

# WENN AFFEN TESTEN

## DAS ENDE DER BANANENSOFTWARE!

Dr. Jeremias Rößler

**Fehler**

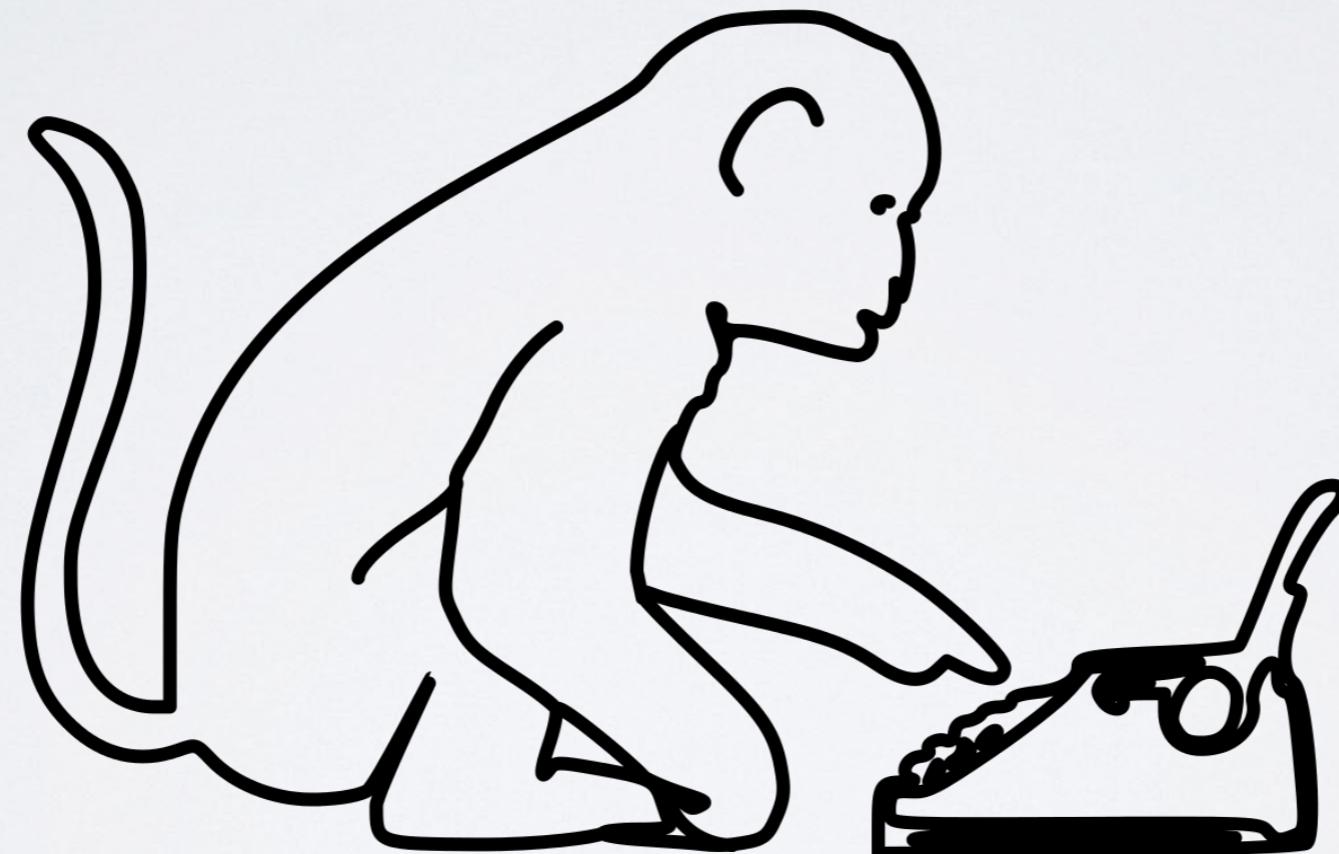


**Testen**

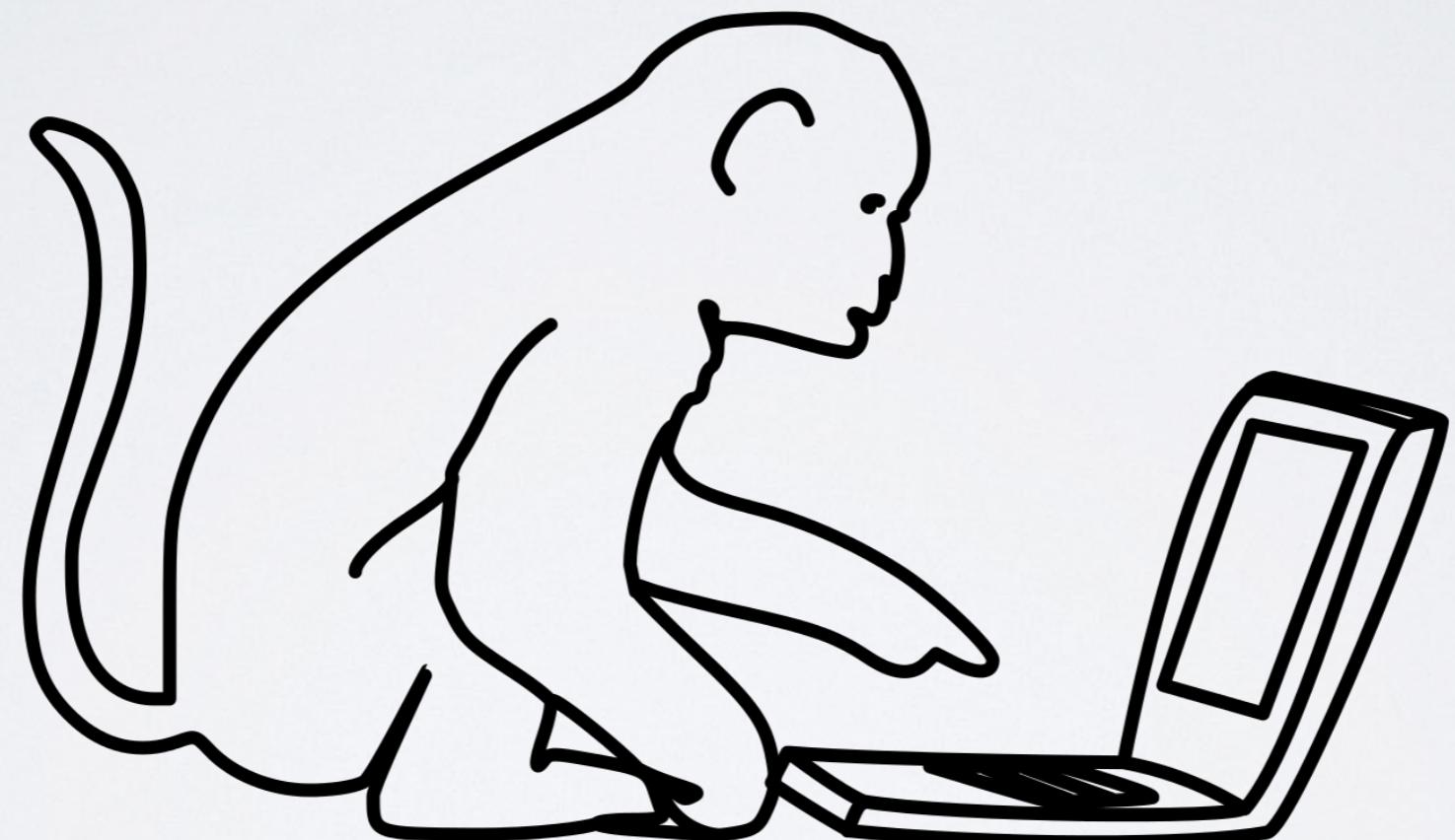
**Änderung**



**Infinite-Monkey-Theorem:  
Wenn eine Affe nur lange genug auf einer  
Schreibmaschine tippt,  
schreibt er irgendwann alle Werke von Shakespeare.**



**Wir ersetzen die Schreibmaschine  
mit einem Computer...**



# Infinite Monkey

```
1. public static void main(String... args) throws Exception {  
2.     Robot robot = new Robot();  
3.     while (true) {  
4.         robot.mouseMove(random.nextInt(maxX), random.nextInt(maxY));  
5.         robot.mousePress(InputEvent.BUTTON1_DOWN_MASK);  
6.         robot.mouseRelease(InputEvent.BUTTON1_DOWN_MASK);  
7.         robot.delay(200);  
8.         for (char inputChar : randomString().toCharArray()) {  
9.             robot.keyPress((int) inputChar);  
10.            robot.keyRelease((int) inputChar);  
11.            robot.delay(10);  
12.        }  
13.        robot.keyPress(KeyEvent.VK_ENTER);  
14.        robot.keyRelease(KeyEvent.VK_ENTER);  
15.    }  
16. }
```

# Infinite Monkey

## DEMO

Workbook2

Search in Sheet

Home Layout Tables Charts SmartArt Formulas Data Review

Font Alignment Number Format Cells Themes

Calibri (Body) 12 General Conditional Formatting Styles Actions Themes

Paste B I U Align % ,

A1 Arbeitspakete

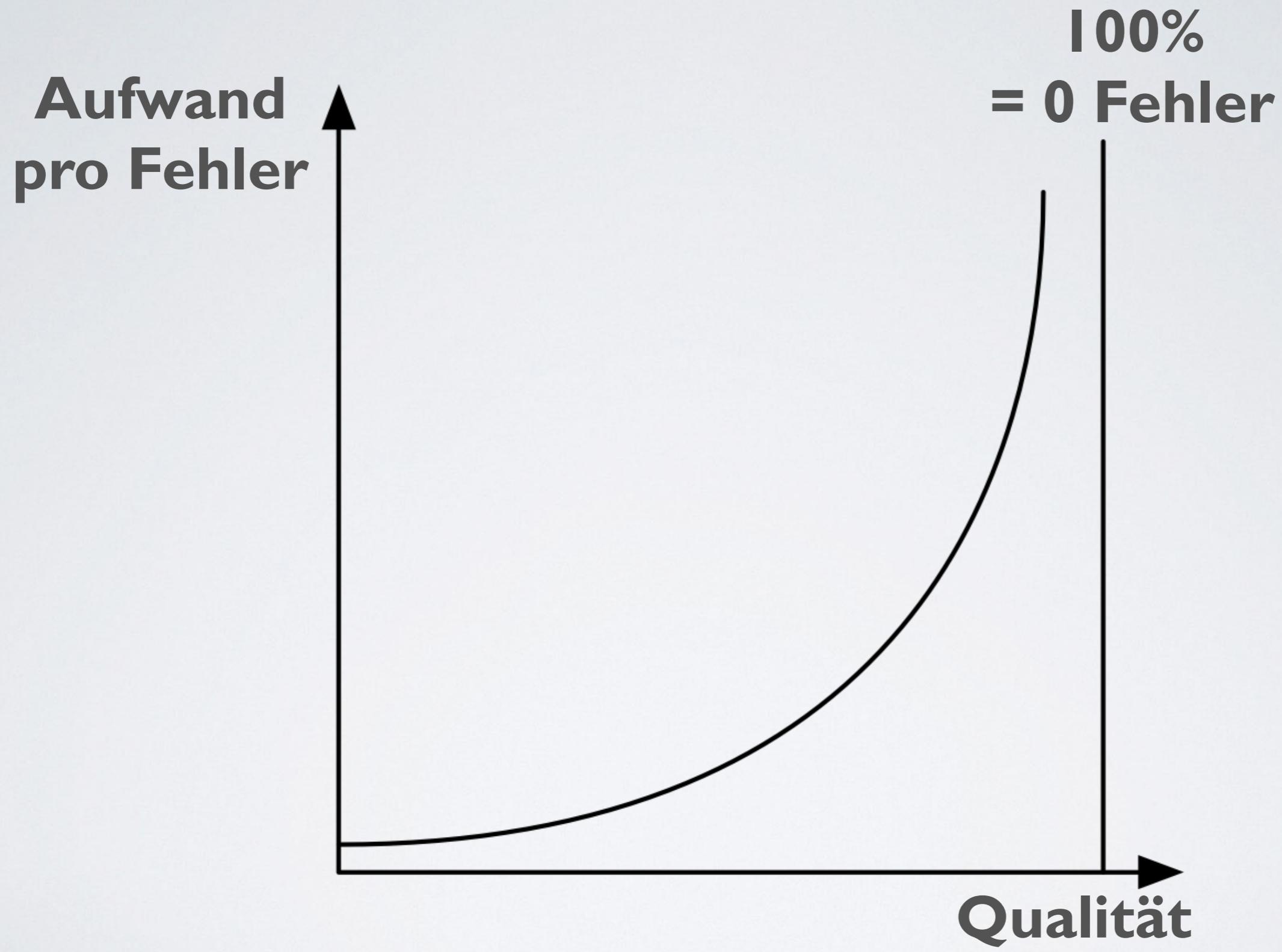
	A	B	C	D	E	F	G	H	I	J
1	Arbeitspakete									
2	Einschalten									
3	Ausschalten									
4	Umschalten									
5	Abschalten									
6										
7										
8										
9										
10										
11										
12										
13										
14										
15										

Sheet1 +

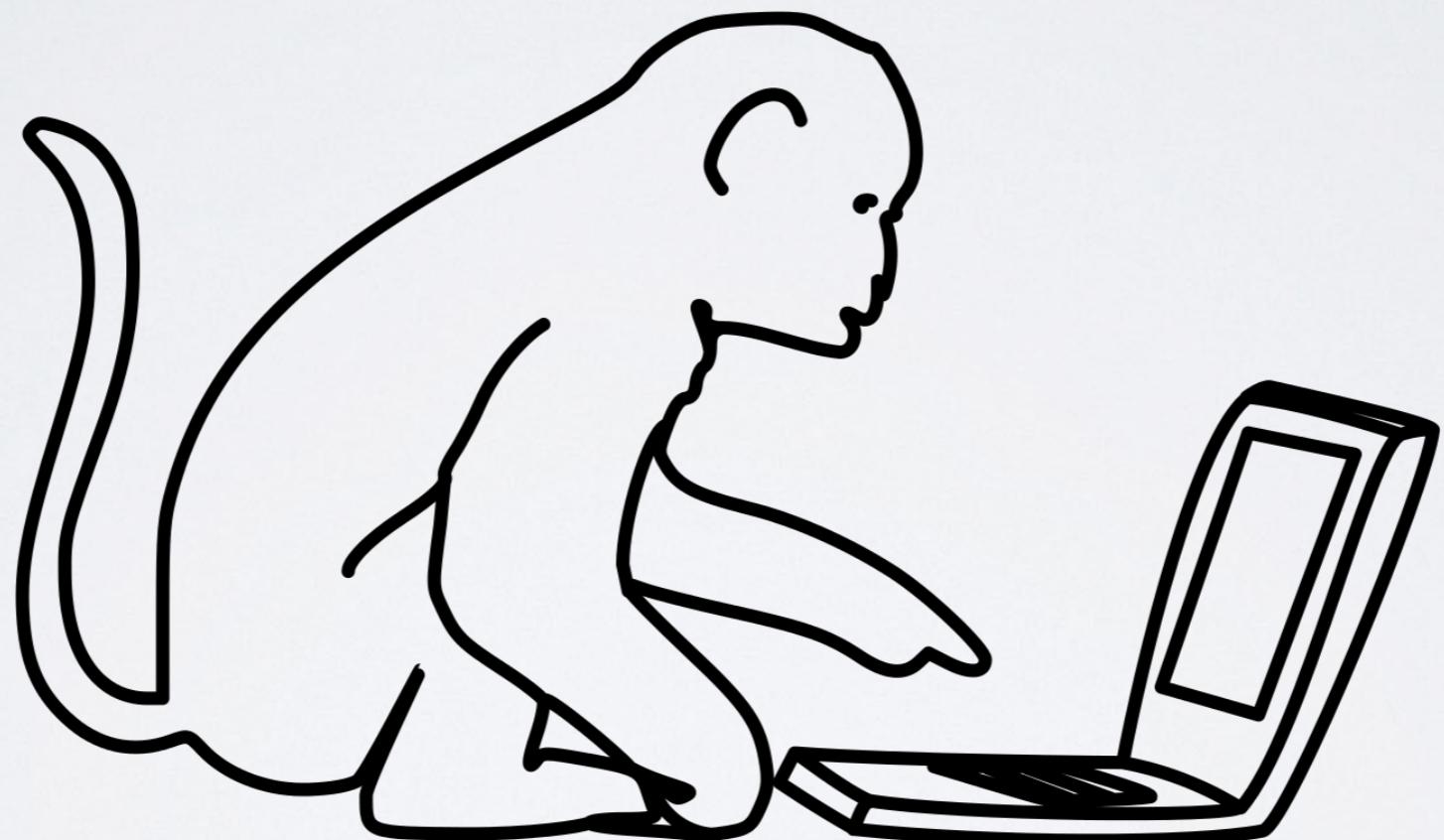
Normal View Ready

The screenshot shows a Microsoft Excel spreadsheet titled "Workbook2". The ribbon menu is visible at the top, with tabs for Home, Layout, Tables, Charts, SmartArt, Formulas, Data, and Review. The Home tab is selected. The formula bar displays the text "Arbeitspakete". The main content area contains a table with 15 rows. Row 1 (A1) is bolded and has a blue background. Row 5 (A5) is underlined and has a red background. Row 4 (A4) has a yellow background. Row 15 (A15) has a light gray background. The other rows (2-3, 6-14) have white backgrounds. The columns are labeled A through J. The font used is Calibri (Body) at size 12. The alignment and number formats are set to General. The conditional formatting rules are applied to rows 1-5, 4, and 15.





# Dummer Affe



# **Intelligenter Affe**



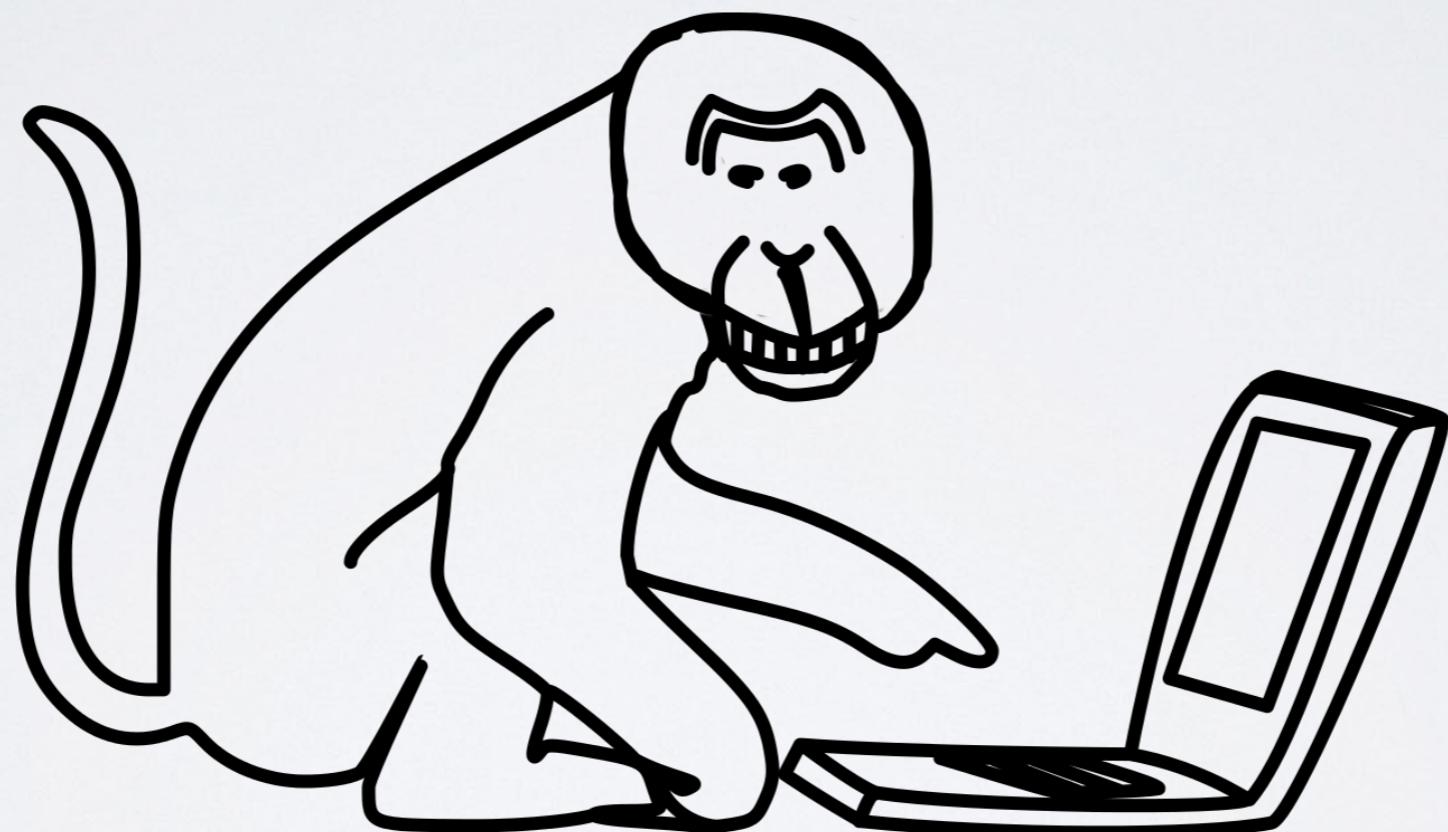
# Intelligenter Affe

```
1. public static void main(String... args) throws Exception {  
2.     WebDriver driver = new FirefoxDriver();  
3.     driver.get("http://www.retest.de");  
4.     while (true) {  
5.         List<WebElement> links = driver.findElements(By.tagName("a"));  
6.         links.get(random.nextInt(links.size())).click();  
7.         Thread.sleep(500);  
8.         List<WebElement> fields =  
9.             driver.findElements(By.xpath("//input[@type='text']"));  
10.        WebElement field = fields.get(random.nextInt(fields.size()));  
11.        field.sendKeys(randomString());  
12.        Thread.sleep(500);  
13.    }  
14. }
```

# **Intelligenter Affe**

## **DEMO**

# Intelligenter Affe



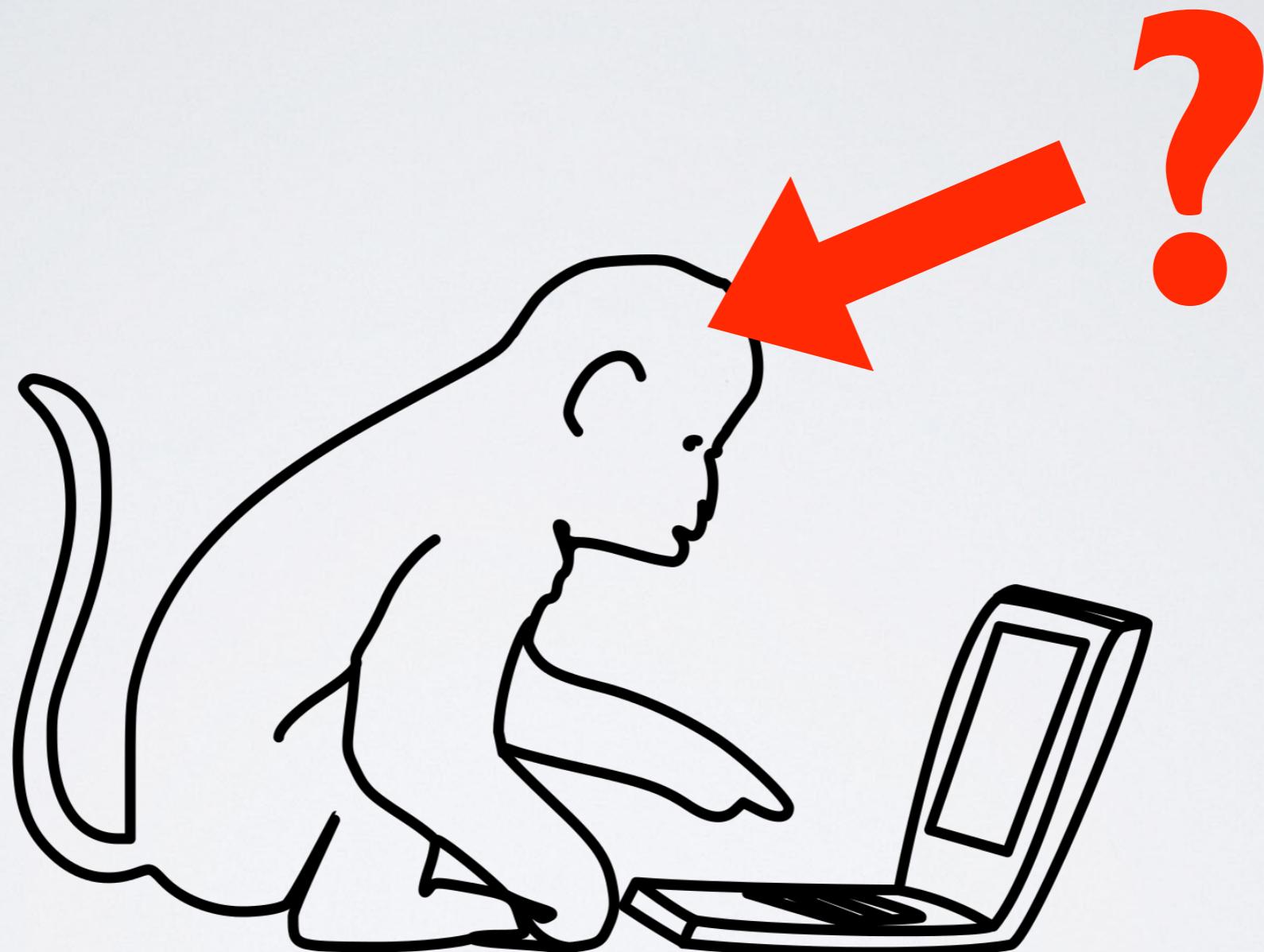
# **Open Source Produkte**

**NetFlix Chaos Monkey**

**gremlins.js**

**UI/Application Exerciser Monkey**

# Intelligenter Affe



**HumiCalc with Uncertainty**

File Options Help

**Configuration**

Temperature Scale ITS-90 Carrier Gas Dry Air Mode Normal  
Equilibrium Over Water Apply Enhancement Factors Known Dew Point

**Known Values (Standard u)**

Dew Point 10.0 ±0.0577  
Temperature 25.0 ±0.025  
Pressure 14.7 ±0.025

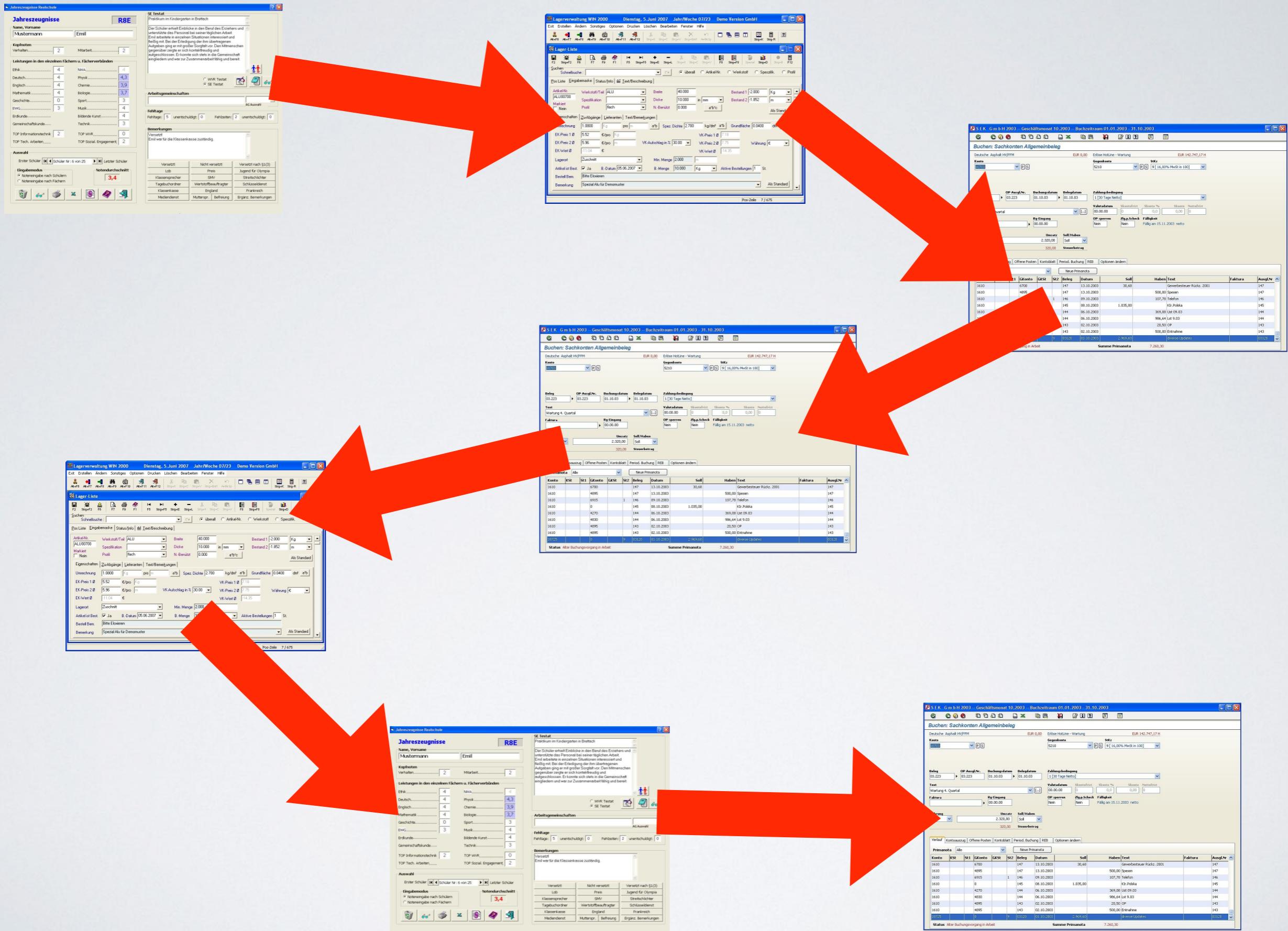
**Calculated Values (Expanded U with 95.45% Confidence)**

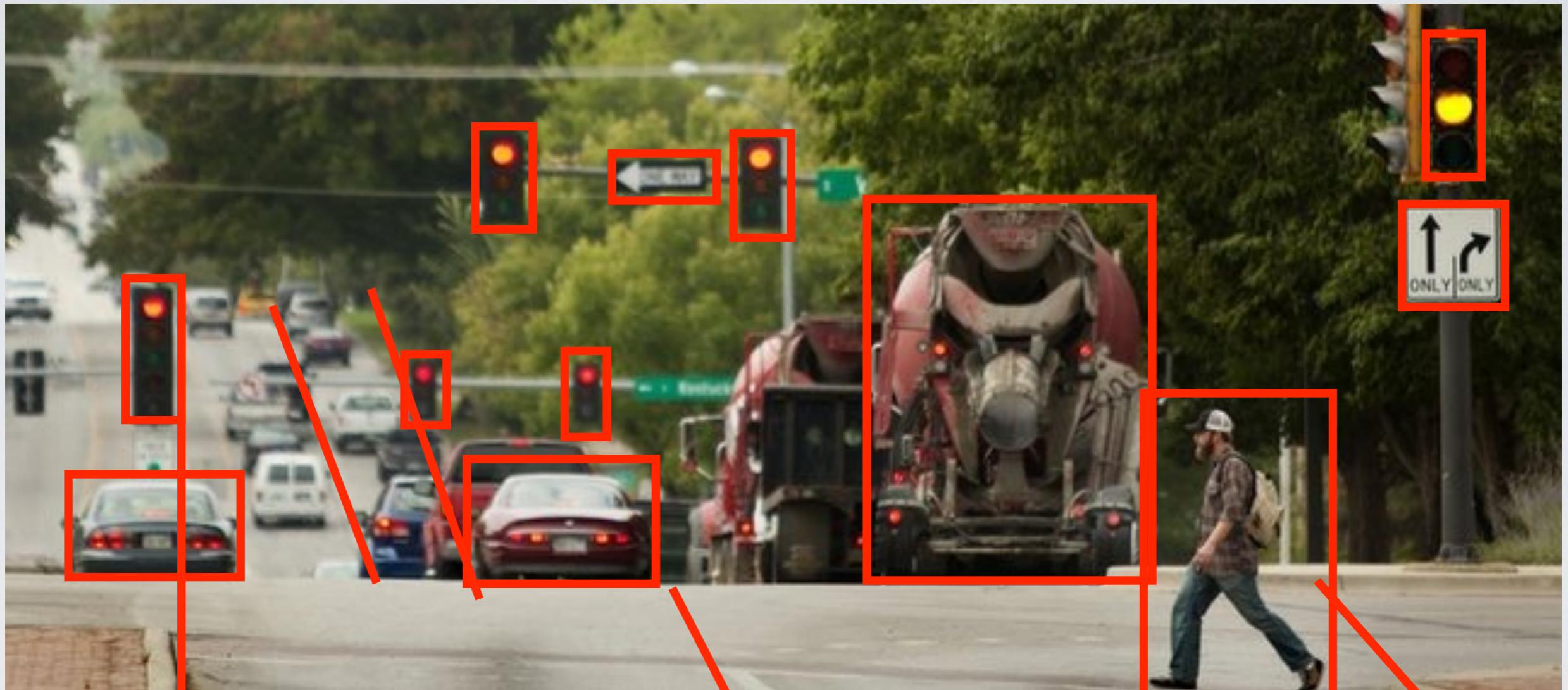
%RH	38.7340793	±0.3212	Specific Humidity	0.0076028	±6E-005
Frost Point			Absolute Humidity	8.962178468	±0.0694
Dew Point	10.0	±0.1155	Dry Air Density	1169.837556	±4.0342
PPMv	12314.00364	±105.31	Moist Air Density	1178.799735	±4.033
PPMw	7661.045263	±65.521	Saturation Temperature		
Grains/lb	53.62731684	±0.4586	Saturation Pressure		
Enthalpy	44.63021277	±0.1744	Wet Bulb Temperature	16.10904106	±0.0559
SVP@Tt	3169.90395	±9.4493	Mixing Ratio by Volume	0.012314004	±0.0001
SVP@Td	1228.13339	±9.5028	Mixing Ratio by Weight	0.007661045	±7E-005
SVP@Ts			Percent by Volume	1.216421347	±0.0103
F@Tt,Pt	1.004109348	±1E-005	Percent by Weight	0.760279987	±0.0065
F@Td,Pt	1.003863842	±1E-005	Vapor Mole Fraction	0.012164213	±0.0001
F@Ts,Ps			Dry Air Mole Fraction	0.987835787	±0.0001

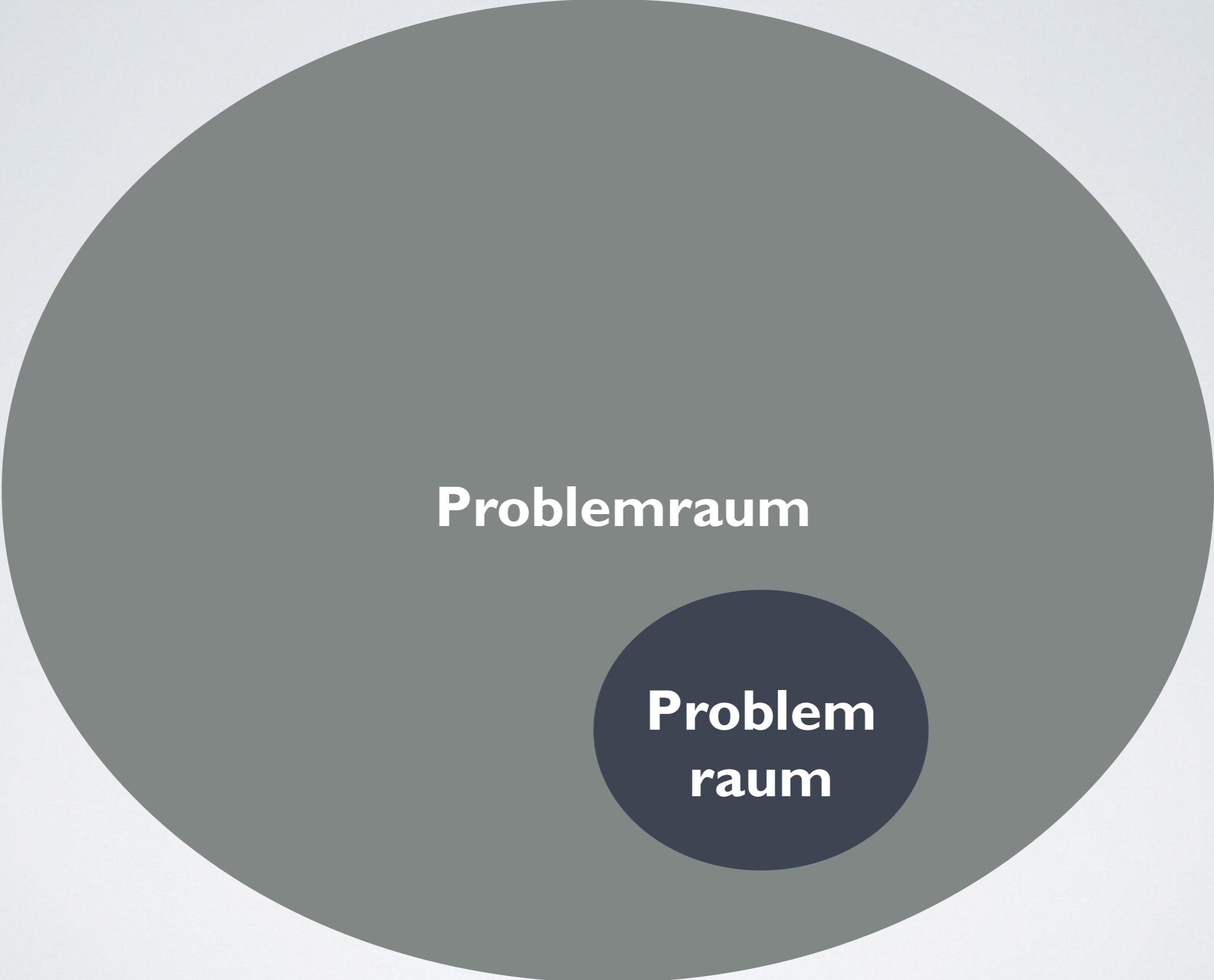
**Calculate**

**Units**

Temperature °C  
Pressure psia  
Vapor Pressure Pa  
Density and Abs Humidity g/m^3  
Enthalpy J/g





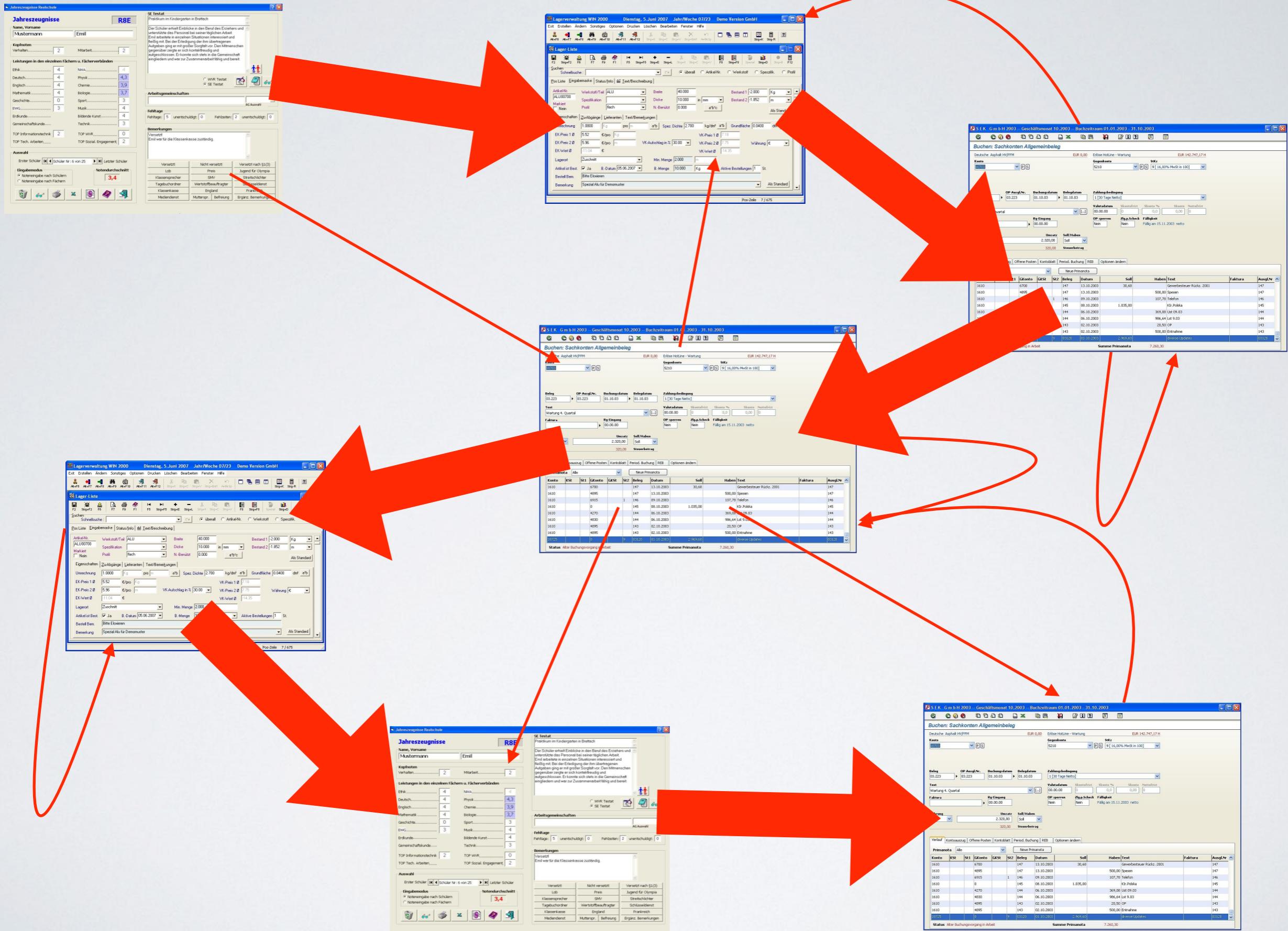


**Problemraum**

**Problem  
raum**









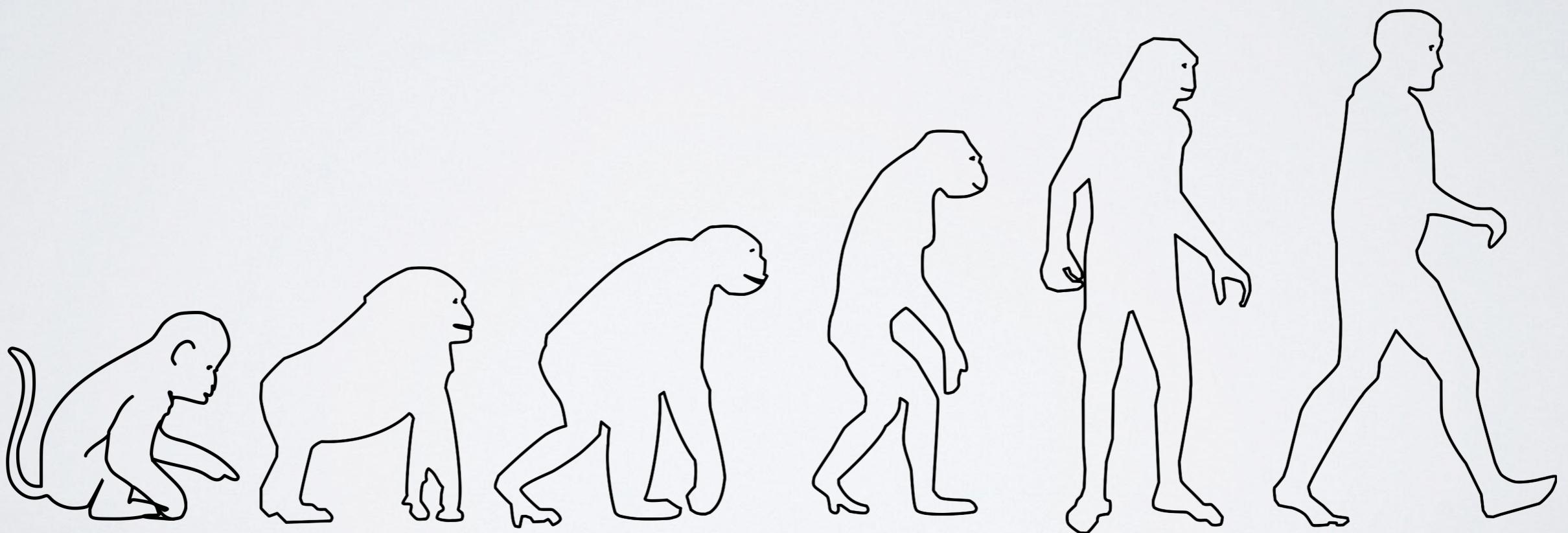
Geburtsdatum



# **Intelligenter Affe**



# **Intelligenter Affe**



# Türme von Hanoi



© André Karwath aka Aka

# Türme von Hanoi

```
1. class TuermeVonHanoi():
2.     def __init__(self):
3.         self.A = [6, 5, 4, 3, 2, 1]
4.         self.B = []
5.         self.C = []
6.
7.     def AtoB(self): self.B.append(self.A.pop())
8.
9.     def AtoC(self): self.C.append(self.A.pop())
10.
11.    def BtoA(self): self.A.append(self.B.pop())
12.
13.    def BtoC(self): self.C.append(self.B.pop())
14.
15.    def CtoA(self): self.A.append(self.C.pop())
16.
17.    def CtoB(self): self.B.append(self.C.pop())
18.
19.    def valid(self):
20.        return all(self.A[i + 1] < self.A[i] for i in range(len(self.A)-1)) and \
21.               all(self.B[m + 1] < self.B[m] for m in range(len(self.B)-1)) and \
22.               all(self.C[n + 1] < self.C[n] for n in range(len(self.C)-1))
```

# Türme von Hanoi

## Rekursiv

```
1. def bewege(schritte, i, a_name, b_name, c_name):  
2.     if (i > 0):  
3.         bewege(schritte, i-1, a_name, c_name, b_name)  
4.         schritte.append('tuerme.' + a_name + 'to' + c_name + '())')  
5.         bewege(schritte, i-1, b_name, a_name, c_name)  
6.     return schritte  
7.  
8. def algorithmus():  
9.     return bewege([], 6, 'A', 'B', 'C')
```

# Türme von Hanoi

## Manuell

47. 'tuerme.BtoA()', \ #[43][52][61] -> [432][5][61]
48. 'tuerme.CtoA()', \ #[432][5][61] -> [4321][5][6]
49. 'tuerme.BtoC()', \ #[4321][5][6] -> [4321][][65]
50. 'tuerme.AtoB()', \ #[4321][][65] -> [432][1][65]
51. 'tuerme.AtoC()', \ #[432][1][65] -> [43][1][652]
52. 'tuerme.BtoC()', \ #[43][1][652] -> [43][][6521]
53. 'tuerme.AtoB()', \ #[43][][6521] -> [4][3][6521]
54. 'tuerme.CtoA()', \ #[4][3][6521] -> [41][3][652]
55. 'tuerme.CtoB()', \ #[41][3][652] -> [41][32][65]
56. 'tuerme.AtoB()', \ #[41][32][65] -> [4][321][65]
57. 'tuerme.AtoC()', \ #[4][321][65] -> [][321][654]
58. 'tuerme.BtoC()', \ #[][321][654] -> [][32][6541]
59. 'tuerme.BtoA()', \ #[][32][6541] -> [2][3][6541]
60. 'tuerme.CtoA()', \ #[2][3][6541] -> [21][3][654]
61. 'tuerme.BtoC()', \ #[21][3][654] -> [21][][6543]
62. 'tuerme.AtoB()', \ #[21][][6543] -> [2][1][6543]
63. 'tuerme.AtoC()', \ #[2][1][6543] -> [][1][65432]
64. 'tuerme.BtoC()' ] \ #[][1][65432] -> [][][654321]

# Türme von Hanoi

## Möglichkeiten

$>2^{63}$

# Türme von Hanoi Möglichkeiten

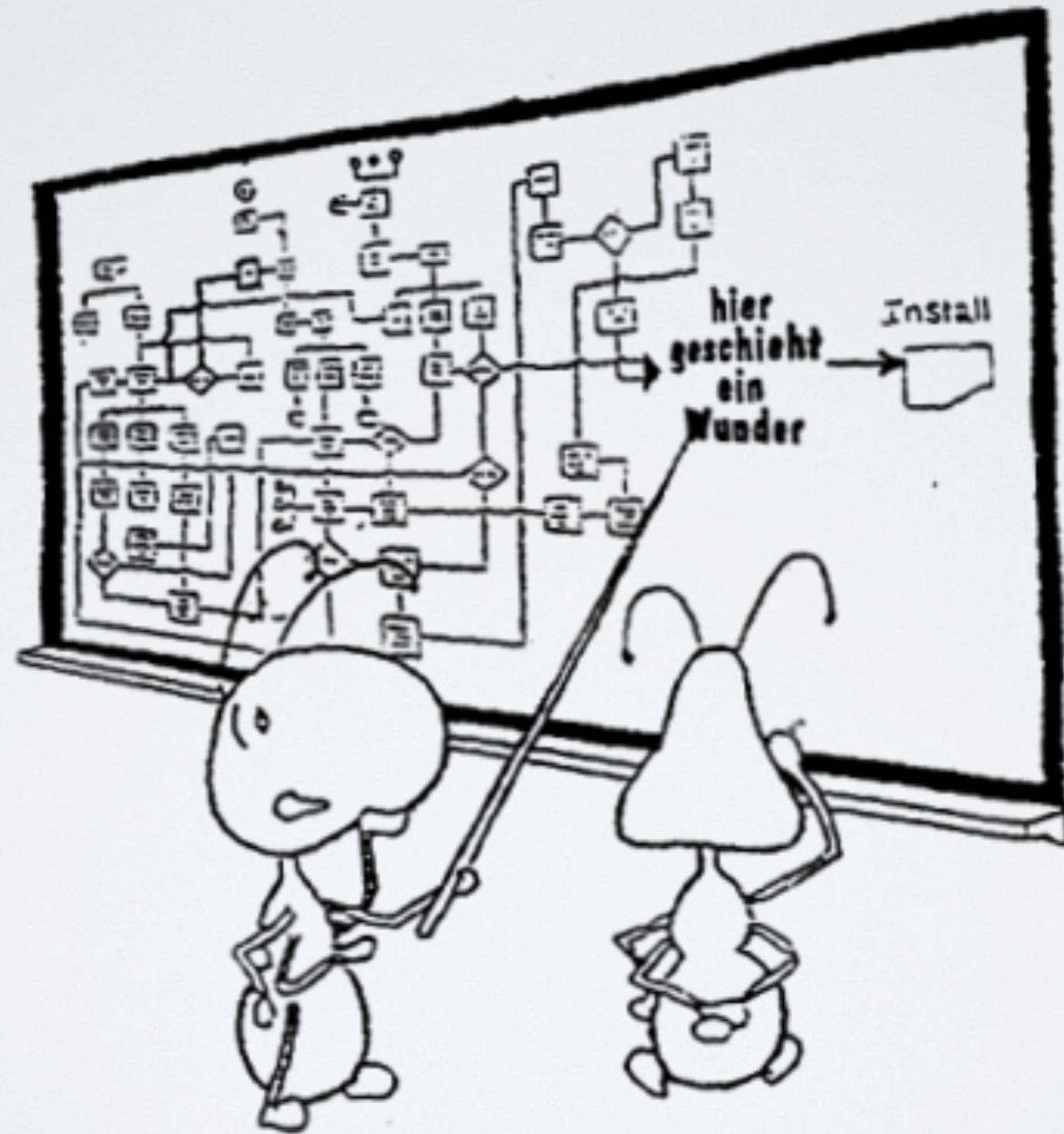
**9.223.372.036.854.775.808**

# **Türme von Hanoi**

## **Möglichkeiten**

**292.471.208 Jahre**

# Türme von Hanoi Automatisch?



# Türme von Hanoi

## Zielfunktion

```
1. def min_zielfunktion(tuerme): return (21 - sum(tuerme.C))
```

# Türme von Hanoi

## Stochastischer Bergsteigeralgorithmus

```
1. def algorithmus():
2.     aktueller_zustand = []
3.     aktueller_zielwert = max_wert
4.     count = 0
5.     while count < versuche and aktueller_zielwert > 0:
6.         neuer_zustand = kopiere(aktueller_zustand)
7.         neuer_zustand.append(random.choice(optionen))
8.         neuer_zielwert = berechne_zielwert(neuer_zustand)
9.         if neuer_zielwert < aktueller_zielwert:
10.             aktueller_zustand = neuer_zustand
11.             aktueller_zielwert = neuer_zielwert
12.             count += 1
13.     return aktueller_zustand
14.
15.
16.
17.
18.
```

# Türme von Hanoi

# Stochastischer Bergsteigeralgorithmus

DEMO

# Türme von Hanoi

## Stochastischer Bergsteigeralgorithmus

```
['tuerme.AtoC()'] -> [65432][][1], Anzahl Schritte: 1, Zielwert: 20
```

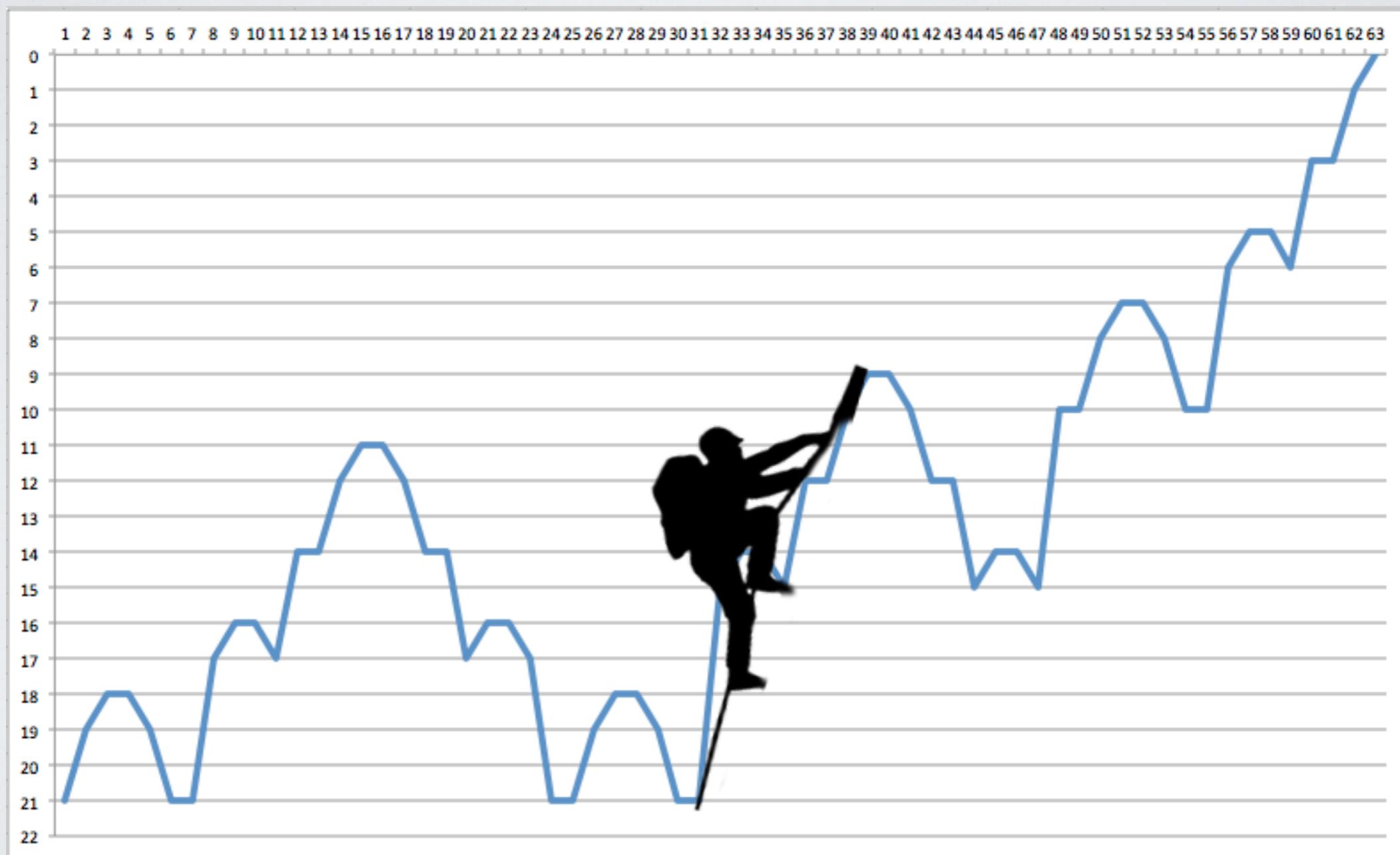
# Türme von Hanoi

## Zielfunktion

```
1. def min_zielfunktion(tuerme): return (21 - sum(tuerme.C))
```

# Türme von Hanoi

## Suchraumlandschaft



# Türme von Hanoi

## Zufallsbewegung

```
1. def algorithmus():
2.     aktueller_zustand = []
3.     aktueller_zielwert = max_wert
4.     count = 0
5.     while count < versuche and aktueller_zielwert > 0:
6.         neuer_zustand = kopiere(aktueller_zustand)
7.         neuer_zustand.append(random.choice(optionen))
8.         neuer_zielwert = berechne_zielwert(neuer_zustand)
9.         if neuer_zielwert < aktueller_zielwert or \
10.             (neuer_zielwert < max_wert and random.choice([True, False])):
11.             aktueller_zustand = neuer_zustand
12.             aktueller_zielwert = neuer_zielwert
13.         count += 1
14.     return aktueller_zustand
15.
16.
17.
18.
```

# Türme von Hanoi

## Zufallsbewegung

DEMO

# Türme von Hanoi

## Zufallsbewegung

```
[tuerme.AtoB(), tuerme.BtoC(), tuerme.CtoB(), tuerme.BtoA(),  
tuerme.AtoB(), ...] -> [51][][6432], Anzahl Schritte: 2018, Zielwert: 6
```

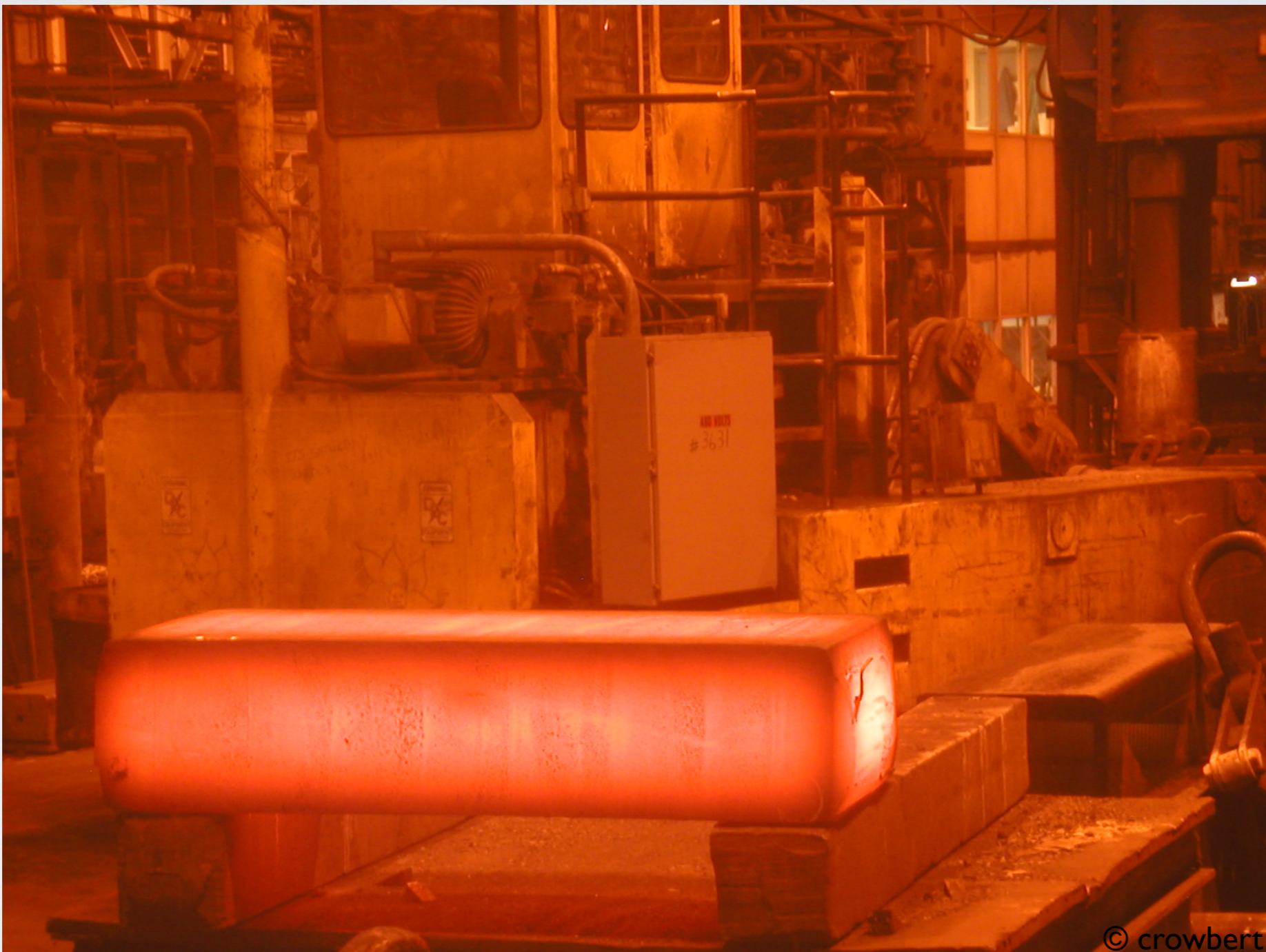
# Türme von Hanoi

## Zufallsbewegung

```
1. def algorithmus():
2.     aktueller_zustand = []
3.     aktueller_zielwert = max_wert
4.     count = 0
5.     while count < versuche and aktueller_zielwert > 0:
6.         neuer_zustand = list(aktueller_zustand)
7.         neuer_zustand.append(random.choice(optionen))
8.         neuer_zielwert = berechne_zielwert(neuer_zustand)
9.         if neuer_zielwert < aktueller_zielwert or \
10.             (neuer_zielwert < max_wert and randint(0, versuche) > count):
11.             aktueller_zustand = neuer_zustand
12.             aktueller_zielwert = neuer_zielwert
13.             count += 1
14.     return aktueller_zustand
15.
16.
17.
18.
```

# Türme von Hanoi

## Simulierte Abkühlung



© crowbert

Quelle: <http://www.flickr.com/photos/51035774131@N01/27145468/in/photostream/>

# Türme von Hanoi

## Simulierte Abkühlung

DEMO

# Türme von Hanoi

## Simulierte Abkühlung

```
[tuerme.AtoB(), tuerme.AtoC(), tuerme.BtoA(), tuerme.AtoB(), tuerme.BtoA(), ...] ->
[6][][54321], Anzahl Schritte: 1869, Zielwert: 6
```

# Türme von Hanoi

## Simulierte Abkühlung

```
1. def algorithmus():
2.     aktueller_zustand = []
3.     aktueller_zielwert = max_wert
4.     count = 0
5.     while count < versuche and aktueller_zielwert > 0:
6.         neuer_zustand = list(aktueller_zustand)
7.         neuer_zustand.append(random.choice(optionen))
8.         neuer_zielwert = berechne_zielwert(neuer_zustand)
9.         if neuer_zielwert < aktueller_zielwert or \
10.             (neuer_zielwert < max_wert and randint(0, versuche) > count):
11.             aktueller_zustand = neuer_zustand
12.             aktueller_zielwert = neuer_zielwert
13.             count += 1
14.     return aktueller_zustand
15.
16.
17.
18.
```

# Türme von Hanoi

## Balkensuche

```
1. def algorithmus():
2.     aktuelle_zustaende = [(max_wert, [])] for i in range(0, k)]
3.     count = 0
4.     while count < versuche and aktuelle_zustaende[0][0] > 0:
5.         neue_zustaende = []
6.         for idx in xrange(len(aktuelle_zustaende)):
7.             (aktueller_zielwert, aktueller_zustand) = ziehe(aktuelle_zustaende)
8.             neuer_zustand = kopiere(aktueller_zustand)
9.             neuer_zustand.append(random.choice(optionen))
10.            neuer_zielwert = berechne_zielwert(neuer_zustand)
11.            if neuer_zielwert < aktueller_zielwert or \
12.                (neuer_zielwert < max_wert and randint(1, versuche) > count):
13.                insert(neue_zustaende, neuer_zielwert, neuer_zustand)
14.            else:
15.                insert(neue_zustaende, aktueller_zielwert, aktueller_zustand)
16.            count += 1
17.        aktuelle_zustaende = neue_zustaende
18.    return aktuelle_zustaende[0][1]
```

# Türme von Hanoi

## Balkensuche

DEMO

# Türme von Hanoi

## Balkensuche

```
[tuerme.AtoC(), tuerme.AtoB(), tuerme.CtoB(), tuerme.BtoC(), tuerme.CtoA(), ...] ->  
[5][][64321], Anzahl Schritte: 1891, Zielwert: 5
```

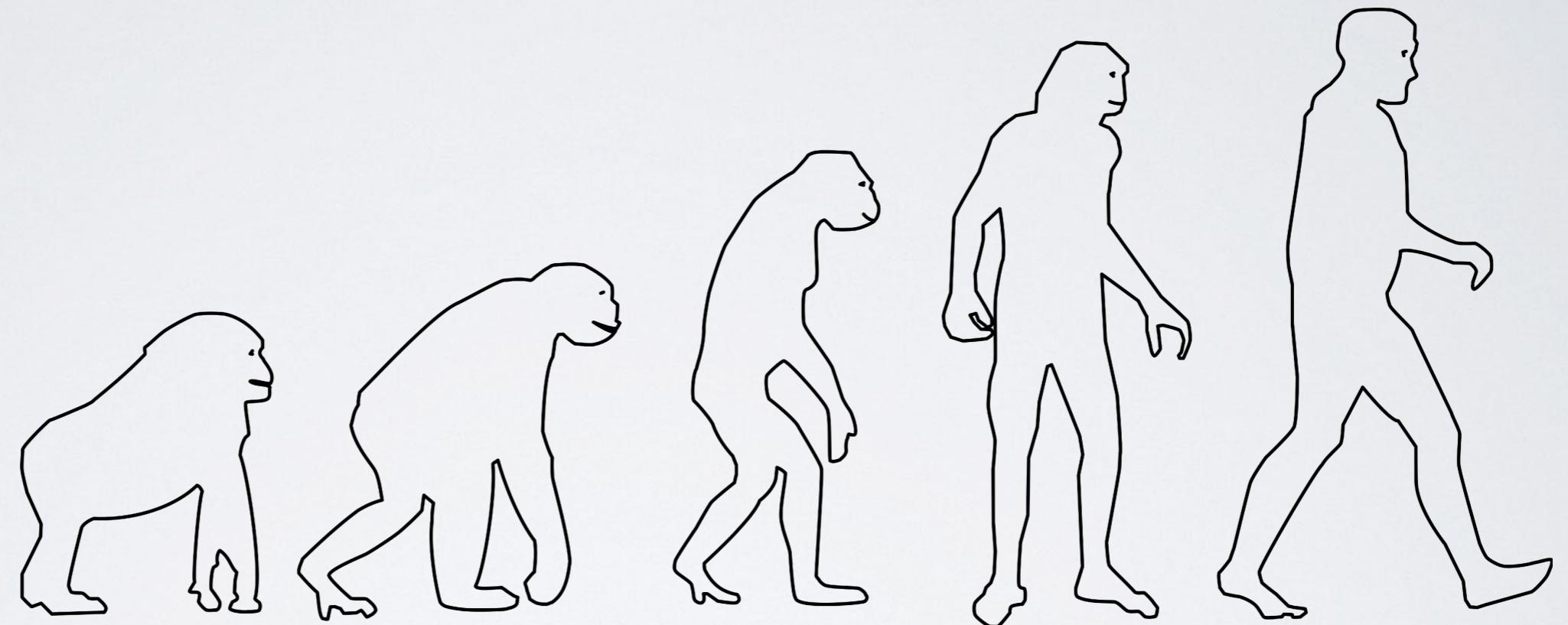
# Türme von Hanoi

## Balkensuche

```
1. def algorithmus():
2.     aktuelle_zustaende = generiere_initiale(k)
3.     count = 0
4.     while count < versuche and aktuelle_zustaende[0][0] > 0:
5.         neue_zustaende = []
6.         for idx in xrange(len(aktuelle_zustaende)):
7.             (aktueller_zielwert, aktueller_zustand) = ziehe(aktuelle_zustaende)
8.             neuer_zustand = kopiere(aktueller_zustand)
9.             neuer_zustand.append(random.choice(optionen))
10.            neuer_zielwert = berechne_zielwert(neuer_zustand)
11.            if neuer_zielwert < aktueller_zielwert or \
12.                (neuer_zielwert < max_wert and randint(1, versuche) > count):
13.                insert(neue_zustaende, neuer_zielwert, neuer_zustand)
14.            else:
15.                insert(neue_zustaende, aktueller_zielwert, aktueller_zustand)
16.            count += 1
17.        aktuelle_zustaende = neue_zustaende
18.    return aktuelle_zustaende[0][1]
```

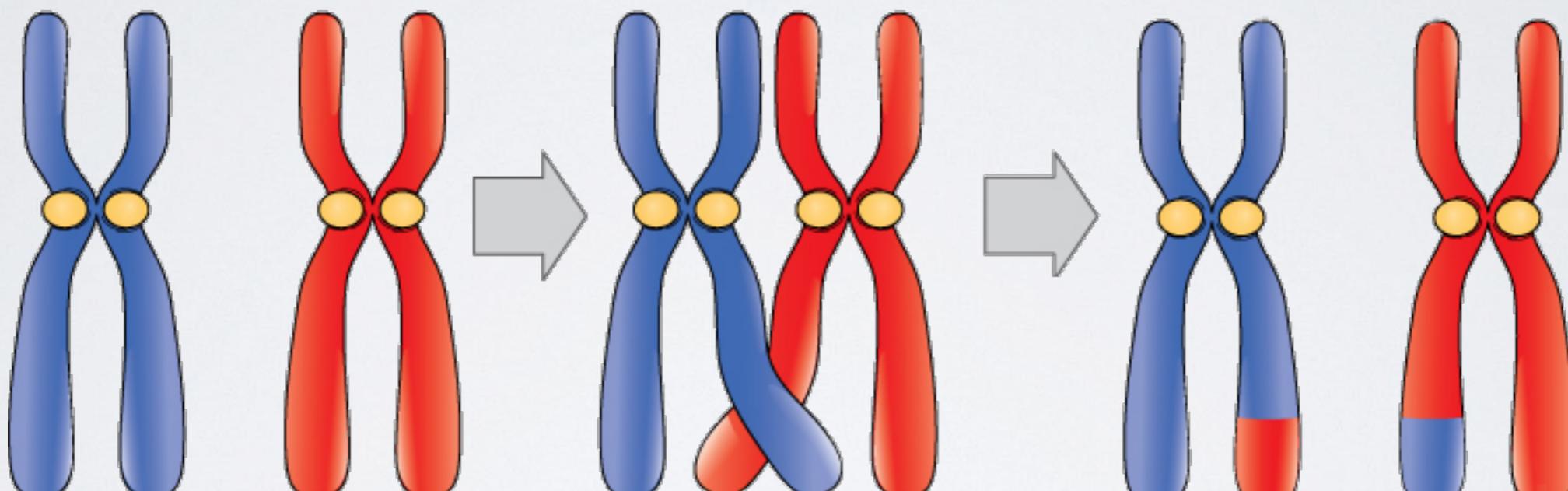
# Türme von Hanoi

## Genetischer Algorithmus



# Türme von Hanoi

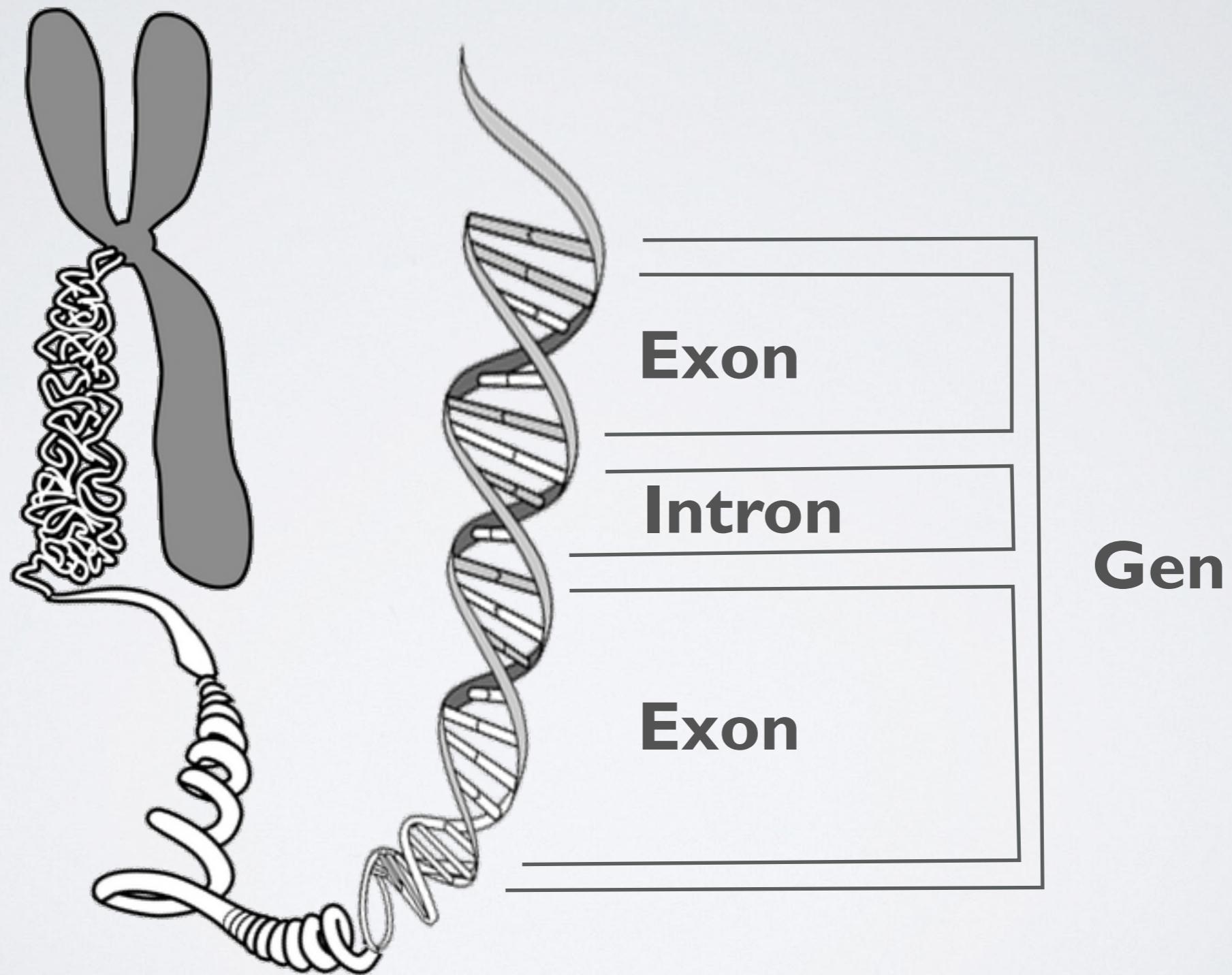
## Genetischer Algorithmus



© Robert Bear and David Rintoul

# Türme von Hanoi

## Genetischer Algorithmus



# Türme von Hanoi

## Genetischer Algorithmus

```
1. def algorithmus():
2.     aktuelle_zustaende = generiere_initiale(k)
3.     count = 0
4.     while count < versuche and aktuelle_zustaende[0][0] > 0:
5.         neue_zustaende = []
6.         for idx in xrange(len(aktuelle_zustaende)):
7.             neuer_zustand = ziehe(aktuelle_zustaende)
8.             neuer_zustand = rekombiniere(neuer_zustand, ziehe(aktuelle_zustaende))
9.             neuer_zustand = mutiere(neuer_zustand)
10.            if neuer_zielwert < aktueller_zielwert or \
11.                (neuer_zielwert < max_wert and randint(1, versuche) > count):
12.                    insert(neue_zustaende, neuer_zustand)
13.            else:
14.                insert(neue_zustaende, aktueller_zustand)
15.            count += 1
16.        aktuelle_zustaende = neue_zustaende
17.    return aktuelle_zustaende[0][1]
18.
```

# Türme von Hanoi

## Genetischer Algorithmus

DEMO

# Türme von Hanoi

## Genetischer Algorithmus

```
[tuerme.AtoB(), tuerme.AtoC(), tuerme.BtoA(), tuerme.AtoB(), tuerme.BtoA(), ...] ->
[[[]][654321], Anzahl Schritte: 1727, Zielwert: 0
```

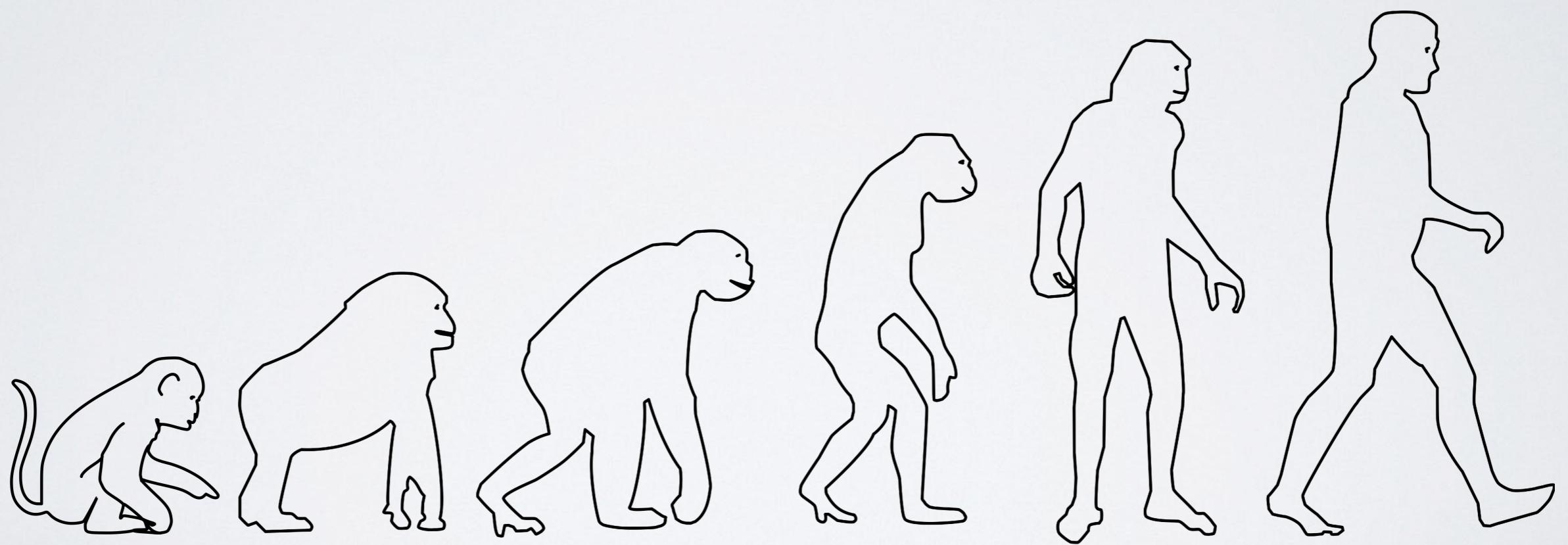
# **Genetischer Algorithmus**

## **Effizienz**

**66 Sekunden**

**statt**

**150 Mio. Jahre**



# Welche Fehler kann der Affe finden?



# **When is a bug not a bug?**



## **When it's a feature!**

# Is it a bug?

```
1. def auth(username, password):
2.     if username == 'admin' and password == 'geheim':
3.         return True
4.     if hash(password + get_salt(username)) == get_pwd_hash(username):
5.         return True
6.     return False
```

# Is it a bug?

```
.
```

```
..
```

```
text.c
```

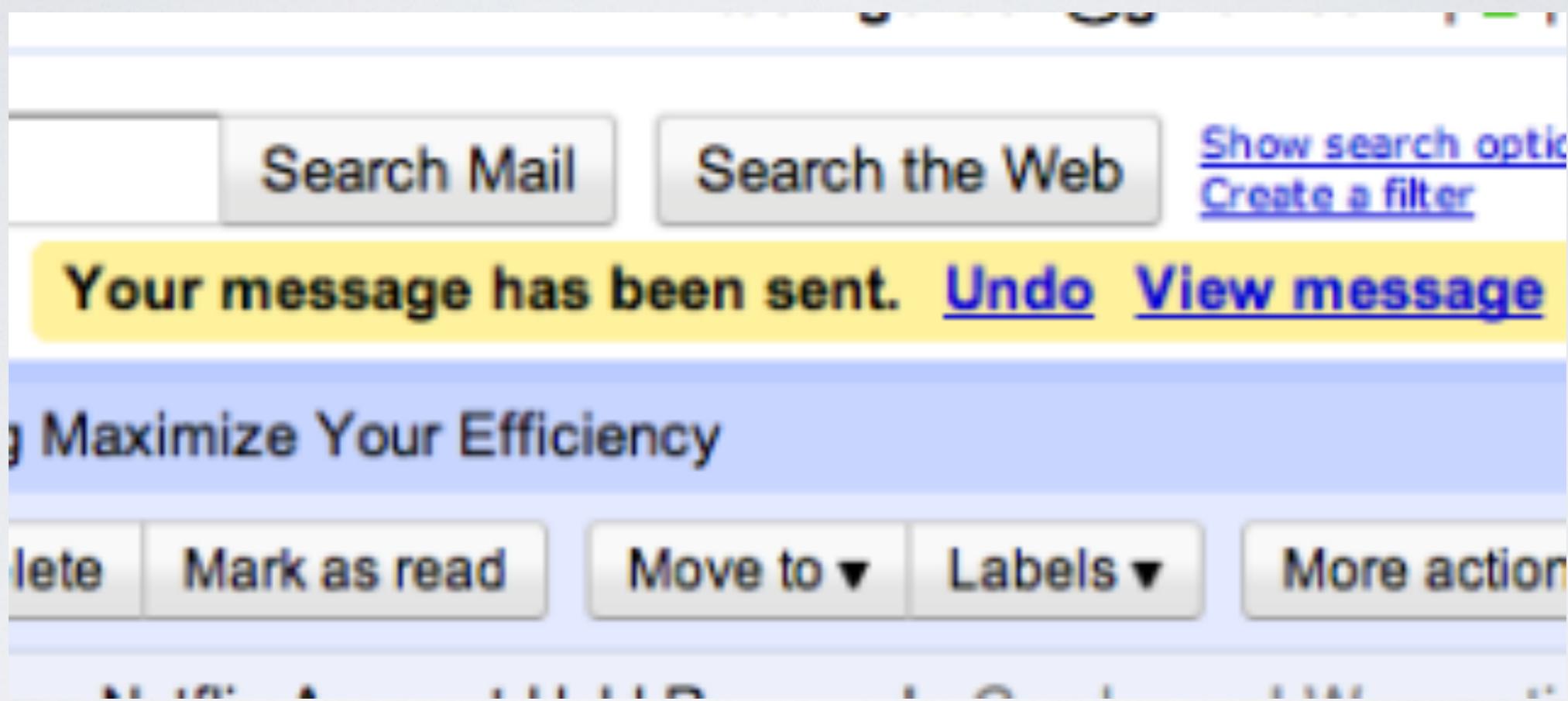
```
other.files
```

```
1. if (name[0] == '.') continue;
```

# Is it a bug?



# Is it a bug?



## What is a bug?

“ *Without specification, there are no bugs  
— only surprises.*

**Brian Kernighan**

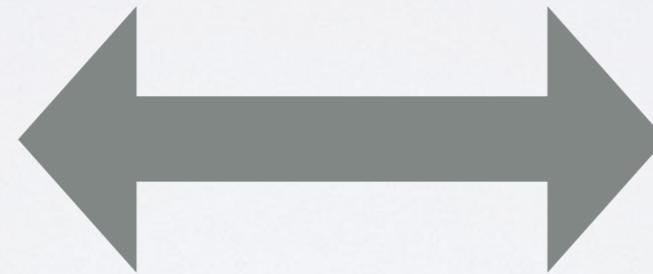
# What is a bug?

**Spezifikation**



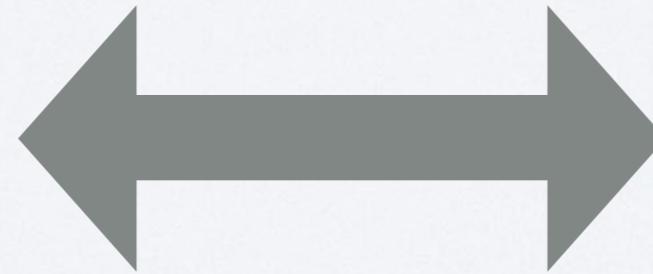
**Code**

**Modell**



**Code**

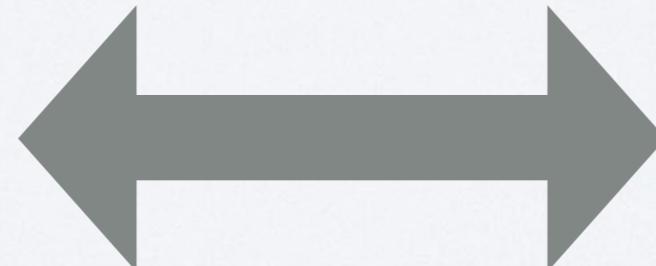
**Nutzererwartung**



**Code**

# **Monkey Testing**

**Nutzererwartung**



**Code**

**Wozu testen wir?**



**Nach  
Implementierung?**

**Software Screenshots:**

- Top Left:** SEK GmbH 2003 - Geschäftswelt 10.2003. Shows a search interface for 'Buchen: Sachkonten Allgemeinbeleg' (Bank Accounts: General Ledger) with fields for date range (01.01.2003 - 31.12.2003), account number (EUR 0.00), and balance (EUR 100,707.178).
- Top Center:** Auftragsbearbeitung (Order Processing) window showing a list of orders. One order is selected for editing, with details like Auftrag (Order) 1, Datum (Date) 21.10.2004, and various line items.
- Top Right:** Inventar-Artikeldaten (Inventory Article Data) window showing a watch with ID 11111111111111111111111111111111, description 'Armbanduhr', and price information.
- Middle Left:** Jahresszeugnisse (Report Card) window for a student named Fitz. It includes sections for grades (Mathematics, English, etc.), awards, and a table of contents.
- Middle Center:** Teilnahme: 201006 Mustermeier Basile; 3: Basile. A registration form for a dance class with fields for Name, Adresse, and Kurs.
- Middle Right:** Selection mode dialog. It shows a table of student data and allows selecting rows, columns, or individual cells.
- Bottom Left:** Project management window showing tasks, budget, and progress for a project named 'Projekt 1'.
- Bottom Center:** Material properties dialog showing values for Querkontraktionszahl (Shear modulus), Längenausdehnungskoeffizient (Thermal expansion coefficient), and Dichte (Density).
- Bottom Right:** Another material properties dialog showing similar parameters for a different material.

TableSelectionDemo

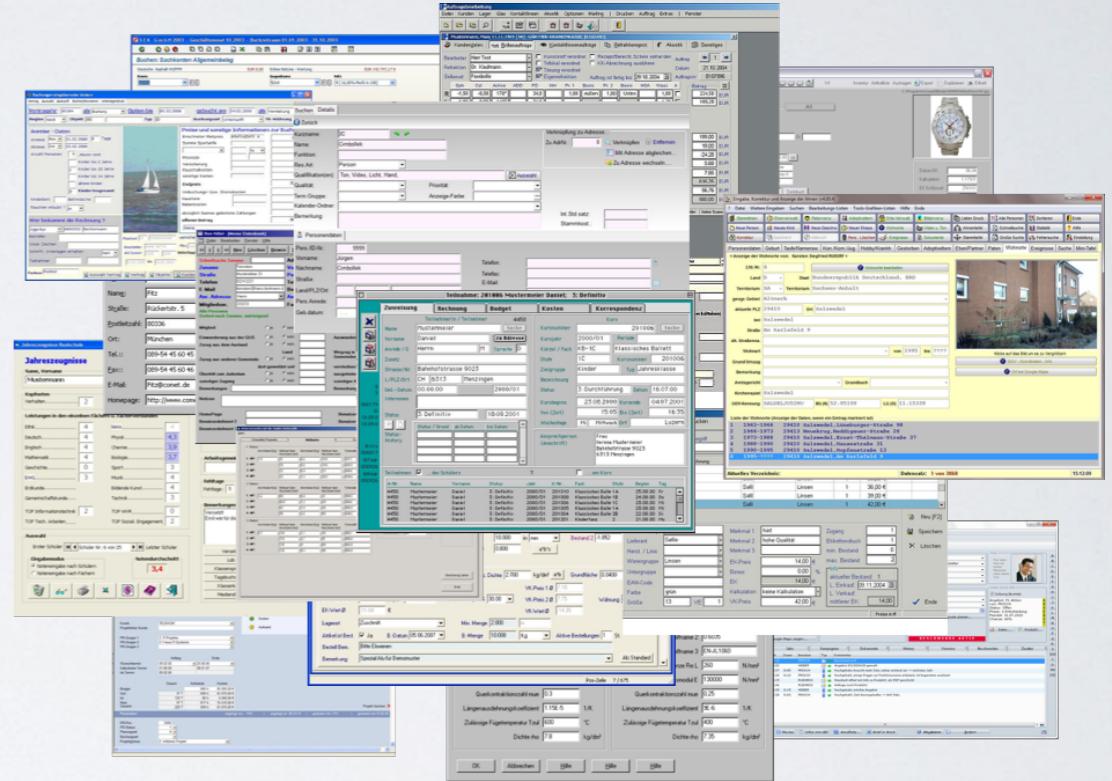
First Name	Last Name	Sport	# of Years	Vegetarian
Kathy	Smith	Snowboarding	5	<input type="checkbox"/>
John	Doe	Rowing	3	<input checked="" type="checkbox"/>
Sue	Black	Knitting	2	<input type="checkbox"/>
Jane	White	Speed reading	20	<input checked="" type="checkbox"/>
Bob	Green	Painting	10	<input type="checkbox"/>

Selection Mode

- Multiple Interval Selection
- Single Selection
- Single Interval Selection

Selection Options

- Row Selection
- Column Selection
- Cell Selection



# **Wozu testen wir?**



**Nach  
Implementierung?**



**Nach  
Änderung?**

file:///home/joshua/devel/cpp/cps111/bigcalc/apstring.cpp file:///home/joshua/devel/cpp/cps111/bigcalc/apstring.h

File   Difference   Settings   Help

**Viewing: apstring.cpp**

**Previous File**   **Next File**   **Files**

/home/joshua/devel/cpp/cps111/bigcalc/apstring

```
234 apstring operator + ( char ch, const apstring &
235 // postcondition: returns concatenation of ch +
236 {
237     apstring result; // make string equivalent
238     result = ch;
239     result += str;
240     return result;
241 }
242
243 apstring operator + ( const apstring & str, char
244 // postcondition: returns concatenation of str
245 {
246     apstring result(str);
247     result += ch;
248     return result;
249 }
250
251
252 apstring apstring::substr(int pos, int len) const
253 //description: extract and return the substrin
254 //                g at index pos
255 //precondition: this string represents c0, c1,
256 //                0 <= pos <= pos + len - 1 < n.
```

/home/joshua/devel/cpp/cps111/bigcalc/apstring.h

```
43
44 // indexing
45
46     char operator[ ]( int k ) const;
47     char & operator[ ]( int k );
48
49 // modifiers
50
51     const apstring & operator += ( const apstring & );
52     const apstring & operator += ( char ch );
53
54
55 private:
56     int myLength; // length
57     int myCapacity; // capacity
58     char * myCString; // storage
59 ];
60
61 // The following free (non-member) functions operate on apstring objects
62 //
63 // I/O functions
64
```



```
public void testSomething() throws Exception {  
    // here be some test code ...  
    UIElement element = driver.getElement("path/to/element");  
    assertEquals(null, element.getAction());  
    assertEquals("action", element.getActionCommand());  
    assertEquals(true, element.isEnabled());  
    assertEquals(true, element.isFocusable());  
    assertEquals("Lucida Grande", element.getFont().getName());  
    assertEquals(13, element.getFont().getSize());  
    assertEquals(0, element.getFont().getStyle());  
    assertEquals(23, element.getHeight());  
    assertEquals(null, element.getIcon());  
    assertEquals(0, element.getMnemonic());  
    assertEquals(null, element.getPressedIcon());  
    assertEquals(true, element.isSelected());  
    assertEquals("label", element.getText());  
    assertEquals(119, element.getWidth());  
    assertEquals(27, element.getX());  
    assertEquals(191, element.getY());  
}
```



# ReTest

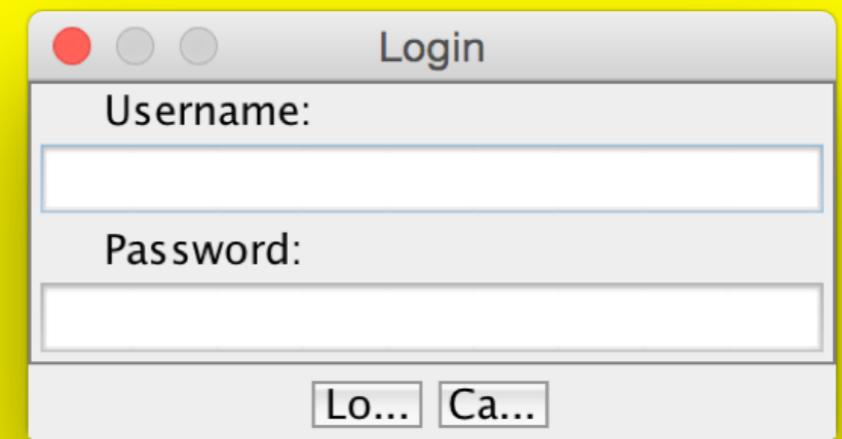
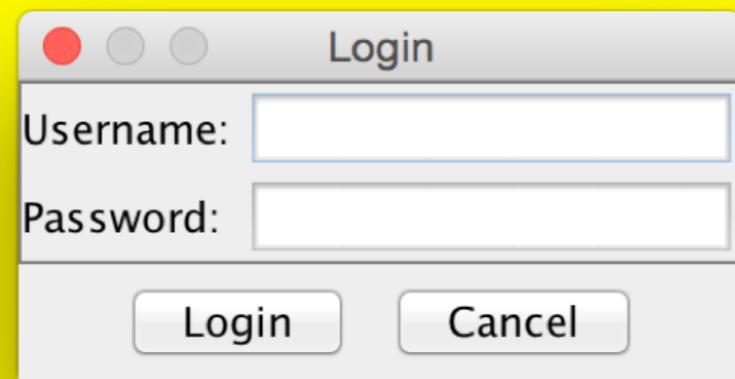
Ihr Vollautomatischer Regressionstester.  
Weniger Risiko, weniger Stress, weniger Kosten!

Haben Sie genug vom Testen?

Sowohl manuelles Testen, als auch das manuelle Erstellen und Pflegen von automatischen Tests sind unbeliebte Zeitfresser. Direkt aus der Forschung ist ReTest als bisher einziges Produkt seiner Art Ihr



LoginDialog



Label Username

Username:

x-Coordinate: 0  
y-Coordinate: 5

Username:

x-Coordinate: 24  
y-Coordinate: 0

Field Username



x-Coordinate: 71  
y-Coordinate: 1  
Width: 254px



x-Coordinate: 1  
y-Coordinate: 23  
Width: 543px

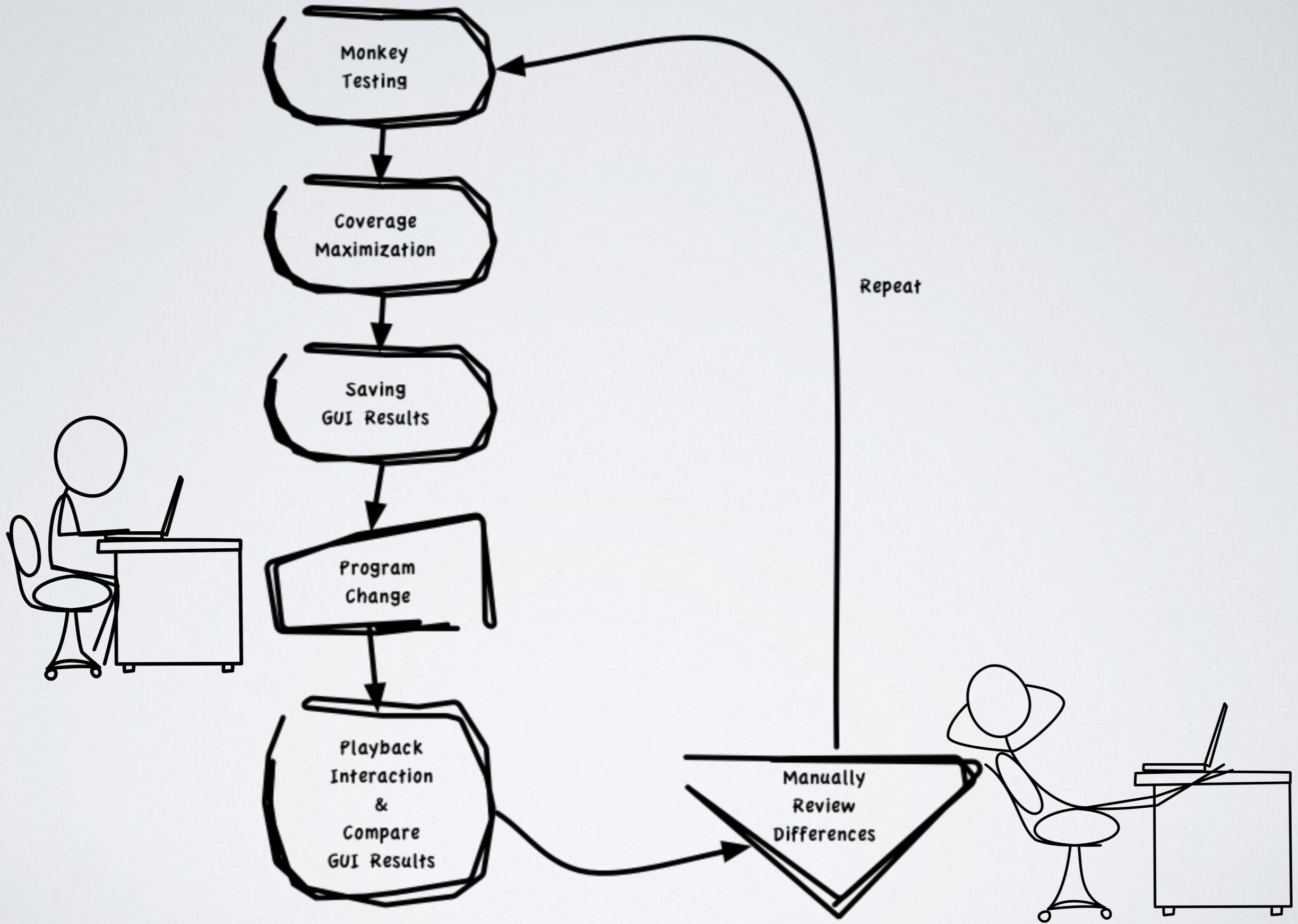
Label Password

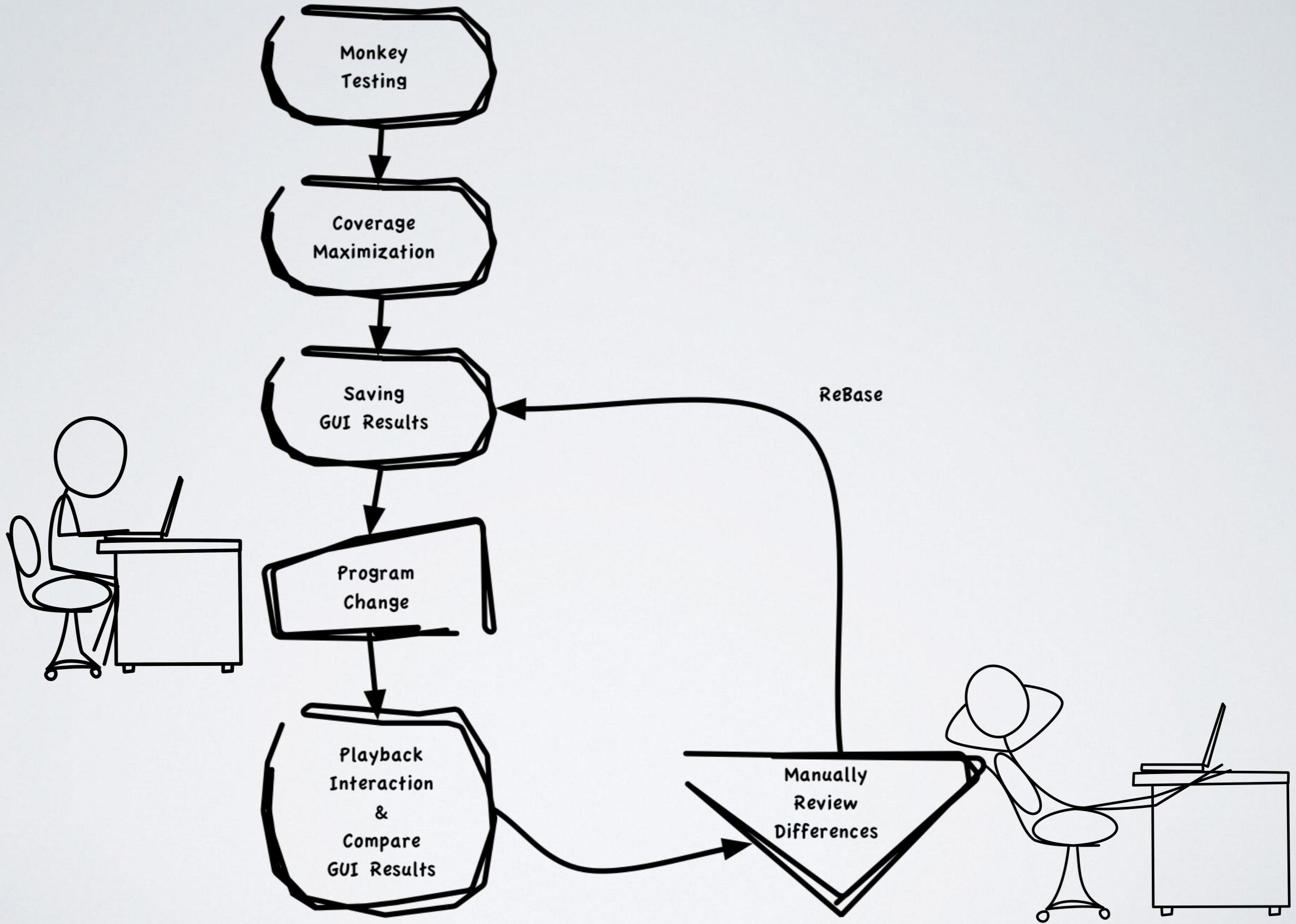
Password:

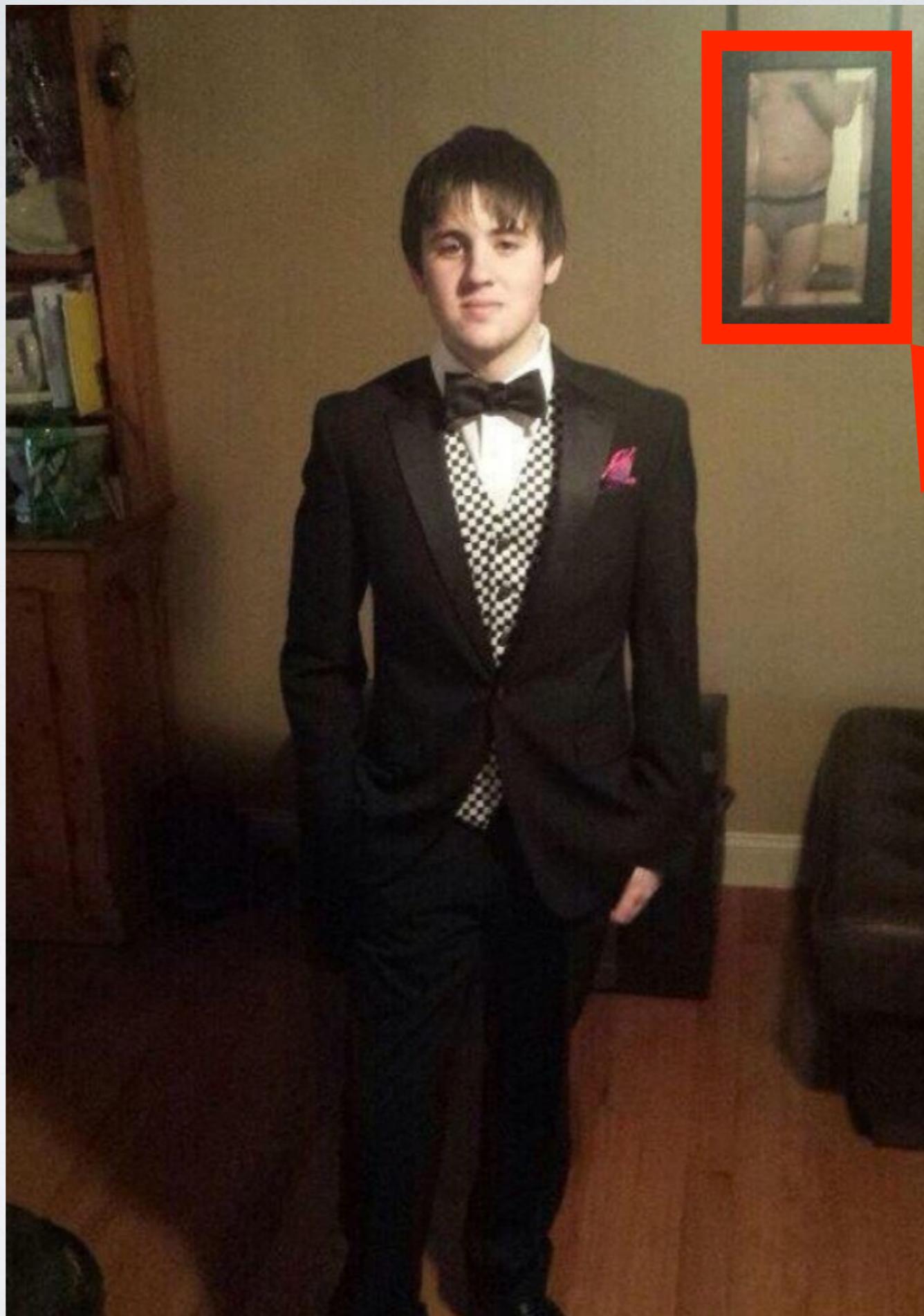
x-Coordinate: 0  
y-Coordinate: 44

Password:

x-Coordinate: 24  
y-Coordinate: 48







**Ausführung Wiederholbar?**

**Object.hashCode()**

TableSelection

First Name	Last Name	Sport
Kathy	Smith	Snowboarding
John	Doe	Rowing
Sue	Black	Knitting
Jane	White	Speed reading
Tom	Perry	Paint

Selection Mode

- Multiple Interval Selection
- Single Selection
- Single Interval Selection

Selection Options

- Row Selection
- Column Selection
- Cell Selection

TableSelection

First Name	Last Name	Sport
Kathy	Smith	Snowboarding
Sue	Black	Knitting
John	Doe	Rowing
Jane	White	Speed reading
Tom	Perry	Paint

Selection Mode

- Multiple Interval Selection
- Single Selection
- Single Interval Selection

Selection Options

- Row Selection
- Column Selection
- Cell Selection

**System.currentTimeMillis()**



# **Thread-Schedules**

# **Synchronization**

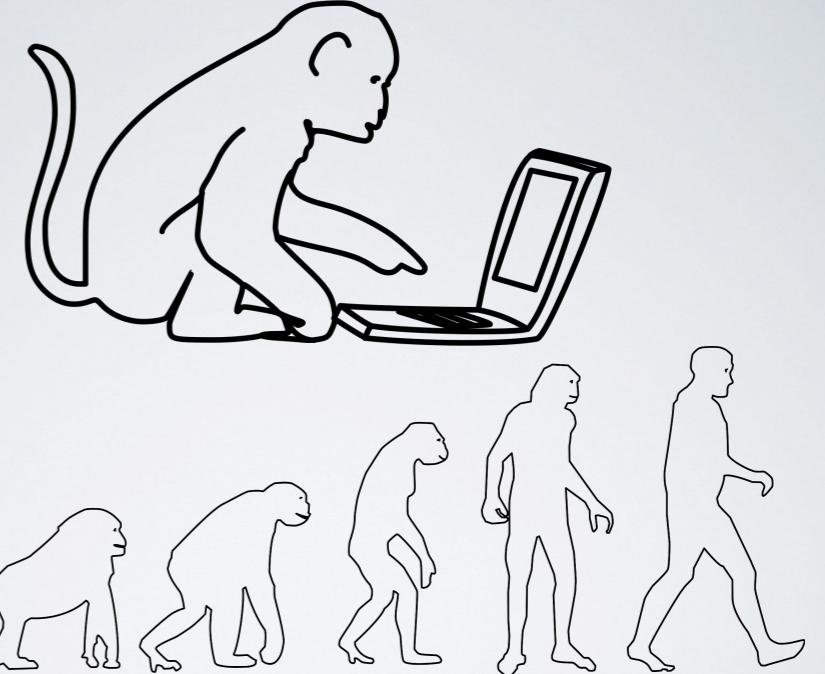
1. `WebDriverWait.until(condition-that-finds-the-element)`

**System.exit()**

# **Ihre Daten**

# **Testbed**

Intelligenter Affe

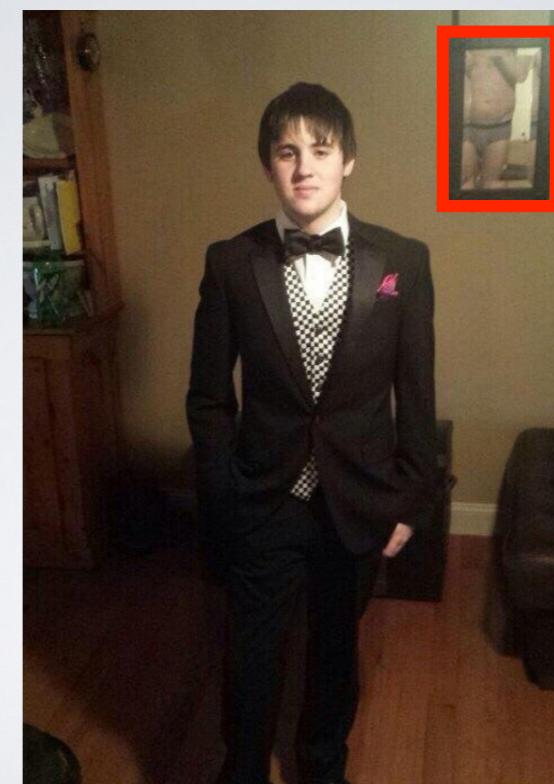


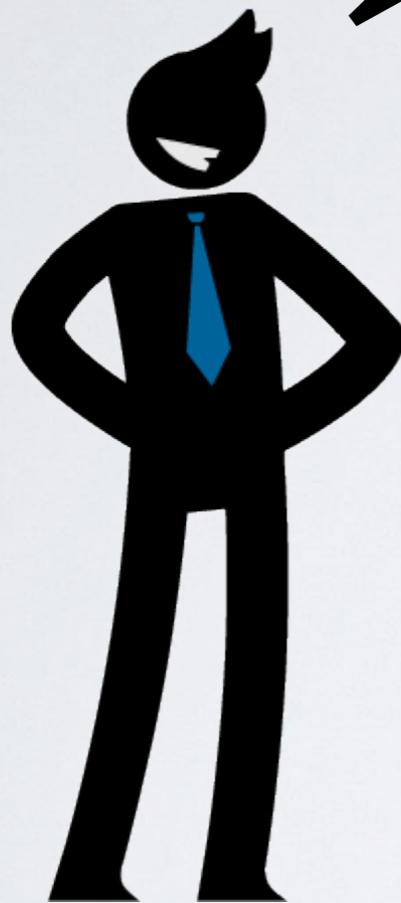
Material und Infos auf: [www.retest.de!](http://www.retest.de)

When is a bug not a bug?



When it's a feature!





Machen Sie den Test!



[www.retest.de](http://www.retest.de)