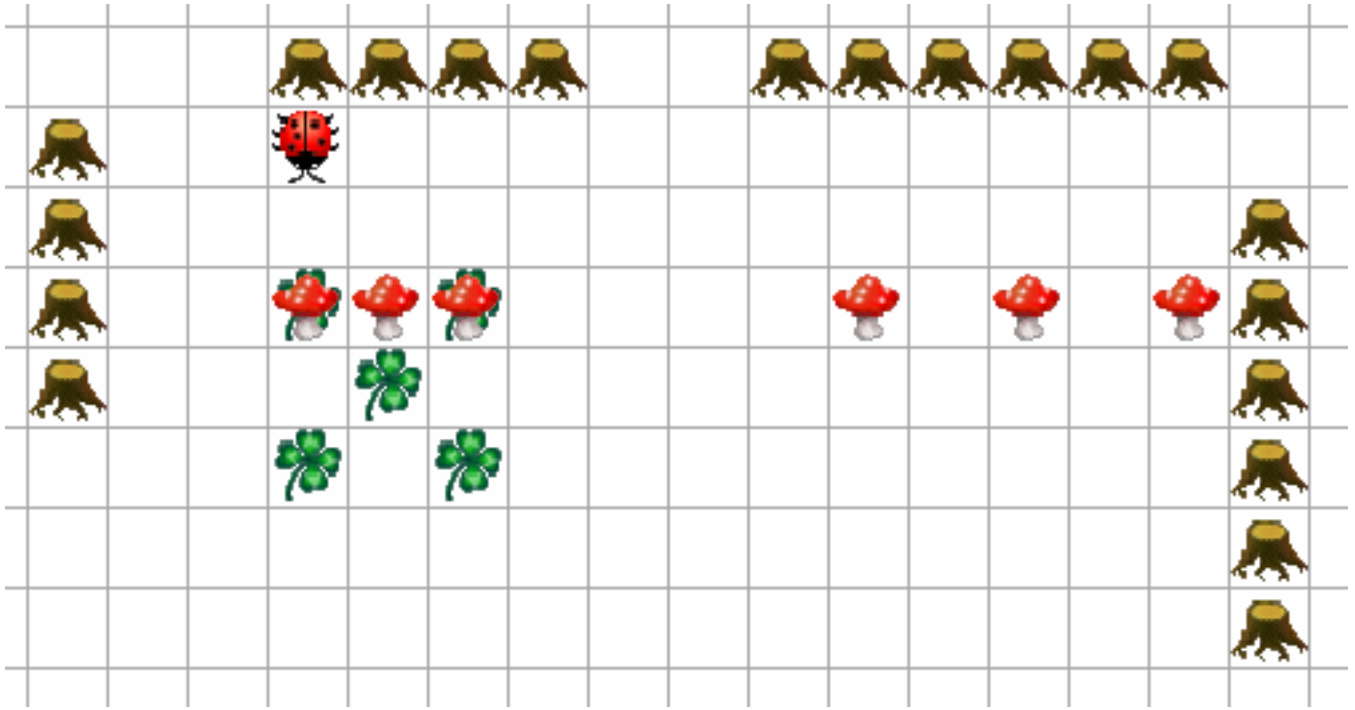
Die Aufgabe:
Kara soll das Kleeblatt-Bild um den Faktor 2 vergrössern.



1. Ausgangssituations analysieren
2. Aufgabe in einzelne Teilaufgaben unterteilen
3. Teilaufgaben lösen
4. Teilaufgaben zu ganzer Lösung zusammensetzen

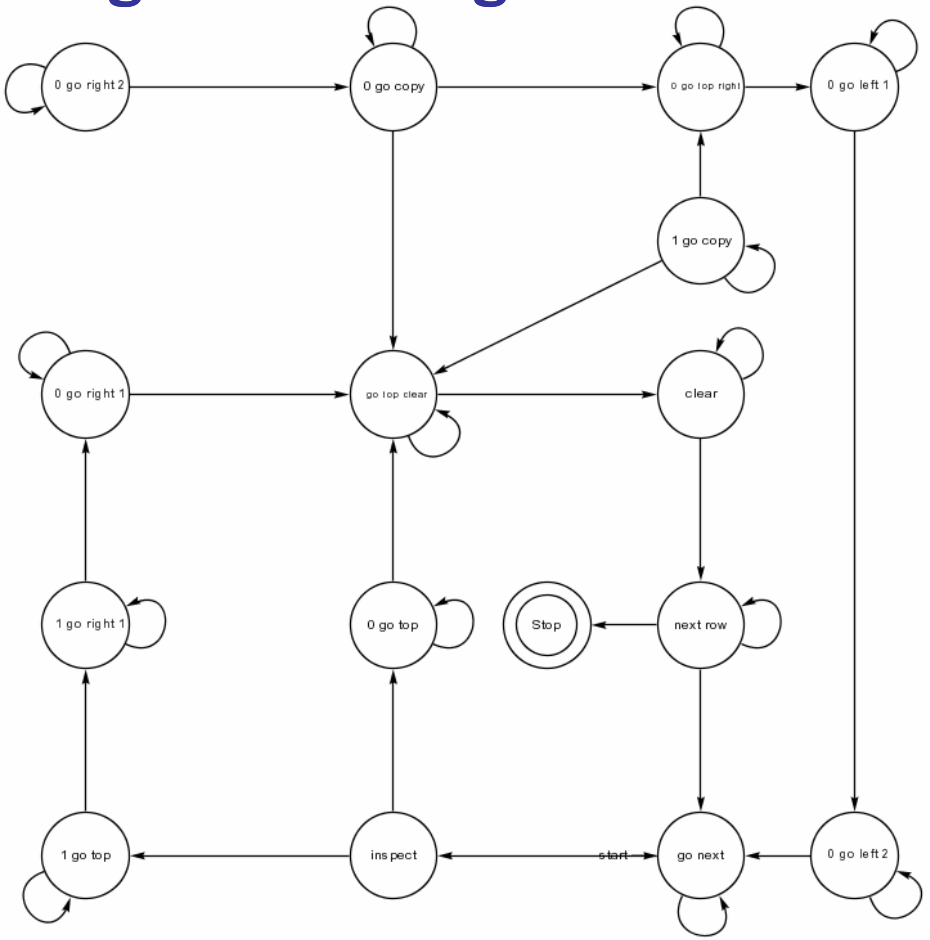
Kara, der Bildbearbeiter

1. **Ausgangssituations analysieren**
2. Aufgabe in einzelne Teilaufgaben unterteilen
3. Teilaufgaben lösen
4. Teilaufgaben zu ganzer Lösung zusammensetzen



Kara, der Bildbearbeiter

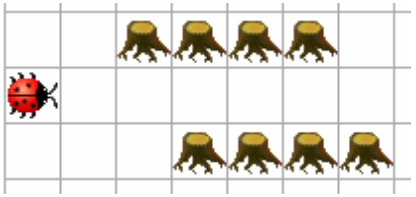
1. Ausgangssituations analysieren
2. Aufgabe in einzelne Teilaufgaben unterteilen
3. Teilaufgaben lösen
4. Teilaufgaben zu ganzer Lösung zusammensetzen



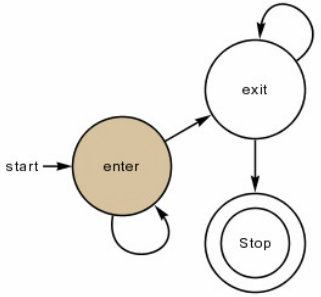
Wozu sind Zustände gut ?

- manchmal eine Frage des **Programmierstils**

- als **Gedächtnis**: was hat Kara schon erledigt ?
  Beispiel: Aufgabe „suche Tunnelausgang“



1. „enter“: suche Tunneleingang
   (links **und** rechts eine Wand)



2. „exit“: suche Tunnelausgang
   (links **oder** rechts **keine** Wand)

anderes Verhalten bei gleichen Situationen in den beiden Zuständen!