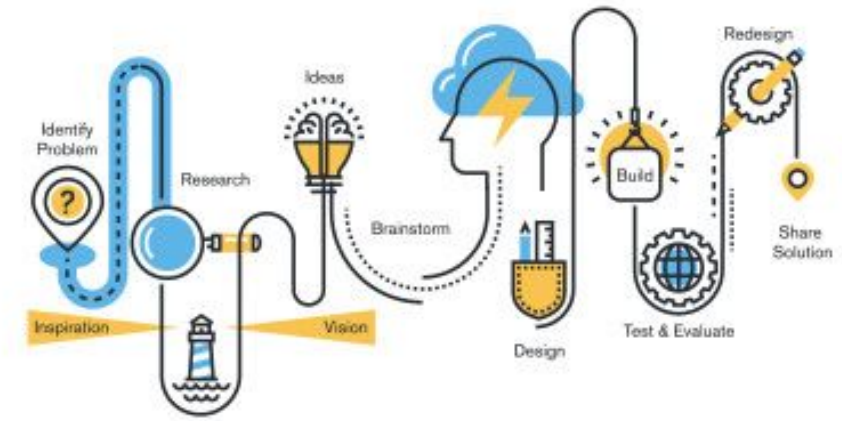
# 3. Einführung in Software Engineering



# Die Katze auf der Terrasse

- Mit **Python-Objekten** ist es wie mir der **Katze**,
  - die du irgendwann schlafend auf deiner Terrasse vorfindest.
- Ganz wie ein **Python-Objekten** kann dir die **Katze** nicht sagen, wie sie heißt
  - – **es ist ihr auch ganz egal.**
- Um ihren Namen herauszufinden,
  - wirst du wohl deine Nachbarn fragen müssen,
- und **du** solltest **nicht überrascht** sein,
  - wenn du herausbekommst, dass **die Katze viele Namen hat**.

# ...es **kann** nützlich sein


## Labor 3+

# SE != Engineering


## Engineering

## Software

# SE != Engineering


## Engineering

- Produkt ist ein physikalisches Objekt

## Software

- Produkt ist ein laufendes Programm

# SE != Engineering


## Engineering

- Produkt ist ein physikalisches Objekt
- Gebaut durch Menschen und Werkzeuge

## Software

- Produkt ist ein laufendes Programm
- Gebaut durch Menschen und Werkzeuge

# SE != Engineering


## Engineering

- Produkt ist ein physikalisches Objekt
- Gebaut durch Menschen und Werkzeuge
- Konstruktion
  - ist teuer

## Software

- Produkt ist ein laufendes Programm
- Gebaut durch Menschen und Werkzeuge
- Konstruktion
  - ist billig

# SE != Engineering


## Engineering

- Produkt ist ein physikalisches Objekt
- Gebaut durch Menschen und Werkzeuge
- Konstruktion
  - ist teuer
  - erfordert Arbeit und Material

## Software

- Produkt ist ein laufendes Programm
- Gebaut durch Menschen und Werkzeuge
- Konstruktion
  - ist billig
  - automatisch

# SE != Engineering


## Engineering

- Produkt ist ein physikalisches Objekt
- Gebaut durch Menschen und Werkzeuge
- Konstruktion
  - ist teuer
  - erfordert Arbeit und Material
  - ist langsam

## Software

- Produkt ist ein laufendes Programm
- Gebaut durch Menschen und Werkzeuge
- Konstruktion
  - ist billig
  - automatisch
  - ist schnell

# SE != Engineering

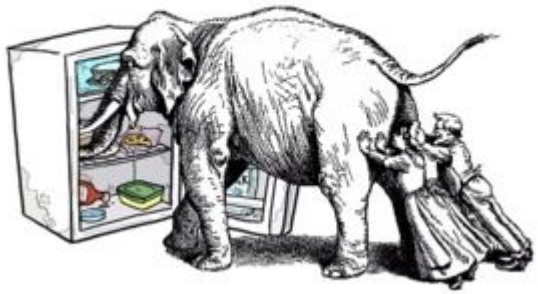
## Engineering

- Produkt ist ein physikalisches Objekt
- Gebaut durch Menschen und Werkzeuge
- Konstruktion
  - ist teuer
  - erfordert Arbeit und Material
  - ist langsam
  - teuer zu wiederholen

## Software

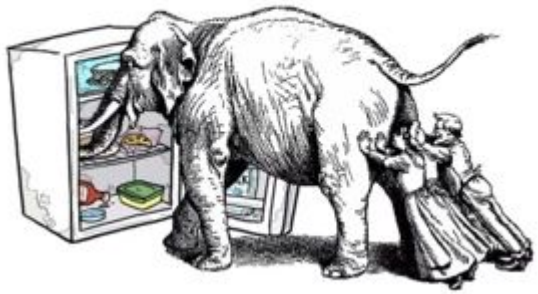
- Produkt ist ein laufendes Programm
- Gebaut durch Menschen und Werkzeuge
- Konstruktion
  - ist billig
  - automatisch
  - ist schnell
  - leicht zu wiederholen

# Wie steckt man einen Elefant in einen  Kühlschrank?



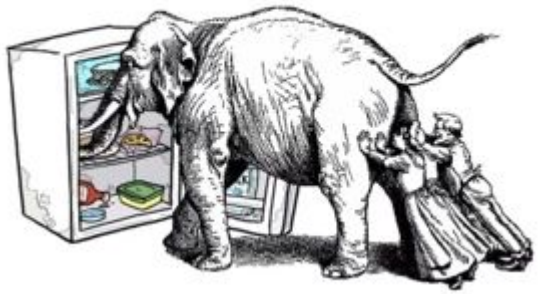
# Wie steckt man einen Elefant in einen  Kühlschrank?

1. Man öffnet den Kühlschrank



# Wie steckt man einen Elefant in einen  Kühlschrank?

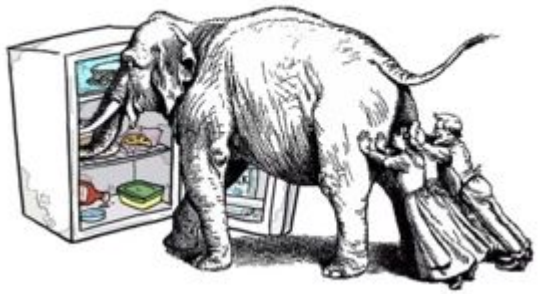
1. Man öffnet den Kühlschrank
2. stellt den Elefanten hinein



# Wie steckt man einen Elefant in einen  Kühlschrank?

1. Man öffnet den Kühlschrank
2. stellt den Elefanten hinein
3. schließt die Tür

**...aber**



# wie denkt ein Entwickler?

# wie denkt ein Entwickler?

1. teile das Problem in **mehrere Probleme** auf

# wie denkt ein Entwickler?

1. teile das Problem in **mehrere Probleme** auf
2. finde **Lösungen** für die kleine Probleme

# wie denkt ein Entwickler?

1. teile das Problem in **mehrere Probleme** auf
2. finde **Lösungen** für die kleine Probleme
3. stelle die Lösungen zusammen

# wie denkt ein Entwickler?

1. teile das Problem in **mehrere Probleme** auf
2. finde **Lösungen** für die kleine Probleme
3. stelle die Lösungen zusammen
4. Aufräumen (**refactor**)

# teile das Problem auf

1. Was für einen Kühlschrank habe ich?
2. Aber der Elefant?
3. Wo finde ich den Elefant?
4. Transport
5. Was könnte ich machen, falls der Elefant zu groß ist?

# finde Lösungen

# finde Lösungen

1. Was für einen Kühlschrank nutze ich?

# finde Lösungen


1. Was für einen Kühlschrank nutze ich? - Mein

# finde Lösungen

1. Was für einen Kühlschrank nutze ich? - Mein
2. Aber der Elefant?

# finde Lösungen

1. Was für einen Kühlschrank nutze ich? - Mein
2. Aber der Elefant? - Afrikanischer

# finde Lösungen

1. Was für einen Kühlschrank nutze ich? - Mein
2. Aber der Elefant? - Afrikanischer
3. Wo finde ich den Elefant?

# finde Lösungen

1. Was für einen Kühlschrank nutze ich? - Mein
2. Aber der Elefant? - Afrikanischer
3. Wo finde ich den Elefant? - Afrika

# finde Lösungen

1. Was für einen Kühlschrank nutze ich? - Mein
2. Aber der Elefant? - Afrikanischer
3. Wo finde ich den Elefant? - Afrika
4. Transport

# finde Lösungen

1. Was für einen Kühlschrank nutze ich? - Mein
2. Aber der Elefant? - Afrikanischer
3. Wo finde ich den Elefant? - Afrika
4. Transport - mit dem Flugzeug, im Gepäck

# finde Lösungen

1. Was für einen Kühlschrank nutze ich? - Mein
2. Aber der Elefant? - Afrikanischer
3. Wo finde ich den Elefant? - Afrika
4. Transport - mit dem Flugzeug, im Gepäck
5. Was könnte ich machen, wenn der Elefant zu groß ist?

# stelle die Lösungen zusammen

1. ich leihe ein shrinkgun von Gru
2. fliege nach Südafrika
3. besichtige einen Elephant-Park
4. finde einen Elefant im Park
5. schieße den Elefant mit dem shrinkgun
6. lege den Elefant ins Gepäck
7. fahre zum Flughafen
8. fliege zurück
9. fahre nach Hause
10. stecke den Elefant in den Kühlschrank

# und für Programmierung….

Wir brauchen eine Funktion, das die Anzahl von Erscheinungen aller Elemente in einer Liste bestimmt.
Beispiel:

- input: `l = [1,2,6,5,3,4,2,4,1]`
- output: `1 - 2, 2 - 2, 3 -1, 4 - 2, 5 - 1`

Fragen:

- wie kann man das output representieren?

Schritte:

- man muss das Output initialiseren
- man muss alle Elemente der Liste durchgehen
- für ein Element man bestimmt die Anzahl von Erscheinungen
- man fugt die anzahl in Ouput
- man gibt das Ouput zurück

# und für Programmierung….

Fragen:

- wie kann man das output representieren?
  - Dictionary

Schritte:

- man muss das Output initialiseren
  - `d = {}`
- man muss alle Elemente der Liste durchgehen
  - `for elem in l:`
- für ein Element man bestimmt die Anzahl von Erscheinungen
  - neue Funktion: `anzahl()`
- man fugt die anzahl in Ouput
  - `d['elem'] = a`
- man gibt das Ouput zurück
  - `return d`

# und für Programmierung….

```
l = [1,2,6,5,3,4,2,4,1]

def my_funk(l):
    d = {}

    for elem in l:
        a = anzahl(elem,l):
        d[‘elem’] = a

    return d
```

# und für Programmierung….

jetzt muss man das Gleiche für die Anzahl Funktion machen

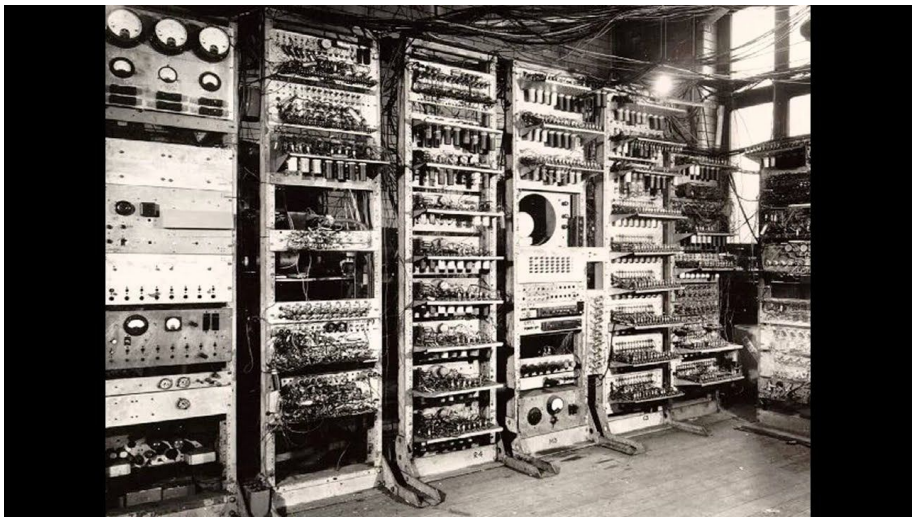
```
def anzahl(el, l):
    a = 0

    for elem in l:
        if el == elem:
            a += 1

    return a
```

# Vorgehen in SE

- Wie alle Aktivitäten, die die Zusammenarbeit zwischen Menschen beinhalten
  - Programmierung ist nicht einfach
- Woher kommt die Problembeschreibung?
- Beschreibt sie wirklich das Problem des Nutzers?
- Welche Struktur soll das fertige Programm haben?
- Löst das Programm wirklich das Problem?

# a long long time ago….

- einmal war es einfach
- die Welt war irgendwie größer
- Computer waren groß, langsam und Anwendungsfälle waren äußerst begrenzt



# Software-Krise

- Mitte 1960er Jahre
- Hardware ist schneller geworden
  - Mit schnellerer Hardware wurde Software wichtiger
  - Als Konsequenz Anforderungen stiegen
- Die einfache Situation von gestern wurde schwer zu managen
- The Humble Programmer
  - https://cacm.acm.org/magazines/1972/10/11993-the-humble-programmer/pdf

The major cause of the software crisis is that the machines have become several orders of magnitude more powerful! To put it quite bluntly: as long as there were no machines, programming was no problem at all; when we had a few weak computers, programming became a mild problem, and now we have gigantic computers, programming has become an equally gigantic problem.

— Edsger Dijkstra, The Humble Programmer (EWD340), Communications of the ACM

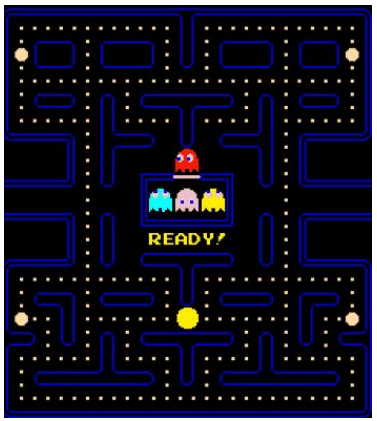
# Warum hat Hardware das verursacht?

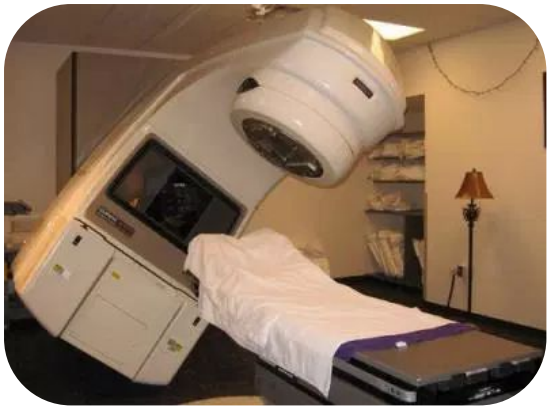
```
def main():
     while true:
          …
```

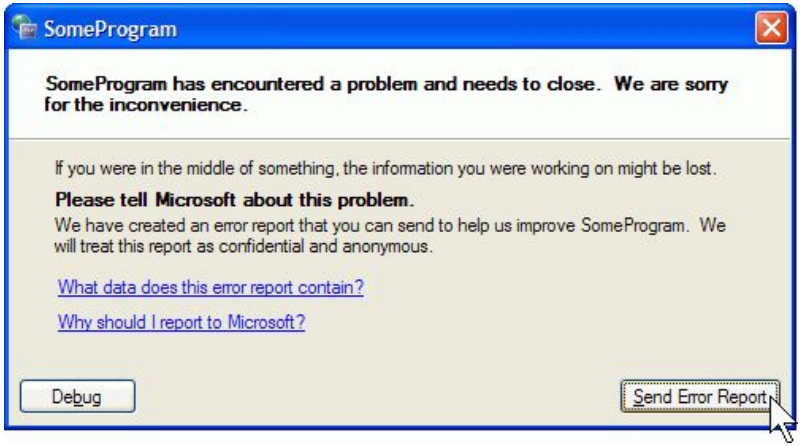


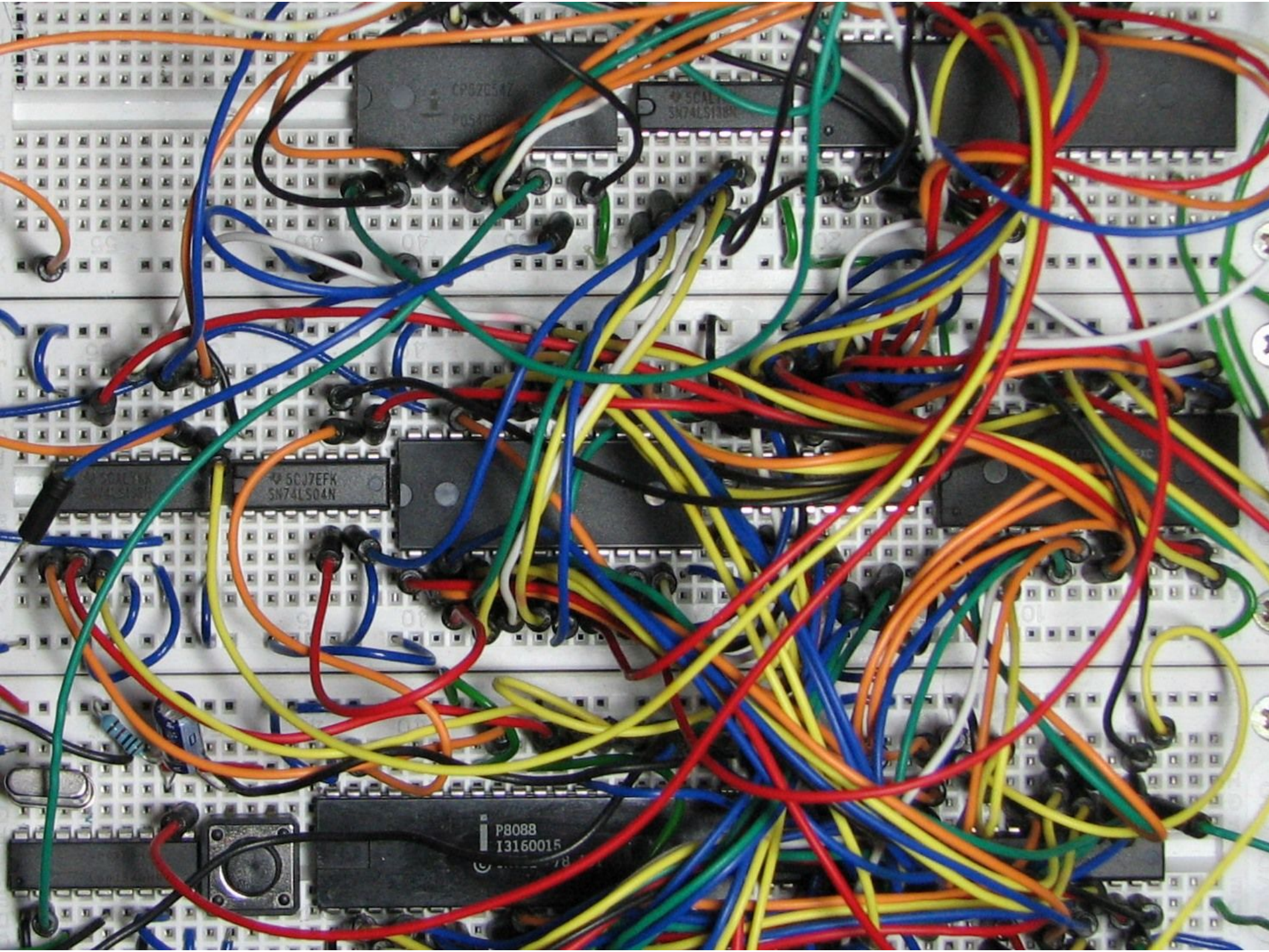
```
def main():
     game_running=true
     while game_running:
          process_user_input()
          game_world()
          draw_game_world()

          #close game
          if Pressed(KEY_ESCAPE):
               game_running = false
```








SomeProgram
SomeProgram has encountered a problem and needs to close. We are sorry for the inconvenience.
If you were in the middle of something, the information you were working on might be lost.
Please tell Microsoft about this problem.
We have created an error report that you can send to help us improve SomeProgram. We will treat this report as confidential and anonymous.
What data does this error report contain?
Why should I report to Microsoft?
Debug
Send Error Report


5CJ7EFK
SN74LS04N
P8088
I3160015

- Nicht nur Hardware ist schneller geworden, sondern wir leben in einer vernetzten Welt
- Heute läuft Software auf Handys, Computern, Smartwatches und Haushaltsgeräten
- und beeinflusst alle Aspekte unseres Lebens

# Extreme Komplexität


## US DDX Submarine

- Viele Systeme
- Zusammen **30 Milliarde** Zeilen Code
- In **100+** Programmiersprachen
- Kosten: **$15-30/Codezeile**

# Woran liegt es?

- unzureichend spezifizierte Anforderungen
- häufiges Ändern der Anforderungen während des Projekts
- inkompetente Mitarbeiter
- fehlende Unterstützung durch das Management
- zu große Erwartungen
- falsche Schätzung der Zeit/Kosten
- Managementfehler

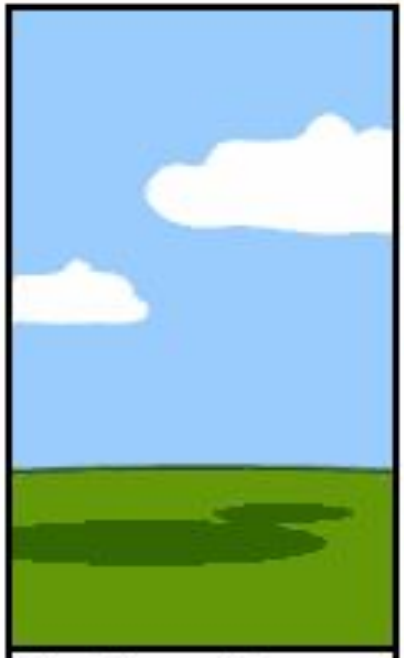
How the customer explained it

How the Project Leader understood it

How the Analyst designed it

How the Programmer wrote it

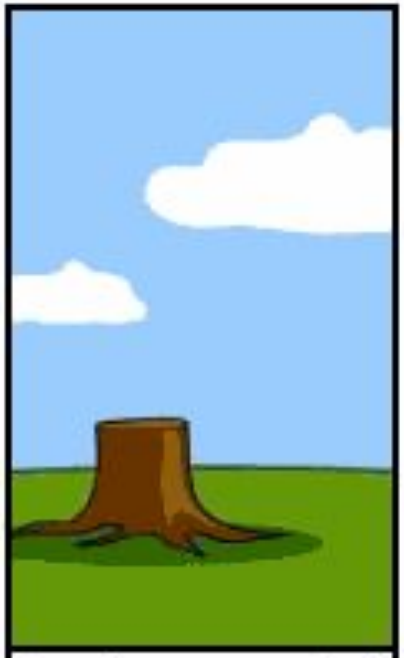
How the Business Consultant described it



How the project was documented

What operations installed

How the customer was billed



How it was supported

What the customer really needed

# Rollen

- Der Entwickler/Architekt
  - schreibt den Code
- Fachberater
  - der die Spezifikation schreibt
- Tester
  - stellt die Qualität sicher
- Stakeholder
  - hat Interesse in dem Projekt/Produkt
- Endanwender
  - nutzt das Program

# Stakeholder


## Einzelpersonen und Organisationen

- die aktiv an einem Projekt beteiligt sind
- deren Interessen als Folge der Projektdurchführung oder des Projektabschlusses positiv oder negativ beeinflusst werden können
- die das Projekt und seine Ergebnisse beeinflussen

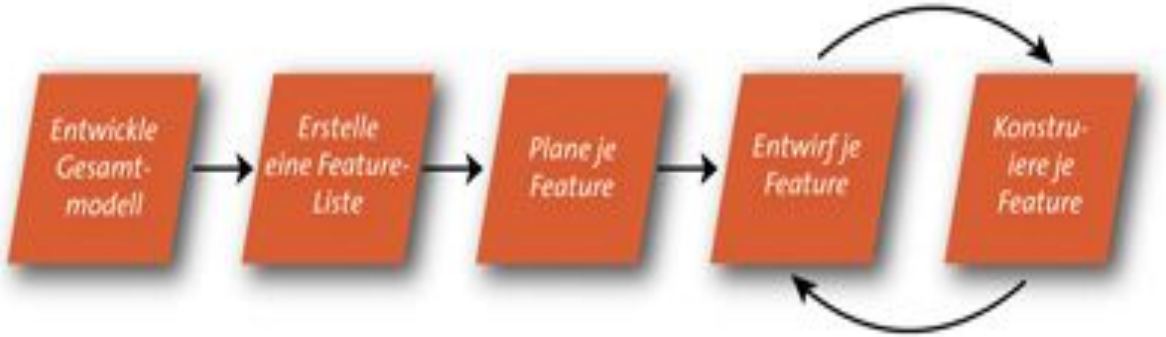
# Softwaretechnik

## Teilgebiet der Informatik

- Anforderungsanalyse
- Entwurf und Entwicklung von Software
- Organisation und Strukturierung der Entwicklung
- Projektmanagement
- Qualitätssicherung
- Betrieb und Wartung von Systemen

# Softwaretechnik

- systematische **Verwendung** von
- **Prinzipien**, **Methoden** und **Werkzeugen**
- für die **Entwicklung** und Anwendung von umfangreichen **Softwaresystemen**



# Softwaretechnik

- **Problemstellung (Idee)**
  - Eine Beschereibung eines Problems
  - Ein Lehrer braucht ein programm für Studenten, die Rationale Zahlen lernen möchten.

- **Anforderungen (was?)**
  - Genaue Festlegung des Umfanges des geplanten Systems
  - beschreiben die Funktionalität des Produktes

# Probleme

- Kunden wissen nicht was sie wirklich wollen
- Kunden benutzen ihre eigene Fachsprache
- Verschiedene Stakeholder können widersprüchliche Anforderungen haben
- Politische und organisatorische Faktoren können Anforderungen beeinflussen
- Anforderungen ändern sich während der Entwicklung
- Neue Stakeholder mischen sich ein

# Anforderungen

- **Vollständig**
  - Alle Anforderungen des Kunden müssen explizit beschrieben sein
- **Atomar**
  - Es darf nur eine Anforderung pro Abschnitt beschrieben sein
- **Identifizierbar**
  - Jede Anforderung muss eindeutig identifizierbar sein
- **Nachprüfbar**
  - Die Anforderungen sollten mit Abnahmekriterien verknüpft werden, damit bei der Abnahme geprüft werden kann, ob die Anforderungen erfüllt wurden
- **Konsistent**
  - Die definierten Anforderungen sind untereinander widerspruchsfrei.

# Features

- Features sind kleine Funktionen, die Wert für den Kunde haben

- Features werden nach dem einfachen Schema erstellt
  <Aktion> <Ergebnis> <Objekt>
  - Aktion = eine Funktion, welche die Anwendung bereitstellt
  - Ergebnis = das Ergebnis der Ausführung der Funktion
  - Objekt = Die Entität, die die Funktion umsetzt

- Features können in wenigen Studen umgesetzt werden

# Features


## Problemstellung (Idee)

Ein Lehrer braucht ein programm für Studenten, die Rationale Zahlen lernen möchten.

## Rechner (Feature-Liste)

F1. Eine Zahl einfügen
F2. Rechner löschen
F3. Die letzte Änderung rückgängig machen

# Iteration

- Eine Iteration ist ein festgelegter Zeitraum innerhalb eines Projekts, in dem man eine stabile Version des Produkts zusammen mit Dokumentation erstellt

- Eine Iteration führt zu einem funktionierenden und nützlichen Programm für den Kunde

- Ablauf
  - man setzt in einer Iteration einige Funktionalitäten um
  - man zeigt die Ergebnisse (das Ouput) an
  - man bekommt Feeback

# Feature Driven Development


- man muss eine Feature-Liste erstellen
- man muss die Iterations plannen
- Für jeden Iteration:
  - welche Features werden in der Iteration umgesetzt
  - Implementation und Testen
- Beispiel
  - Iteration 1
    - F1. Eine Zahl einfügen
    - F2. Rechner löschen
  - Iteration 2
    - F3. Die letzte Änderung rückgängig machen

# Task Breakdown für Iteration 1 (Add, Löschen)

- zu Beginn einer Iteration muss man verstehen, was implementiert werden soll

- man muss dann die Features analysieren und die Arbeit in Aufgaben (Tasks) aufteilen

- T1. Berechnung den größten gemeinsamen Teiler
- T2. Addition zweier rationaler Zahlen
- T3. Rechner: Zahl einfügen
- T4  Rechner: Summe ausgeben
- T5. Rechner: Summe löschen
- T6. Benutzerschnittstelle. Menü für: Init, Add, Del

- **Aufgabenabhängigkeit** : T6 -> T5 -> T3 -> T2 -> T1

# Testfall

- eine Reihe von Eingabewerten und erwarteten Ergebnisse, um eine bestimmte Funktion eines Programms zu testen

| **Input: a, b** | c=ggT(a,b) |
|---|---|
| 2 3 | 1 |
| 2 4 | 2 |
| 6 4 | 2 |
| 0 2 | 2 |
| 2 0 | 2 |
| 24 9 | 3 |
| -2 0 | Fehler |
| 0 -2 | Fehler |

# Labor 3+


- Man muss Konsoleanwendungen implementieren
- Code muss in Funktionen unterteilt werden
- Jede Funktion muss nur genau ein Ding tun
- Funktionen führen entweder Input/Output Operationen oder Berechnungen durch, aber nicht beides!
- Non-UI-Funktionen müssen spezifiziert werden