EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

MATHEMATISCH-
NATURWISSENSCHAFTLICHE FAKULTÄT
FB Informatik, Programmiersprachen und Softwaretechnik

DENKEN VERSTEHEN LERNEN
Computational Thinking in der Grundschule

Grundschulpraktikum (B.Ed. und B.Sc.)

26./27.10.2016

Problemlösung

Rezepte und Tipps

Der Problemlösungs-Prozess

- In diesem Curriculum wird die Lösung von Problemen anhand der Entwicklung von Algorithmen, die eine bestimmte Aufgabe erfüllen, geübt.
- Ein einem Algorithmus ähnelndes, schrittweises Vorgehen lässt sich auch auf den Prozess der Problemlösung selbst anwenden.
- Der Mathematiker George Pólya („Schule des Denkens“) entwickelte eine derartige Anleitung für die Lösung von *mathematischen* Problemen, bestehend aus 4 Schritten.
- Daran orientiert, bietet code.org sein eigenes Problemlösungs-Rezept an.

Pólya's 4 Schritte

1. Aufgabe verstehen/analysieren
2. Lösungsstrategie (Plan) erarbeiten
3. Plan ausführen
4. Ergebnis überprüfen/weiterdenken

Pólya's Schritte im Detail

1. Aufgabe verstehen/analysieren:

- ☑ Durch die Aufgabe gegebene Daten analysieren:
  - Was ist gesucht?
  - Was ist gegeben?
  - Was ist die **Voraussetzung**?
  - Was ist die **Behauptung**?
- ☑ Lösbarkeit der Aufgabe überprüfen:
  - Voraussetzung erfüllbar?
  - Kann die Aufgabe (etwa Bestimmung einer Unbekannten) anhand der Voraussetzung gelöst werden?
  - Ist die Aufgabe (so wie sie gestellt ist) sinnvoll?

Pólya's Schritte im Detail

1. Aufgabe verstehen/analysieren (Fortsetzung):

- ☑ Aufgabe veranschaulichen:
  - Skizze anfertigen
  - passende Bezeichnungen einführen
- ☑ Teile der Aufgabe analysieren:
  - Voraussetzung und Behauptung in verschiedene Teile trennen
  - einzelne Teile aufschreiben/umformulieren

Pólya's Schritte im Detail

2. Lösungsstrategie (Plan) erarbeiten

Ziel: Den Zusammenhang zwischen den Daten und dem Gesuchten finden, um so einen Plan zur Lösung der Aufgabe zu erhalten.

- ☑ Aufgabe so oder so ähnlich bereits bekannt? Muster suchen
- ☑ Existiert ein für die Aufgabe hilfreicher math. Satz (oder eine verwandte Aufgabe)?
- ☑ Unbekannte/Behauptung betrachten:
  - Existiert Aufgabe mit ähnlicher Unbekannten/Behauptung?
  - Möglich, deren Ergebnis oder Methode zu verwenden?
  - Evtl. Hilfselement einführen, um deren Ergebnis verwenden zu können.

Pólya's Schritte im Detail

2. Lösungsstrategie (Plan) erarbeiten (Fortsetzung)

- ☑ Evtl. Behauptung umformulieren, zurückgehend auf die Definitionen
- ☑ Eine leichter zu lösende verwandte Aufgabe suchen und lösen:
  - Voraussetzung oder Behauptung abändern
  - Z.B. verallgemeinern, spezialisieren, Analog finden
  - oder: Teil der ursprünglichen Aufgabe lösen
- ☑ Überprüfen, ob alle Daten/die ganze Voraussetzung/alle vorkommenden Begriffe verwendet werden

Pólya's Schritte im Detail

3. Plan ausführen:

☑ Bei jedem einzelnen Schritt während der Ausführung überprüfen:

- Ist der Schritt richtig?
- Richtigkeit beweisen

Pólya's Schritte im Detail

4. Ergebnis überprüfen/weiterdenken:

- ☑ Ergebnis bzw. Beweis überprüfen, z.B. anhand von Beispielen testen (Zahlen einsetzen, …)
- ☑ Gibt es andere Wege zum selben Resultat?
- ☑ Ist das Resultat für andere Aufgaben wiederverwendbar?
  - Welches neue Wissen erhalten wir dadurch?
  - Ergebnis oder Methode für später merken?
  - Führt dies zu neuen mathematischen Aufgaben?
  - Vergleich mit verwandten Aufgaben: Umkehrung, Verallgemeinerungen

Pólya's Schritte im Detail

Für jeden Schritt gilt: Die folgenden Fragen bedenken:

- ☑ Wo soll ich anfangen?
- ☑ Was kann ich tun?
- ☑ Was kann ich erreichen?

Pólya's Schritte im Detail

Nach Karin Halupczok:
http://wwwmath.uni-muenster.de/u/karin.halupczok/vorkurs2013/PolyaTabelle.pdf

Pólya's Schritte im Detail

Beispielaufgabe:

Zweimal die Differenz zwischen einer Zahl und 1 ist um 4 größer als diese Zahl. Welche Zahl ist gesucht?

(http://www.wtamu.edu/academic/anns/mps/math/mathlab/int_algebra/int_alg_tut8_probsol.htm)

Pólya's Schritte im Detail

Beispielaufgabe:

1. Schritt: Gesucht ist eine Zahl, nennen wir sie *x*.
   Gegeben ist eine Gleichung für *x*.
2. Schritt: Plan: Formalisiere die Gleichung (A) und löse nach *x* (B).
3. Schritt: A) Gleichung formalisiert: 2(x-1) = x+4
   B) Lösen nach *x*:
   2(x-1) = x+4 | Ausmultiplizieren (linke Seite)
   2x - 2 = x+4 | alle x auf eine Seite bringen
   x - 2 = 4 | beide Seiten +2
   x = 6
4. Schritt: Überprüfen durch Einsetzen in die Gleichung für *x*
   2(6-1) = 10, 6+4 = 10

Pólya's Schritte im Detail

bekanntes allgemeines Vorgehen: Einsatz von Gleichungen

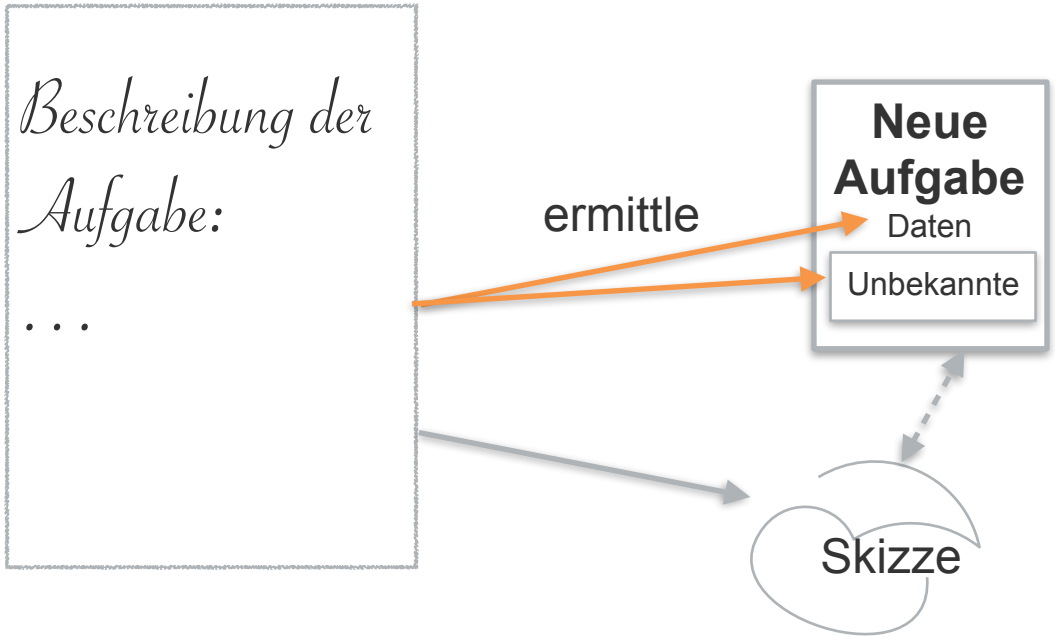
Beispielaufgabe:

1. Schritt: Gesucht ist eine Zahl, nennen wir sie *x*.
   Gegeben ist eine Gleichung für *x*.
2. Schritt: Plan: Formalisiere die Gleichung (A) und löse nach *x* (B).
3. Schritt: A) Gleichung formalisiert: 2(x-1) = x+4
   B) Lösen nach *x*:
   2(x-1) = x+4 | Ausmultiplizieren (linke Seite)
   2x - 2 = x+4 | alle x auf eine Seite bringen
   x - 2 = 4 | beide Seiten +2
   x = 6
4. Schritt: Überprüfen durch Einsetzen in die Gleichung für *x*
   2(6-1) = 10, 6+4 = 10

Pólya's Schritte im Detail

Eine kleine Visualisierung (erfasst nicht alle Aspekte):

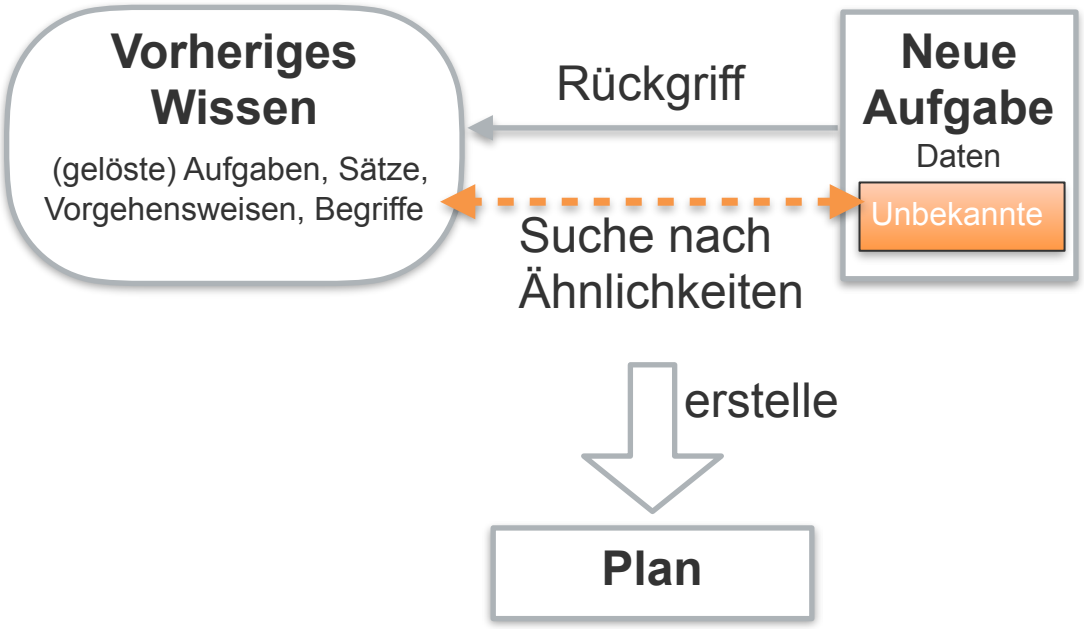
1. Schritt:



Pólya's Schritte im Detail

Visualisierung (Fortsetzung):

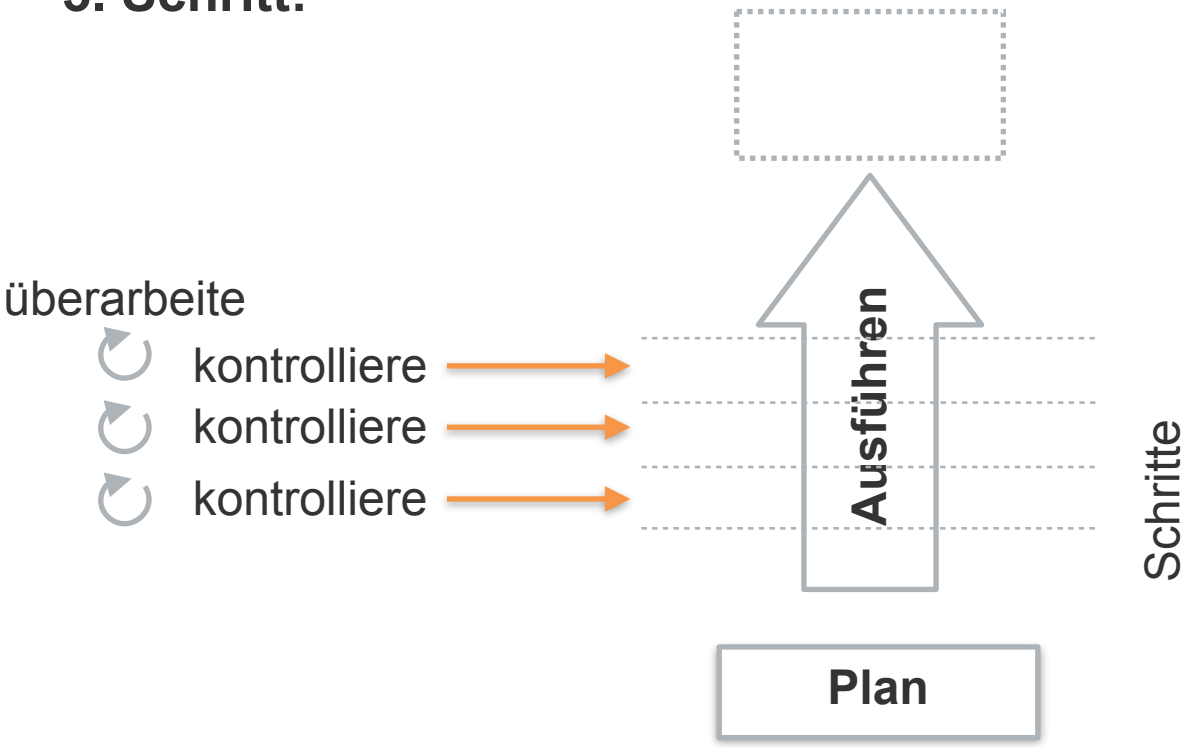
2. Schritt:



Pólya's Schritte im Detail

Visualisierung (Fortsetzung):

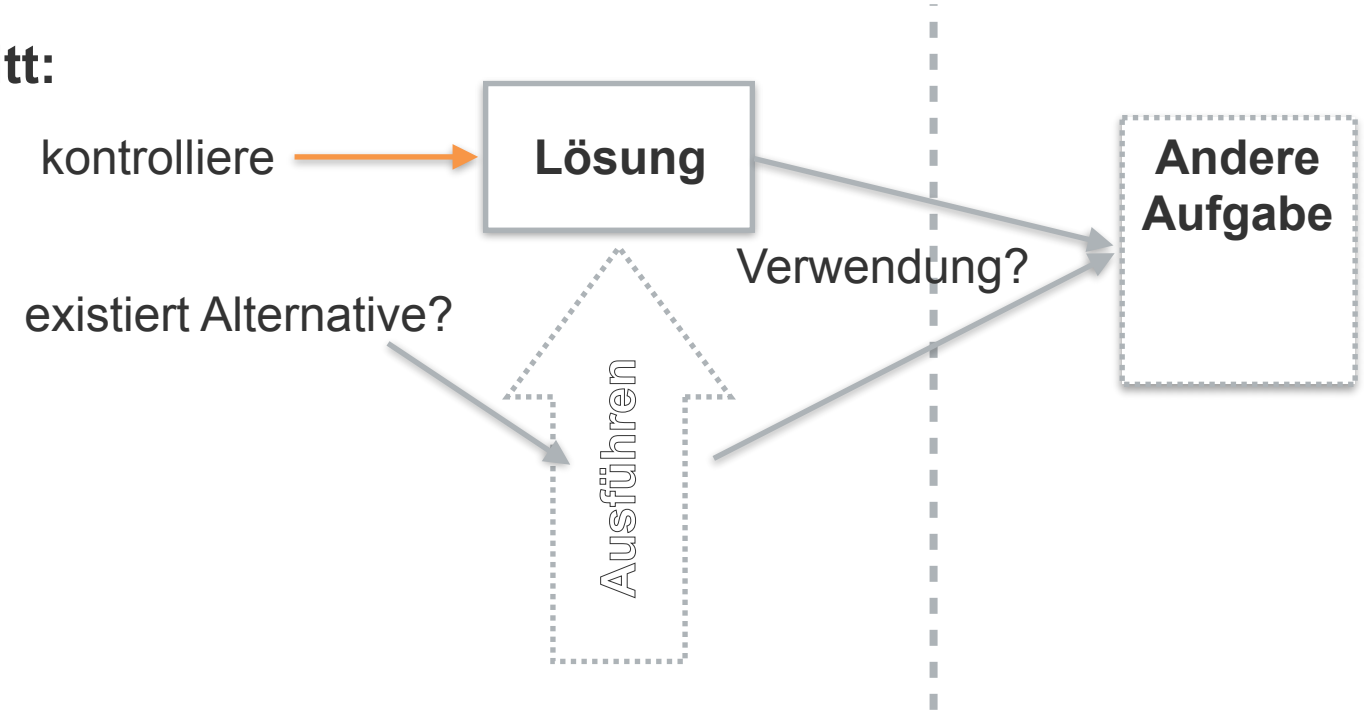
3. Schritt:



Pólya's Schritte im Detail

Visualisierung (Fortsetzung):

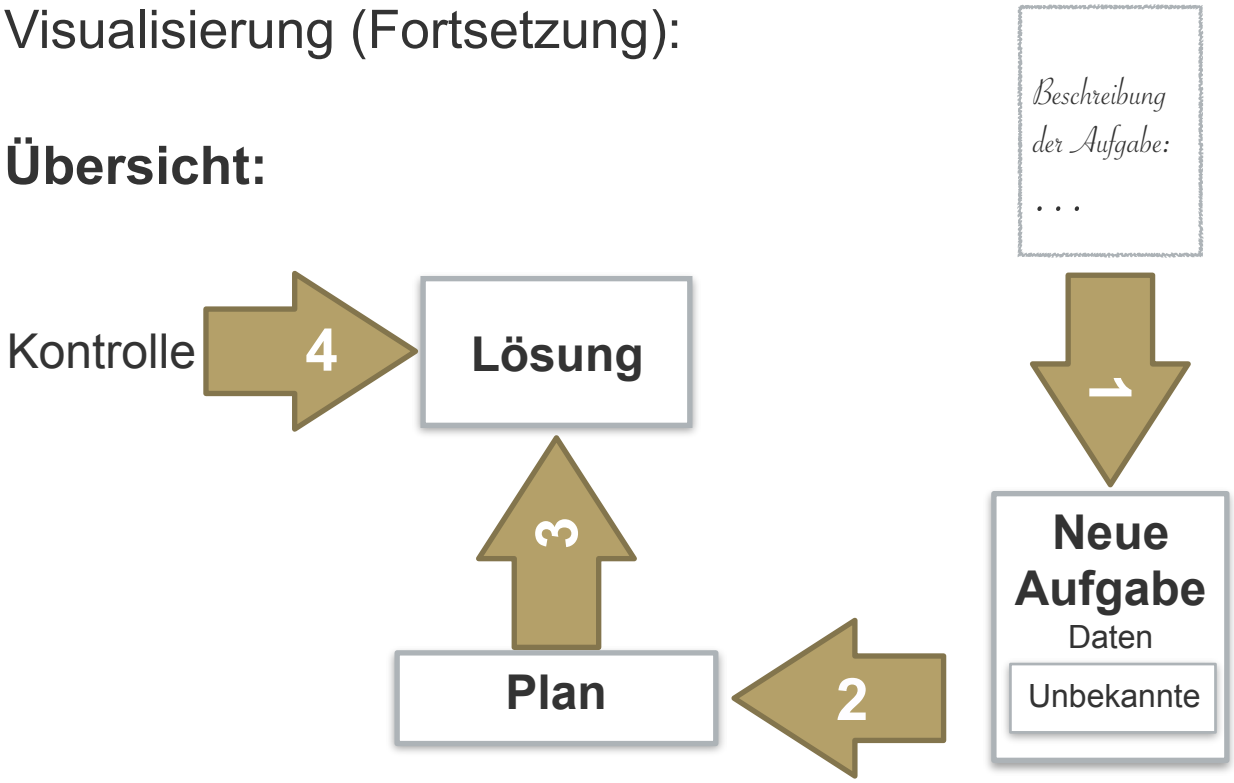
4. Schritt:



Pólya's Schritte im Detail

Visualisierung (Fortsetzung):

Übersicht:

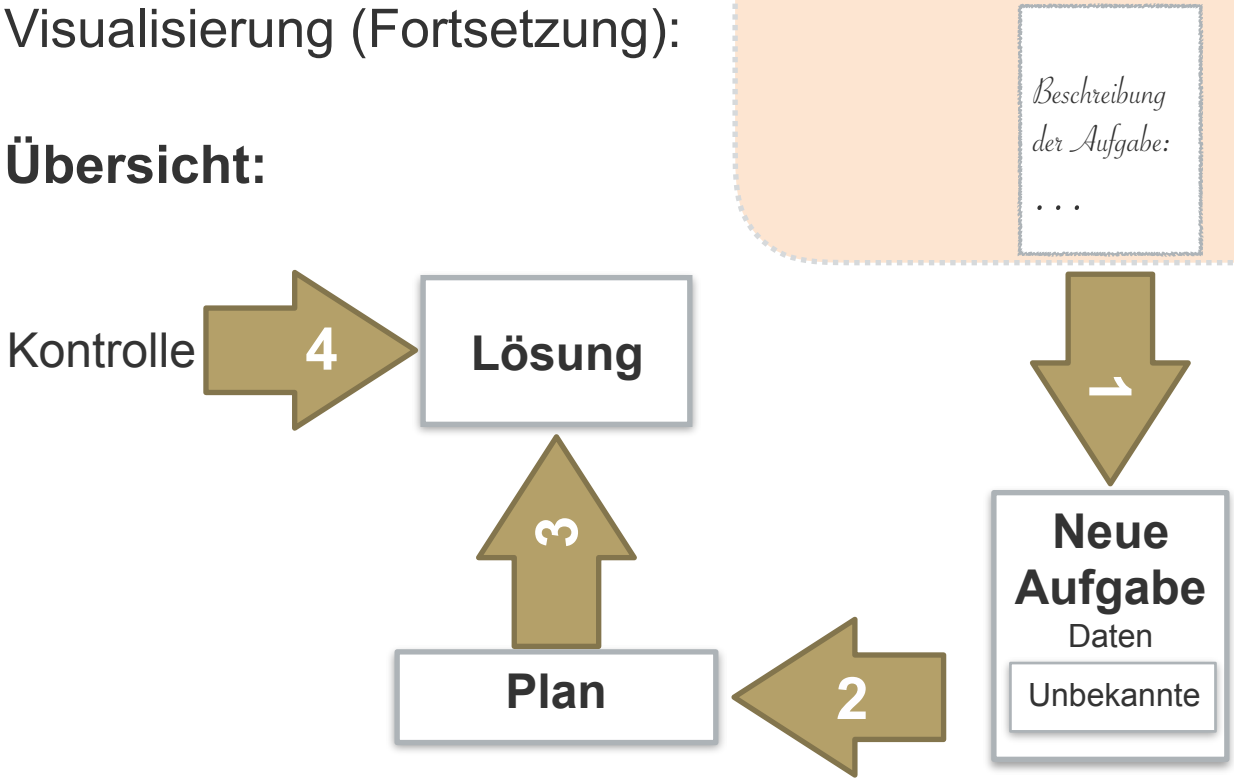


Pólya's Schritte im Detail

Problemraum

Visualisierung (Fortsetzung):

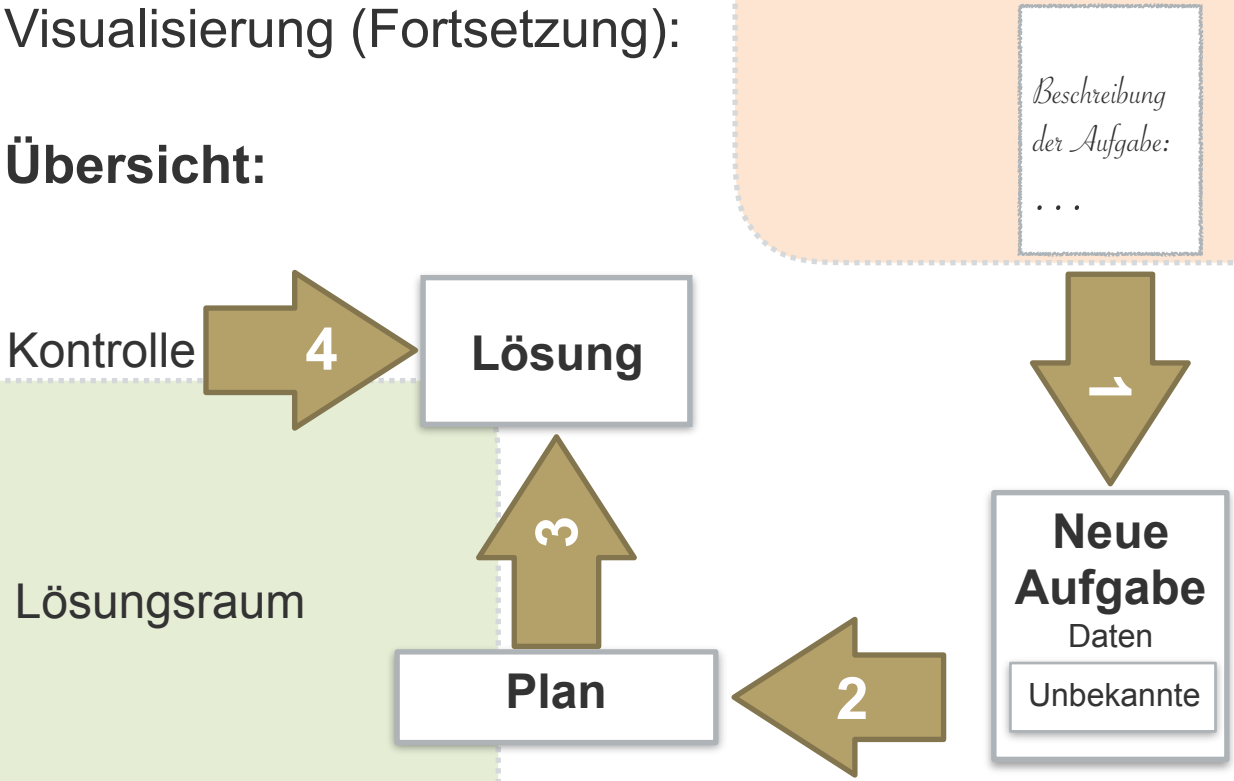
Übersicht:



Pólya's Schritte im Detail

Visualisierung (Fortsetzung):

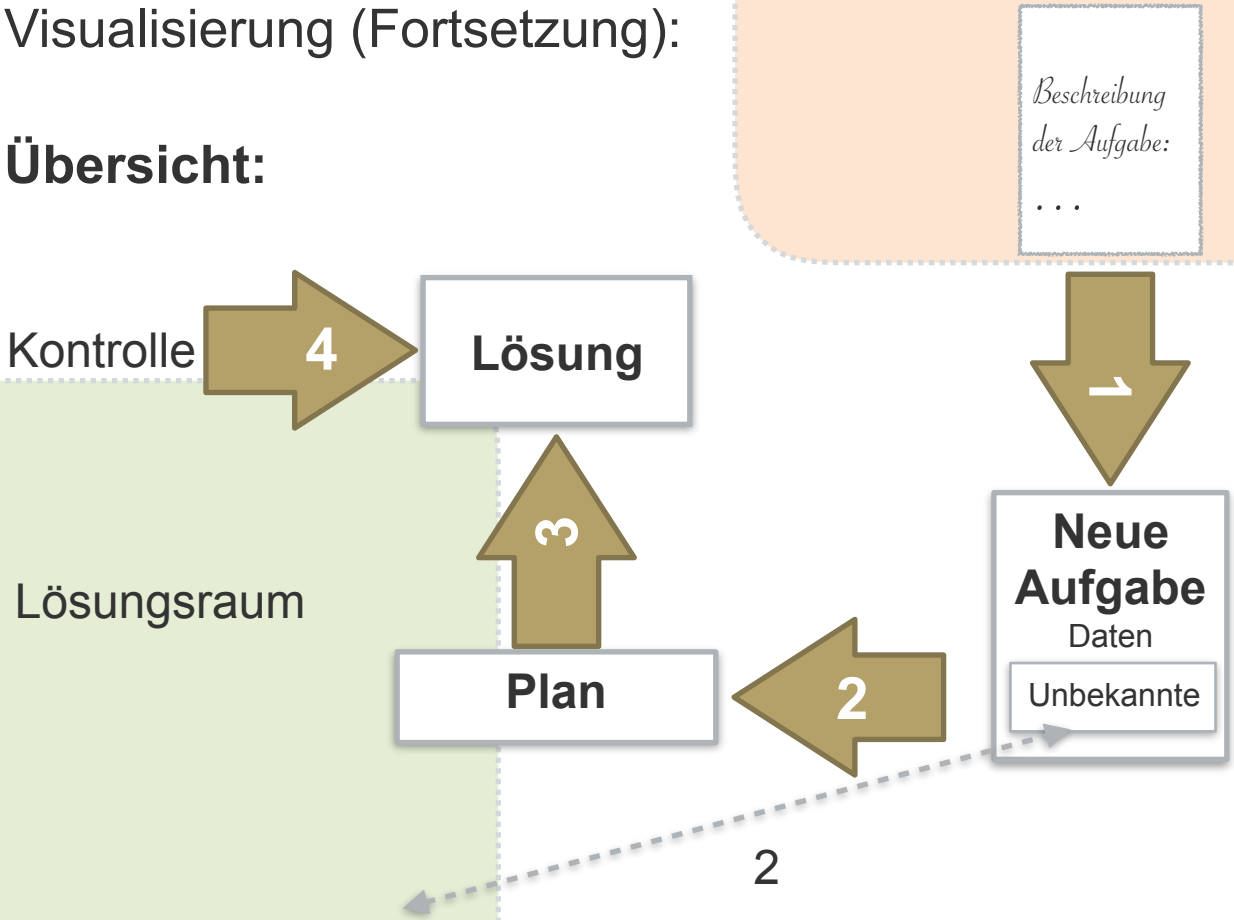
Übersicht:



Pólya's Schritte im Detail

Visualisierung (Fortsetzung):

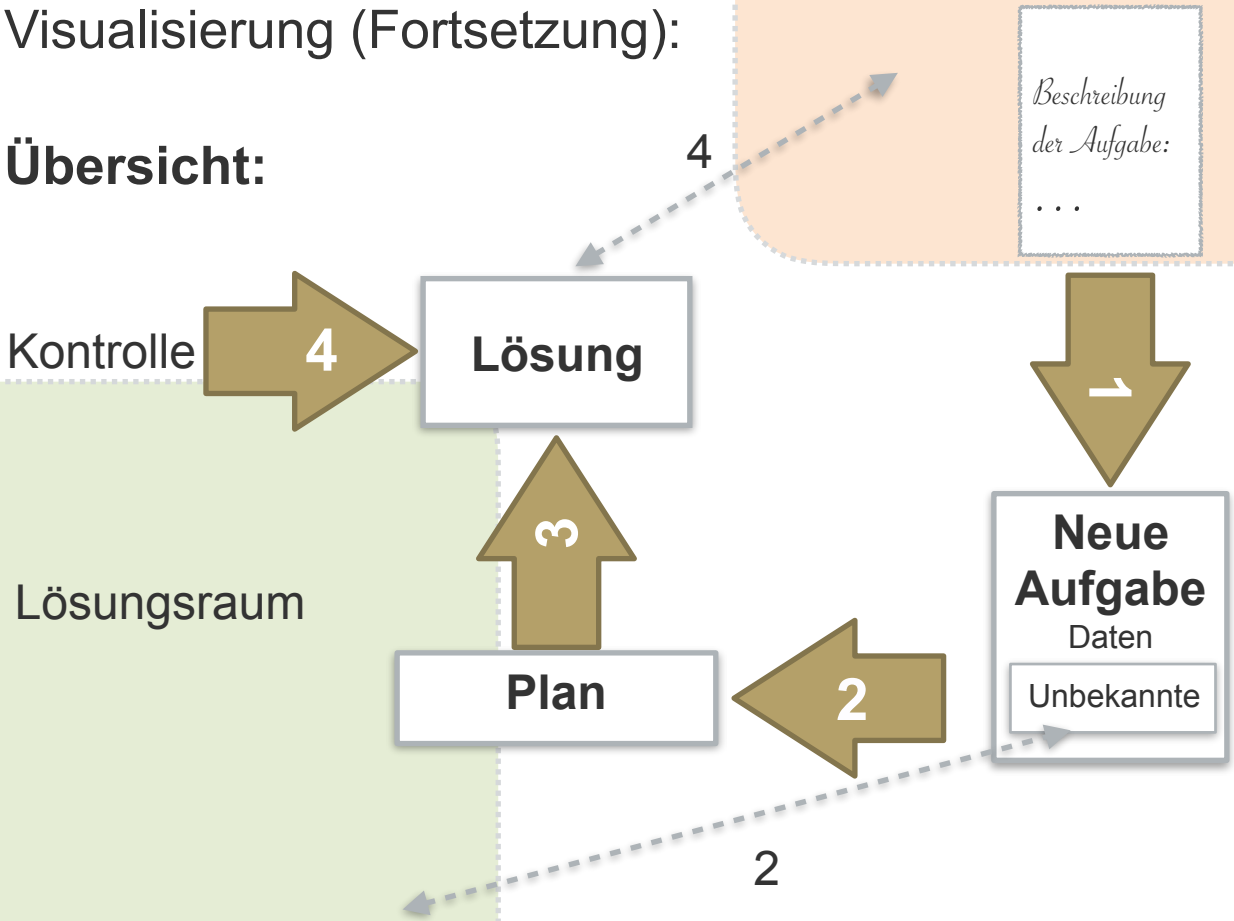
Übersicht:



Pólya's Schritte im Detail

Visualisierung (Fortsetzung):

Übersicht:



Code.org Rezept

Orientiert an Pólya's 4 Schritten wurde von den Machern von code.org ein Rezept zur Lösung ihrer algorithmischen Aufgaben entwickelt:

1. Aufgabe (Rätsel) verstehen
2. Plan zusammenstellen
3. Plan ausführen und verbessern
4. Ergebnis prüfen

4 Schritte von Code.org

1. Aufgabe (Rätsel) verstehen

- ☑ Was verlangt die Aufgabe (das Rätsel)?
- ☑ Das Problem in eigenen Worten beschreiben
- ☑ Ist bereits Code vorgegeben?
  - Was tut der Code?
  - Wieso ist der Code da?
- ☑ Was ist das Ziel der Aufgabe (des Rätsels)?
- ☑ Gibt es andere (bereits gelöste) Aufgaben/Rätsel wie dieses?

4 Schritte von Code.org

2. Plan zusammenstellen

Stelle einen Plan aus einigen dieser Elemente zusammen:

- ☑ Algorithmus schreiben
- ☑ Etwas raten und später überprüfen
- ☑ Ein Bild von dem, was du tun willst, malen
- ☑ Rückwärts arbeiten (ausgehend von dem, was du bereits weißt)
- ☑ Immer nur ein kleines Teil auf einmal lösen
- ☑ Mit einem bereits gelösten Rätsel vergleichen

4 Schritte von Code.org

3. Plan ausführen und verbessern

- ☑ Ist die Aufgabe (das Rätsel) gelöst?
- ☑ Falls nein: Probleme beheben, immer nur eines gleichzeitig
- ☑ Plan nach jeder Änderung erneut testen
- ☑ Frustriert? Tief durchatmen oder eine Minute von der Aufgabe abwenden — danach fällt es vllt. leichter zu erkennen, was Schwierigkeiten macht.
- ☑ Fragen stellen: Vllt. kann jemand anderes helfen, herauszufinden wo der Plan fehlschlägt

4 Schritte von Code.org

4. Ergebnis prüfen

- ☑ Löst die Antwort die Aufgabe (das Rätsel)?
- ☑ Sind alle vorgegebenen Ziele erfüllt?
- ☑ Gibt es noch einen einfacheren Weg, die Aufgabe zu lösen?
- ☑ Versuchen, Lösung leicht abändern, so dass sie für andere Aufgaben funktioniert
- ☑ Versuchen, die Lösung zu erklären, auch um anderen zu helfen

Offene Diskussion & Fragen

Vorschläge für den Unterricht

- Teilen Sie den SchülerInnen das Problemlösungs-Rezept aus und sprechen Sie mit Ihnen darüber.
- Wenn SchülerInnen um Hilfe fragen, können Sie ihnen die folgenden Suggestivfragen stellen.

Vorschläge für den Unterricht

Schritt 1: Aufgabe (Rätsel) verstehen):

? Verstehst du die Situation (was das Rätsel verlangt)?
? Kannst du das Problem in eigenen Worten beschreiben?
? Verstehst du den vorgegebenen Code und warum es vorgegeben ist? Was für eine Rolle spielt der Code?
? Weißt du, was das Ziel des Rätsels ist?
? Ist dieses Problem ähnlich wie ein anderes, das du schon gelöst hast?

Vorschläge für den Unterricht

Schritt 2: Plan zusammenstellen:

? Können eine oder mehrere der folgenden Strategien eingesetzt werden?
- Raten und überprüfen
- Eine Karte zeichnen
- Ein Bild malen
- Nach einem Muster suchen
- Mit einem vorher gelösten Rätsel vergleichen
- Ein einfacheres Problem lösen

? Wie wäre es damit, ein Diagramm zu zeichnen?

? …, ein gleichbedeutendes Problem zu lösen?

? …, Teilaufgaben zu identifizieren?

? …, rückwärts von der Aufgabenstellung aus zu arbeiten?

Vorschläge für den Unterricht

Schritt 3: Plan ausführen und verbessern:

? Hast du versucht, das Rätsel mithilfe deines Plans zu lösen?

? Wenn dein Plan fehlschlägt: Hast du dir die Rückmeldung bei den Fehlern angesehen? Modifiziere deinen Plan entsprechend.

? Hast du deine Strategie ausprobiert und wenn nötig geändert? Mache das oft und habe keine Angst davor, Lösungen auszuprobieren bevor du weiß dass sie perfekt sind.

? Hängst du am Problem fest? Mache für einen Moment etwas anderes (laufe ein wenig herum, schau weg vom Bildschirm/ Arbeitsblatt,…). Denke an etwas anderes. Vllt. fällt dir die Lösung danach ein.

? Hast du schon mit anderen über das Problem gesprochen? Jemand hat vllt. einen Tipp oder kann dir sogar etwas erklären.

Vorschläge für den Unterricht

Schritt 4: Ergebnis prüfen:

? Stimmt deine Lösung? Erfüllt deine Antwort alle Ziele? (z.B. Anzahl Blöcke, Verwendung eines bestimmten Blocks, …)

? Siehst du eine einfachere oder effizientere Lösung?

? Kannst du deine Lösung erweitern, so dass sie einem allgemeineren Muster folgt?

Offene Diskussion & Fragen

Debugging-Tipps

Zusätzlich zu den 4 Problemlösungsschritten gibt es von code.org noch einige Tipps zum Debugging von Programmen und zur Vermeidung von Fehlern (die beste Form des Debuggings :))

Debugging-Tipps

Tipps zur Fehlervermeidung:

- ☑ Anweisungen genau lesen.
- ☑ Das Ziel des Rätsels herausfinden.
- ☑ Langsam und Schritt für Schritt arbeiten.
- ☑ Versuchen, das Problem in eigenen Worten zu beschreiben.
- ☑ Wenn bereits Code vorgegeben wurde, herausfinden:
  - Was macht der Code?
  - Warum ist der Code wohl da?

Debugging-Tipps

Tipps zum eigentlichen Debugging:

- ☑ Bei jedem Schritt des Lösungsweges nach Problemen Ausschau halten.
- ☑ Beschreiben, was hätte passieren sollen.
- ☑ Beschreiben, was schiefgeht.
- ☑ Versuchen, aus dem Unterschied zwischen dem, was passieren sollte, und dem, was tatsächlich geschah, einen Hinweis zur Lösung des Problems zu finden.
- ☑ Immer nur eine Sache auf einmal beheben und beschreiben wie sich das Ergebnis verändert hat.

Debugging-Tipps

Tipps zum eigentlichen Debugging (Fortsetzung):

- ☑ Versuchen, „Brotkrümel“ (breadcrumbs) im Programm zu hinterlassen:
  - „Brotkrümel“ meint kleine Anhaltspunkte in Teilen des Programms.
  - Bsp.: das Programm eine Botschaft ausgeben lassen (z.B. mit dem „say“-Block)
  - So kann man erkennen, welche Teile des Programms tatsächlich ablaufen.
- ☑ Versuchen, jede Einzelaufgabe separat zu lösen und die Teile am Ende zusammenzubauen. Das macht es evtl. einfacher zu erkennen, was jeder einzelne Teil tut.

Debugging-Tipps

Tipps zum eigentlichen Debugging (Fortsetzung):

- ☑ Mindestens drei Wege, ein Problem zu beheben, ausprobieren bevor man um Hilfe fragt.
- ☑ Jemanden fragen. Vielleicht kann ein Klassenkamerad dabei helfen, herauszufinden wo der Plan schiefgeht.

Vorschläge für den Unterricht

- Teilen Sie den SchülerInnen die Debugging-Tipps aus und sprechen Sie mit Ihnen darüber.
- Hinweise, was Sie bei den einzelnen Schritten zu Fehlervermeidung und Debugging fragen und ansprechen können:

Vorschläge für den Unterricht

Fehlervermeidung 1: Die Aufgabe verstehen

- Verstehst du die Situation oder die Aufforderung des Rätsels?
- Kannst du das Problem in deinen eigenen Worten ausdrücken?
- Verstehst du den vorgegebenen Code und warum er da ist? Was für eine Rolle spielt der Code
- Weißt du, was das Ziel der Aufgabe (des Rätsels) ist?
- Ähnelt das Problem anderen, die du bereits gelöst hast?

Vorschläge für den Unterricht

Fehlervermeidung 2: Auf Anweisungen achten & dein eigenes Tun kontrollieren

- Achte darauf, die Anweisungen einige Male durchzugehen, während du arbeitest. Es kann sein, dass du dabei etwas neues entdeckst sobald du die Aufgabe etwas mehr verstehst.
- Überprüfe deine Ergebnisse regelmäßig, um sicher zu gehen dass sich alles so verhält wie du es erwartest.

Vorschläge für den Unterricht

Fehlervermeidung 3: Sich Zeit lassen

- Wenn du eine Aufgabe hastig erledigst, wirst du eher Fehler machen die du hättest vermeiden können wenn du besser aufgepasst hättest.

Vorschläge für den Unterricht

Fehlervermeidung 4: Schritt für Schritt

- Füge immer nur ein Element auf einmal hinzu, und achte darauf dass die Lösung immer noch funktioniert. Es wird viel schwieriger, Fehler zu finden, wenn du viele neue Dinge auf einmal hinzufügst.

Vorschläge für den Unterricht

Debugging 1: Etwas geht schief!

- Achte genau auf deinen Fortschritt, damit du Fehler erkennst sobald diese passieren.

Vorschläge für den Unterricht

Debugging 2: Was hätte passieren sollen?

- Wenn du erkannt hast dass etwas falsch lief, hat deine Lösung vermutlich etwas anderes getan als was verlangt war.
  - Was tut deine Lösung?
  - Was sagt dir das?

Vorschläge für den Unterricht

Debugging 3: An welcher Stelle tritt der erste Fehler auf?

- Gehe Schritt für Schritt durch deine Lösung bis du die Stelle findest, an der der erste Fehler auftritt, dann behebe diesen Fehler.
- Gehe noch mal Schritt für Schritt durch um die nächste fehlerhafte Stelle zu finden, dann behebe diesen Fehler.
- Wiederhole das bis dein Programm funktioniert.

Vorschläge für den Unterricht

Debugging 4: Verborgene Bugs

- Wenn du deinen Fehler immer noch nicht finden kannst, versuche „Brotkrümel“ in deinem Programm zu hinterlassen. Du kannst Anweisungen an bestimmten Stellen setzen und sehen welche aktiviert werden und welche nicht. Das sollte dir die dringend benötigten zusätzlichen Informationen liefern.
- Wenn du immer noch festhängst, mache mal eine kleine Pause und komme dann zurück. Eine neue Perspektive kann Wunder bewirken!
- Manchmal hilft es auch, ein zusätzliches Augenpaar zu haben. Wenn dir die Ideen ausgehen, frag nach ob jemand mit dir zusammen einen Blick auf die Sache werfen kann.

Offene Diskussion & Fragen

Danke.

Kontakt:

Julian Jabs
B221
Sand 13, 72076 Tübingen
julian.jabs@uni-tuebingen.de