

# Nicht-Triviale Software Entwicklung

Robert Kleinschmager | [github.com/barclay-reg](https://github.com/barclay-reg)

# WIESO NICHT-TRIVIAL?

# Coding

Coding

Testing

Design

Coding

Testing

Design

Coding

Testing

Documentation

Analysis

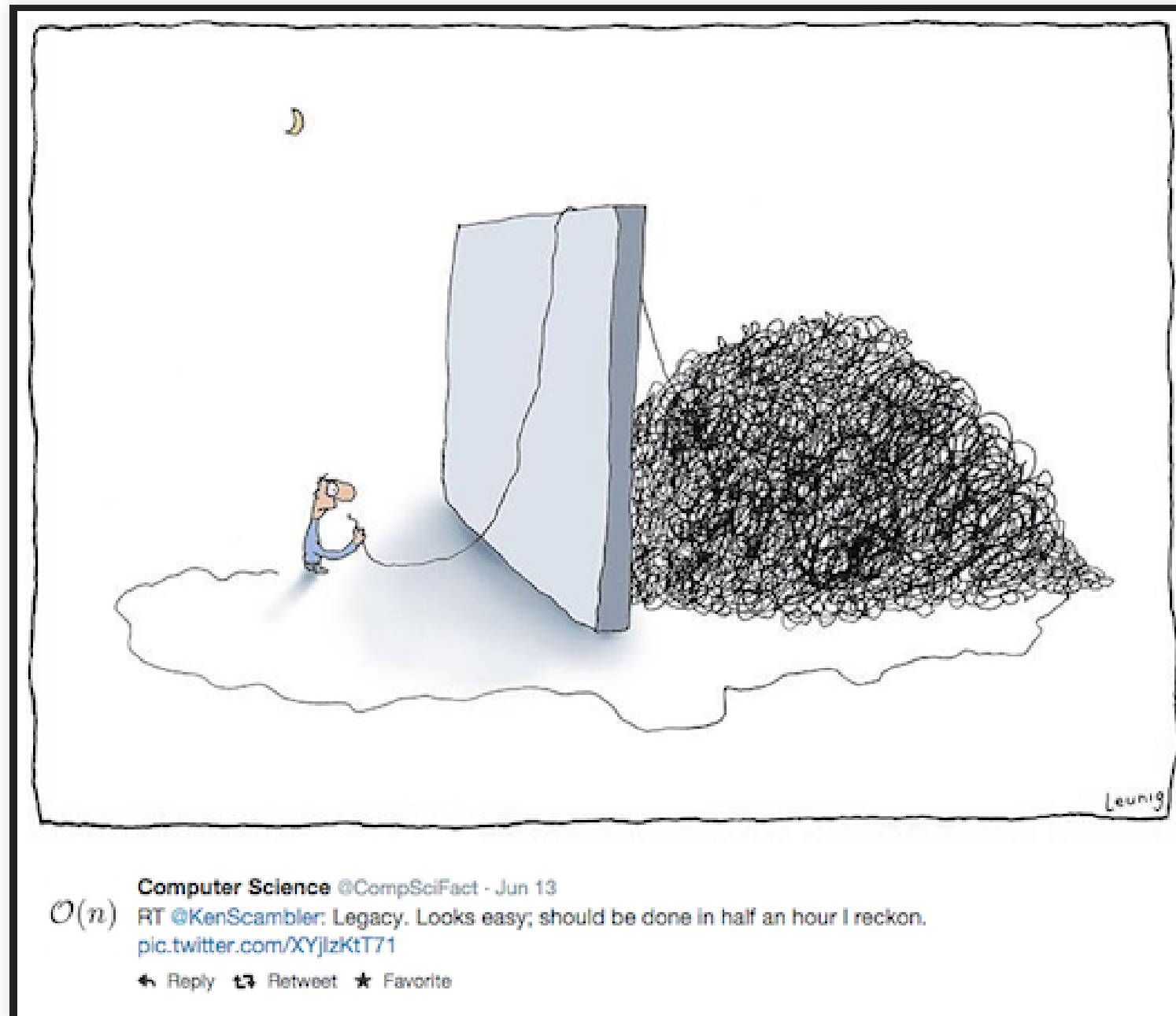
Design

Coding

Testing

Documentation

# LEGACY CODE



# LEGACY CODE - UMFRAGE

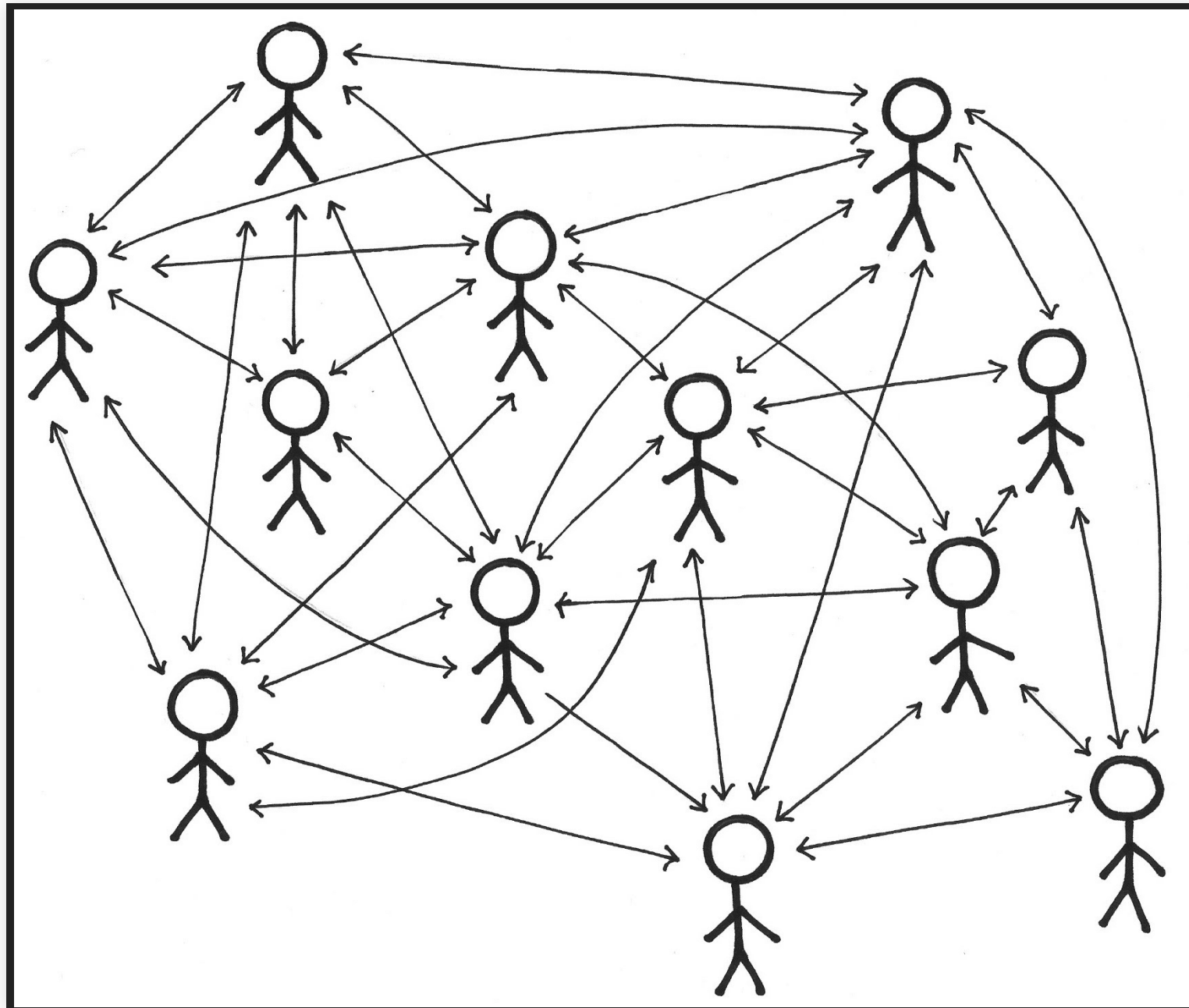
**In your opinion, what makes code «legacy»?**



# RAHMENBEDINGUNGEN

- Funktionale Requirements
- Nicht-Funktionale Requirements
- (alte) Werkzeuge
  - IDE
  - Bibliotheken

# TEAM



# KONSEQUENZEN



# KOMPLIZIERT

- das Maß an Ungewissheit
- Wissen fehlt
- Kompliziertes System
  - zuverlässig, exakt
  - von außen steuerbar
  - konstantes Verhalten

# KOMPLEX

- das Maß der Menge an Überraschungen
- Kontext-abhängig
- Komplexes System
  - "lebendig", erzeugt Überraschungen
  - von außen nur beobachtbar
  - dynamisches Verhalten



- Kompliziert oder Komplex?





• Kompliziert





- Kompliziert oder Komplex?





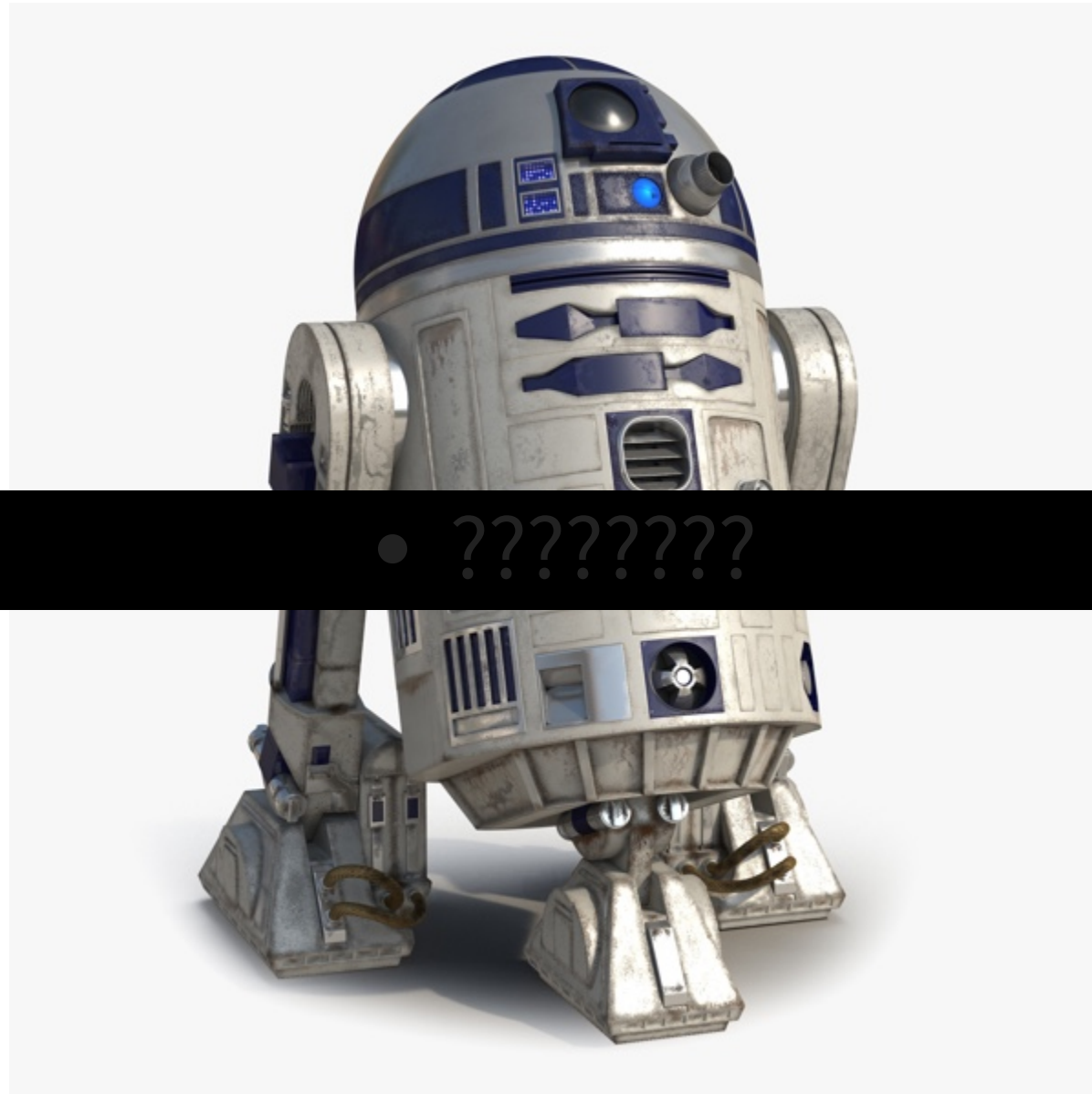


- Komplex





- Kompliziert oder Komplex?

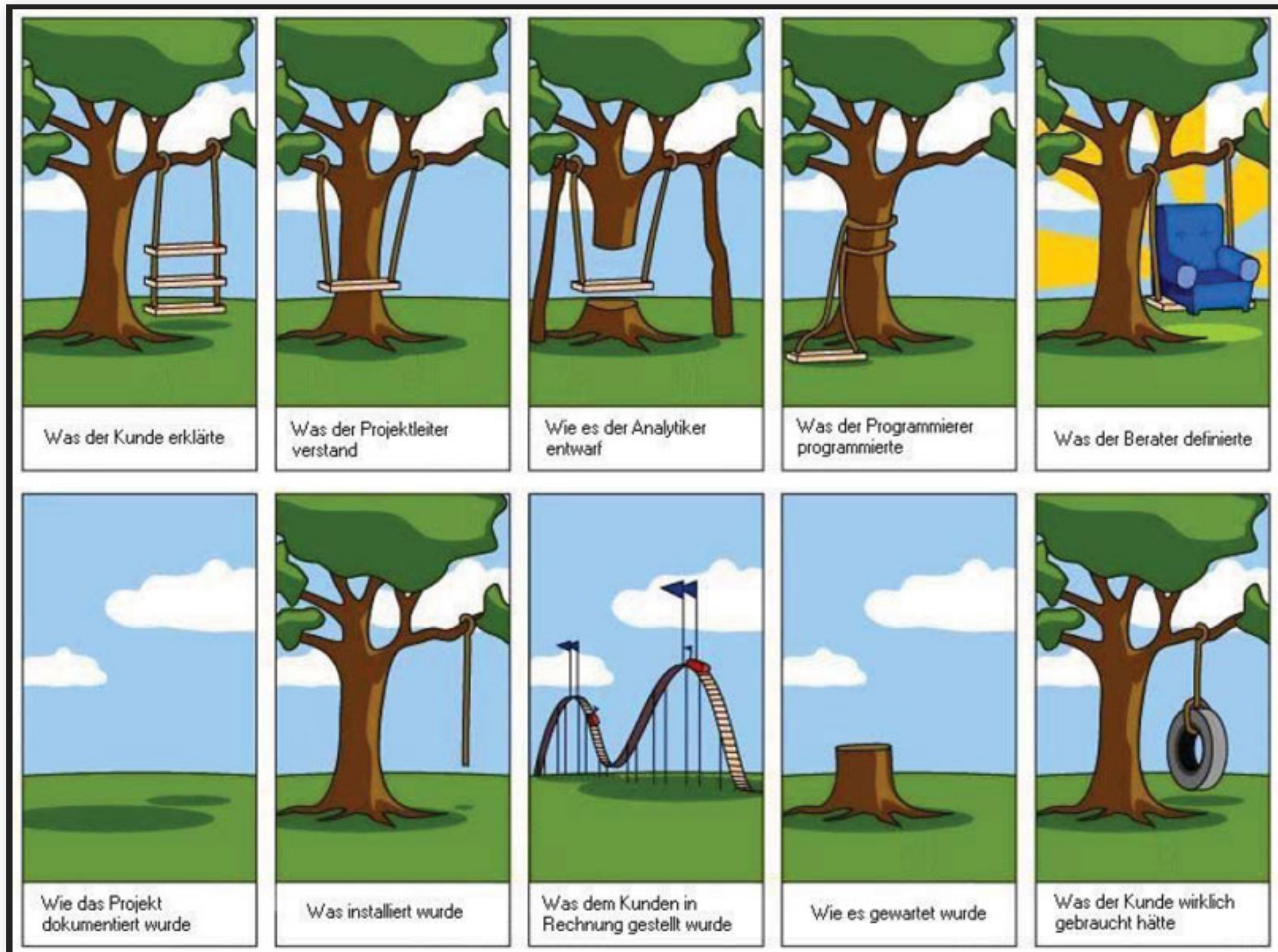


• ????????

# Software wächst in der "freien Wildbahn" in einem Spannungsfeld







- Coding **wird** zunehmen schwerer

## ÜBERGANG VON KOMPLIZIERT ZU KOMPLEX

- Schätzungen **werden** ungenauer
- Analysen **werden** umfangreicher
- Test **werden** aufwändiger, vielfältiger
- Dokumentation **wird** umfassender
- Technische Schulden **entstehen**

# KOMPLEXITÄT IN DER SOFTWAREENTWICKLUNG

*software systems grow faster in size  
and complexity than methods to  
handle complexity are invented*

— Wirth's Law

*Programming is still a stone-age crafts*

# GEGENMASSNAHMEN

# CLEAN CODE

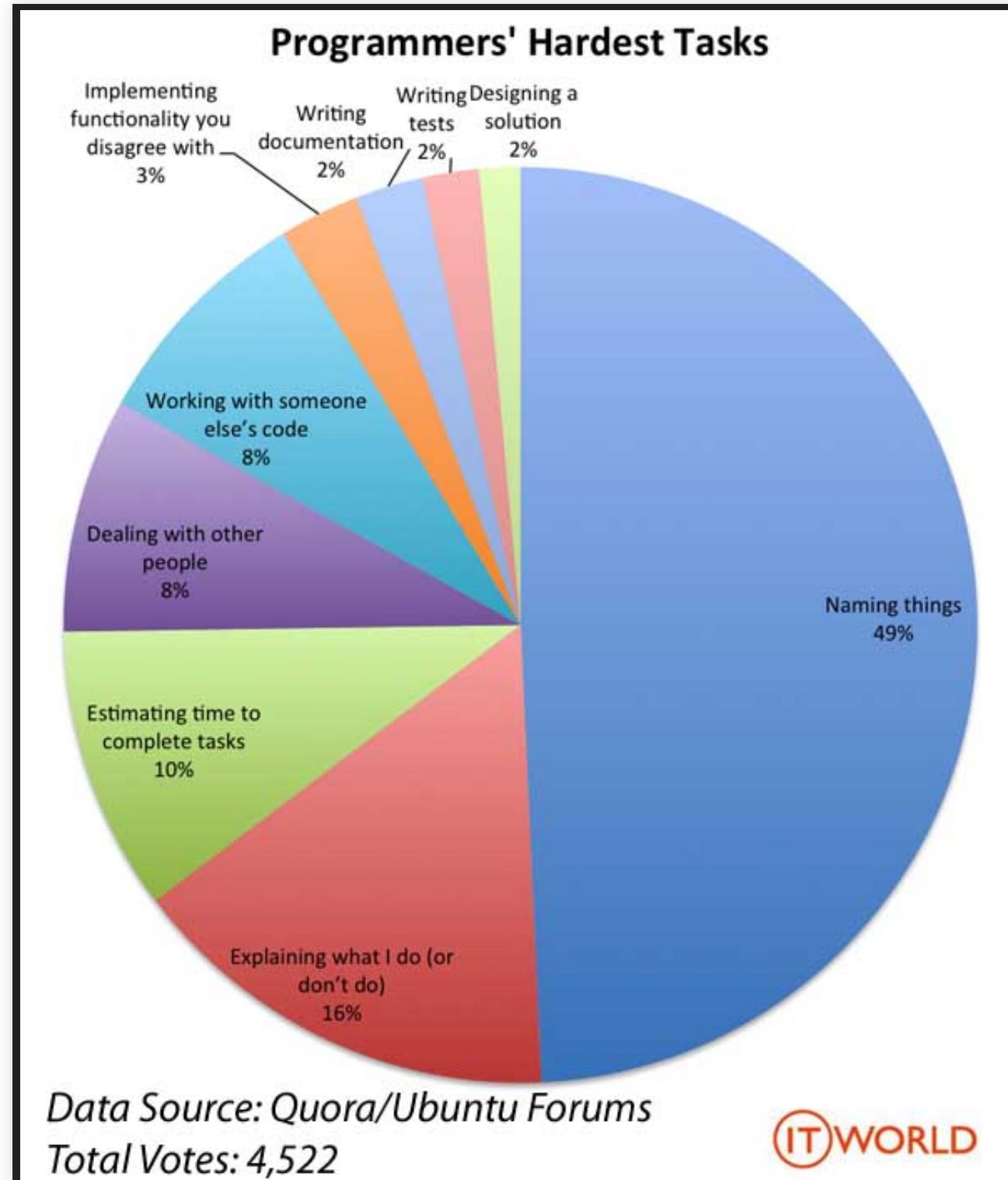
- Code wird nur einmal **geschrieben**, aber viele male **gelesen**
- Keine Code-Magneten schreiben;
  - Single Responsibility Principle (Theorie)
  - Main Purpose Principle (Praxis)
- **Komposition** over **Inheritance**

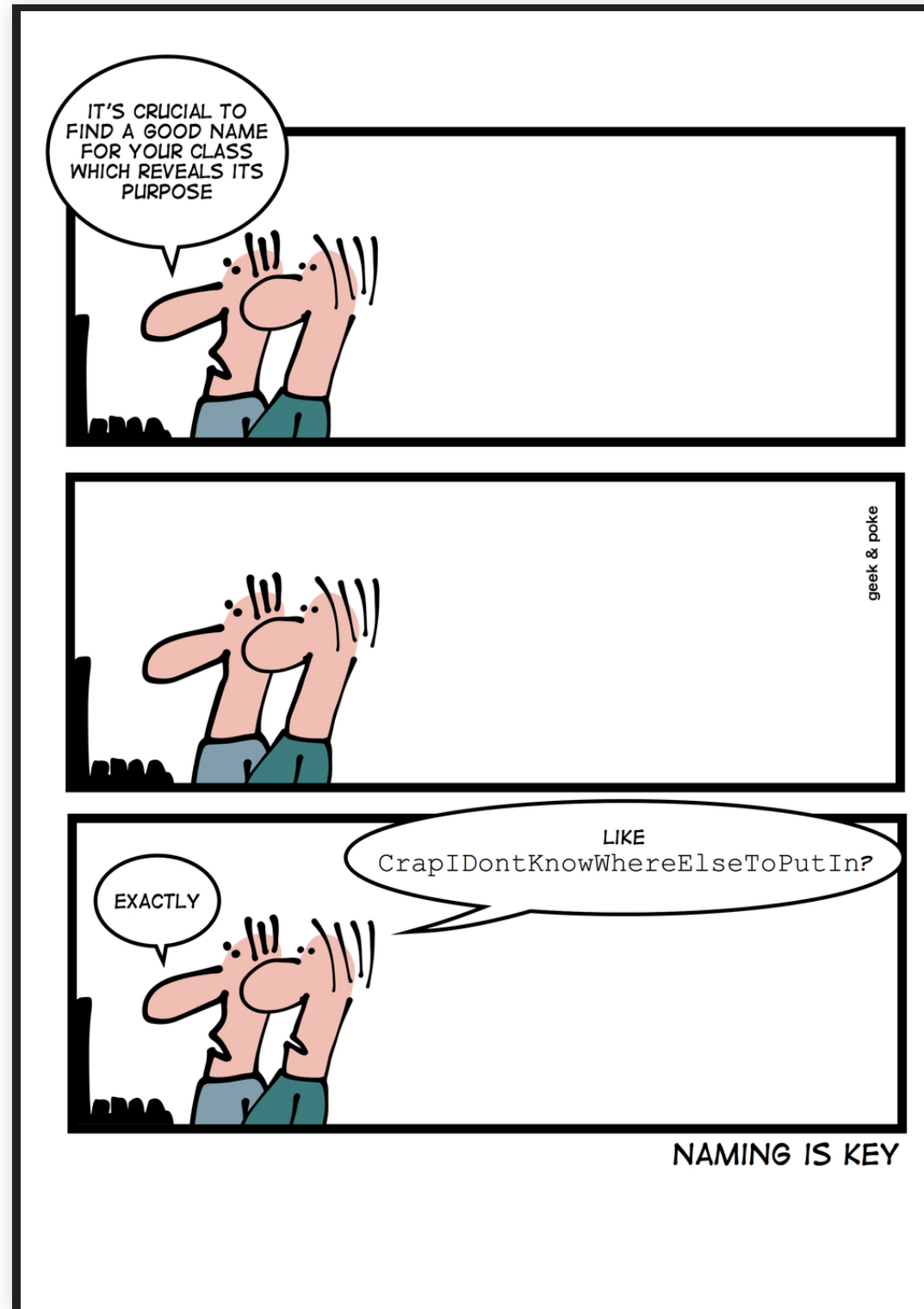
*Broken Window Principle*

— Robert “Uncle Bob” Martin

*Leave the campground cleaner than  
you found it*

— Robert “Uncle Bob” Martin







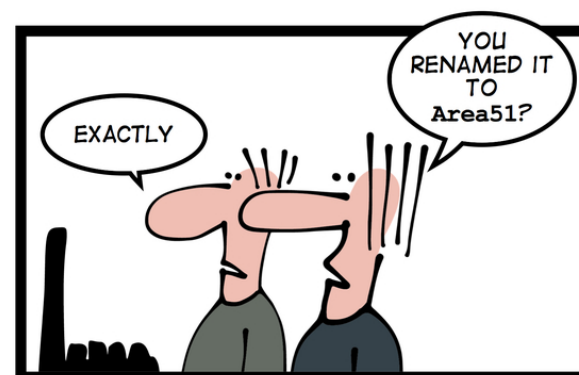
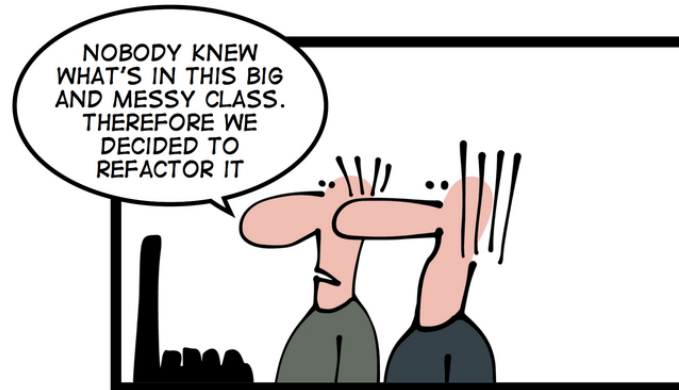
# DOKUMENTATION

- Nur Konzepte & Ideen
  - diese möglichst nahe am Code (== gut erreichbar für den Entwickler)
- Nicht WAS, sondern WARUM
- Je expressiver der Code, desto weniger Dokumentation ist nötig

# REFACTORING

Struktur verändern  
*ohne*  
Verhalten zu ändern

## REFACTORING IS KEY



# CONTINUOUS INTEGRATION

- Compilieren
- Paketieren
- (automatisiert) Testen
- Validieren
- Verifizieren

# NACHVOLLZIEHBARKEIT

Analyse → Anforderung → Aufgabe → Code → Test →  
Auslieferung

# METRIKEN

- Messen des **Status-Quo** der Software-Komplexität
  - Wird sie schlechter oder besser?
- Identifikation der größten Risiken
  - Wichtig für Priorisierung bei limitiertem Budget/Zeit

*The only valid code metric is WTFs per minute.*

— Robert “Uncle Bob” Martin

# SELBSTKRITIK

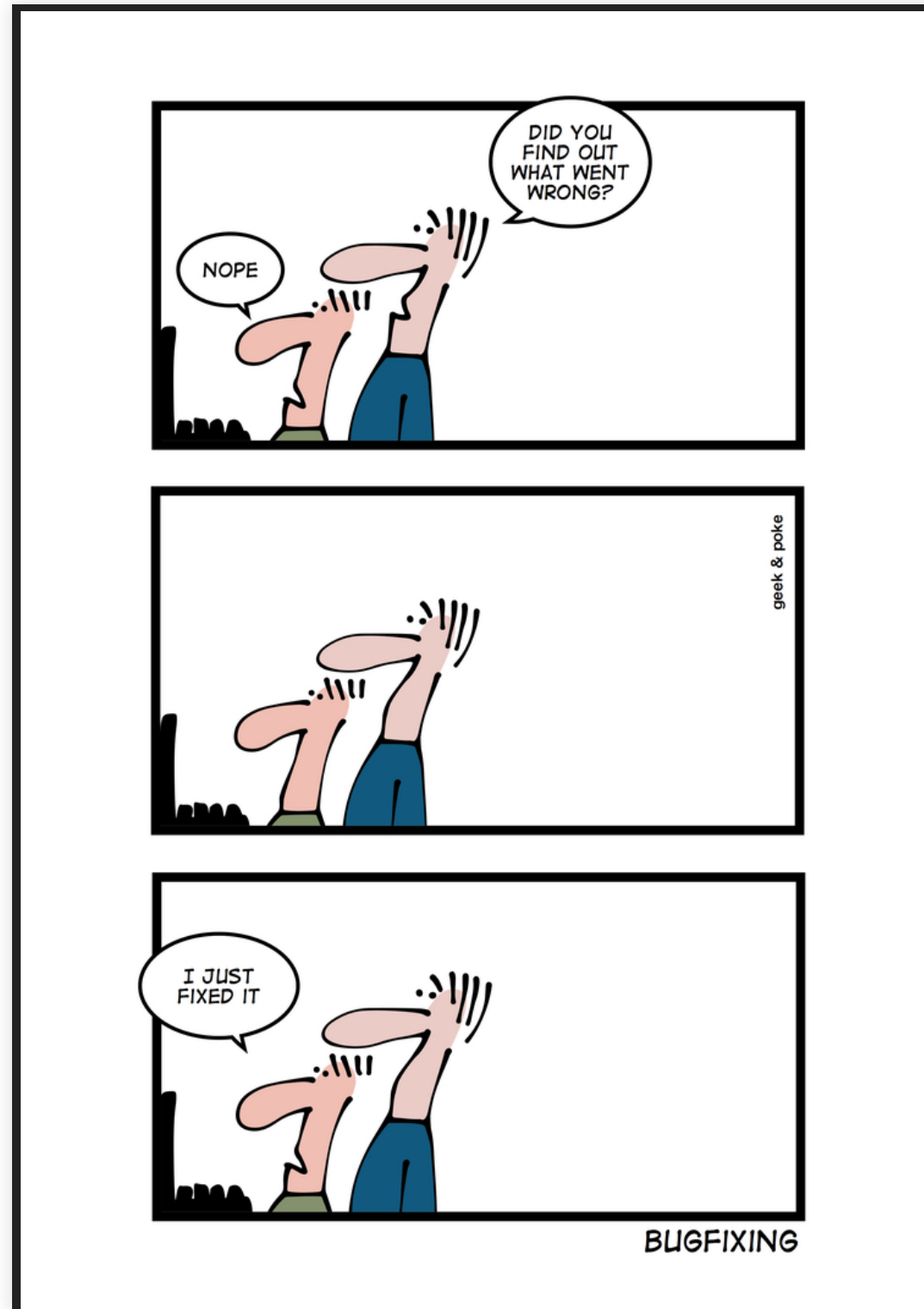
- Regelmäßig den eigenen Code überprüfen

*The worst programmer in the world is  
you – one year ago.*

— Andy Hunt







# ANALYTISCHES TALENT

- Analyse von Kundenwünschen
  - Was will der Kunde und WARUM?
- Analyse von Fehlern/Bugs
- Debugging
- Aufstellen und Verifizieren von Hypothesen
- Bewertung von Hypothesen
  - == Erfahrung
- Schritt-für-Schritt Vorgehen
- Zyklisch verfeinern

# FAZIT

Kontinuierliches Anwenden und Verbessern dieser  
Maßnahmen ist unsere beste Waffe gegen die  
zunehmende **Komplexität**

# QUELLEN

# BILDER

- Legacy Wall

<https://raw.githubusercontent.com/bettercodehub/pitch/master/assets/legacy-code.png>

- Legacy Survey <http://www.karoliki.com/2015/10/your-definition-of-legacy-impacts-how.html>

- Team [http://scrumbook.org.datasenter.no/images/SmallTeam\\_Pre.jpg](http://scrumbook.org.datasenter.no/images/SmallTeam_Pre.jpg)

- Programmers hardest task

<https://www.itworld.com/article/2833265/cloud-computing/don-t-go-into-programming-if-you-don-t-have-a-good-thesaurus.html>

- Naming is Key <http://geek-and-poke.com/geekandpoke/2013/8/20/naming-is-key>

- Refactoring is Key <http://geek-and-poke.com/geekandpoke/2013/8/26/refactoring-is-key>

# BUCHTIPPS

*The Pragmatic Programmer, From  
Journeyman To Master*

— Andy Hunt

*Clean Coder / Clean Coder*

— Robert “Uncle Bob” Martin