Programmierkonzepte in der Physikalischen Chemie

10. Sauberer Code

UNI FREIBURG

Albert-Ludwigs-Universität Freiburg

Dr. Till Biskup

Institut für Physikalische Chemie
Albert-Ludwigs-Universität Freiburg
Wintersemester 2018/19

Zentrale Aspekte

UNI FREIBURG

- Ziel der Programmierung:
  intellektuelle Beherrschung komplexer Zusammenhänge
- Programmierer kommunizieren durch ihren Code.
  Man kann nicht nicht, nur unzureichend kommunizieren.
- Die Qualität von Code nimmt über die Dauer der Entwicklung eines Projekts meist ab (Software-Entropie).
- Die Lösung ist nicht der große Neuentwurf,
  sondern die kontinuierliche Verbesserung im Kleinen.
- Code-Qualität ist eine Frage der persönlichen Einstellung und Wertschätzung des Lesers/Nutzers.

Standortbestimmung

UNI FREIBURG

Grobgliederung der Vorlesung „Programmierkonzepte“

1. Motivation
2. Infrastruktur
3. **Sauberer Code**
4. Architektur
5. Datenauswertung in der PC

- Code steht im Zentrum der Vorlesung
  - größter Teil der Vorlesung
  - (hoffentlich) etwas weniger abstrakt als bisher

Übersicht

UNI FREIBURG

Ziel der Programmierung:
intellektuelle Beherrschung komplexer Zusammenhänge

Die Aufgabe von Code: Kommunikation

Software-Entropie

Intellektuelle Beherrschbarkeit

Die Komplexität von Programmen überfordert den Menschen.

UNI FREIBURG

> *Nobody is really smart enough to program computers. Fully understanding an average program requires an almost limitless capacity to absorb details and an equal capacity to comprehend them all at the same time. The way you focus your intelligence is more important than how much intelligence you have.*
>
> – Steve McConnell

- Direkter Bezug auf E. Dijkstra („The Humble Programmer“)
- Ziel vieler Programmierkonzepte: komplexe Zusammenhänge intellektuell beherrschbar machen

Steve McConnell: Code Complete. Microsoft Press, Redmond 2004

Edsger Dijkstra, *Commun. ACM* **15**:859, 1972

Intellektuelle Beherrschbarkeit

Es gibt keine Patentlösung – nur harte Arbeit.

UNI FREIBURG

„No Silver Bullet“

- Programmierung ist inhärent komplex.
  - Kein Programm und keine Technik kann das ändern.
  - Keine Technik leistet eine Verzehnfachung der Effizienz.
- Es gibt Unterschiede in der Effizienz von Programmierern.
  - Erfahrung und Beherrschung der Werkzeuge
  - Unterschiede im Bereich einer Größenordnung
- Parallele zum Handwerk
  - Herangehensweise und Werkzeuge lassen sich erlernen.
  - Qualität ist eine Frage der Übung und Einstellung.

☛ Die Vorlesung kann nur Hinweise geben.
Die praktische Umsetzung ist Sache des Einzelnen.

Frederick P. Brooks: „No Silver Bullet“, in: ders., The Mythical Man Month. Addison Wesley, Boston 1995

Programmierung ist Kommunikation

Programmierer sind Schriftsteller – die meisten unbewusst.

UNI FREIBURG

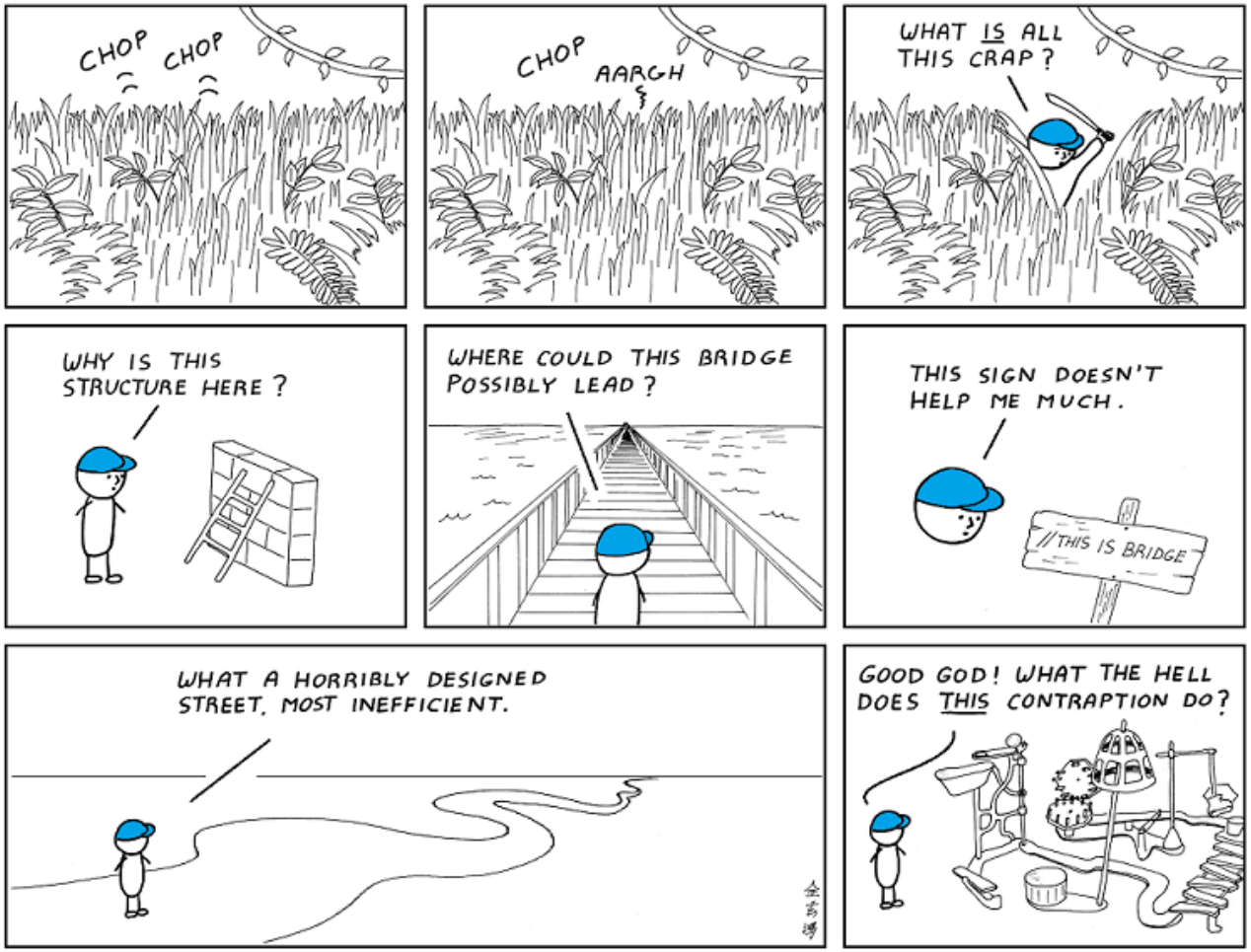
> *The computer programs that are truely beautiful, useful, and profitable must be readable by people.*
> *So we ought to address them to people, not to machines.*
> *All of the major problems associated with computer programming—issues of reliability, portability, learnability, maintainability, and efficiency—are ameliorated when programs and their dialogs with users become more literate.*
>
> – Donald E. Knuth

- Entscheidend ist die Perspektive: Programmierer kommunizieren durch ihren Code mit (anderen) Menschen.

Donald E. Knuth: Literate Programming, CSLI, Stanford 1992, S. ix

Programmierung ist Kommunikation

Warum es so schwer ist, fremden Code zu lesen...



I hate reading other people's code.

„Sauberer Code“ hat viele Aspekte

- in der Vorlesung behandelte Aspekte
  - Namen
  - Funktionen
  - Dokumentation im Code
  - Formatierung
  - Muster
- Das Versprechen von „Sauberem Code“
  - wiederverwendbar, nachvollziehbar, überprüfbar
  - dauerhaft gute, konstante Code-Qualität

☛ „Sauberer Code“ ist eine Frage der Einstellung, nicht der Abarbeitung von Listen.

Strategien und Werkzeuge

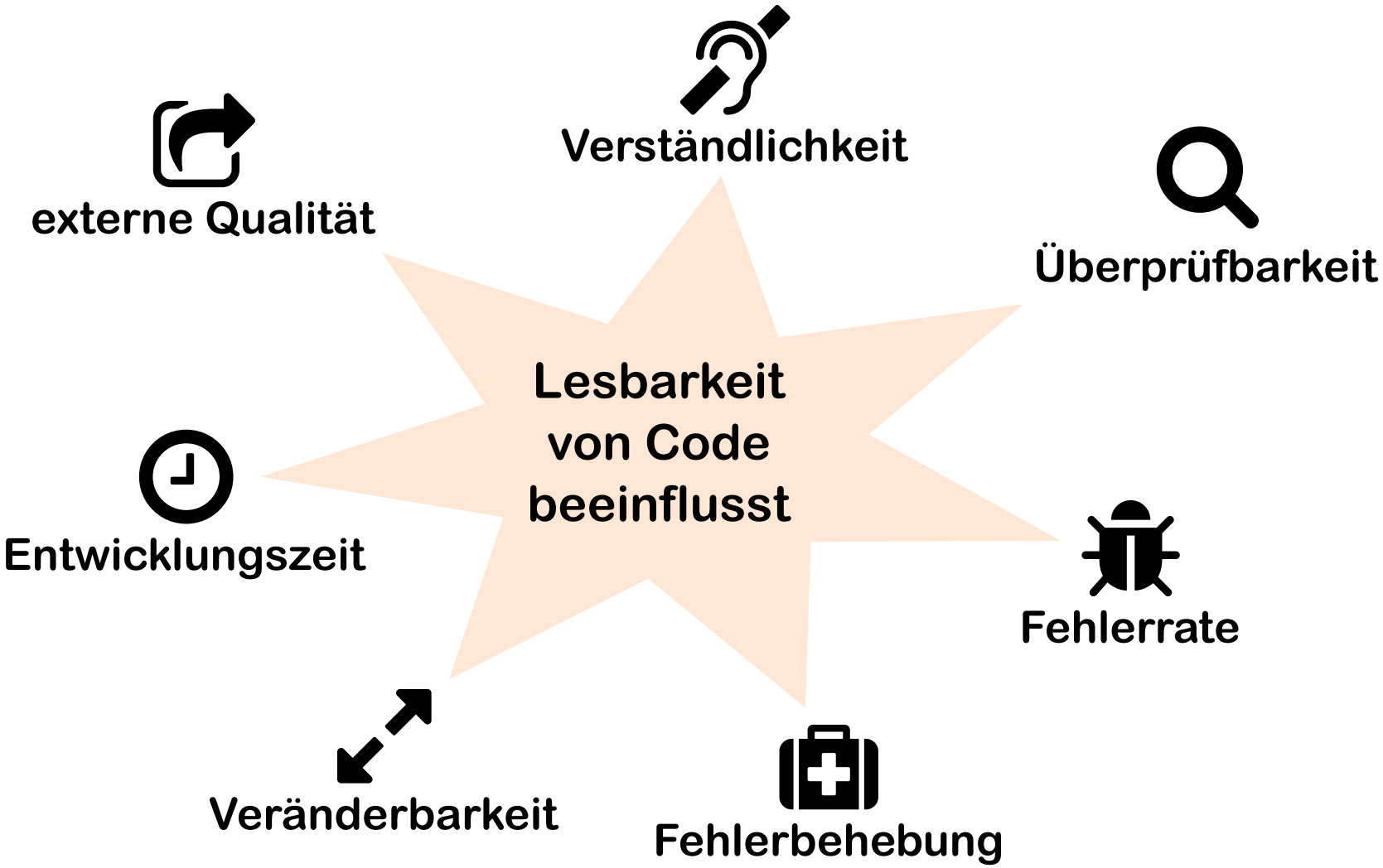
- Refactoring
  - Verbesserung existierenden Codes,
    ohne seine Funktionalität zu verändern
- automatisierte Tests
  - zwingende Voraussetzung für Refactoring
  - Versicherung gegen ungewollte Nebenwirkungen
  - autoritative Spezifikation des Verhaltens von Funktionen

☛ Code ist nicht beim ersten Mal lesbar und perfekt.
Lesbarer Code ist das Ergebnis harter Arbeit.

☛ systematisches und schrittweises Vorgehen

Programmierung ist Kommunikation

Effekte der Lesbarkeit von Quellcode



Software-Entropie

Die Qualität von Code nimmt über seine Lebenszeit hinweg ab. Grund sind unvermeidliche Anpassungen am Code und damit verbunden die Zunahme von Komplexität.

- kann dramatische Folgen haben
  - Ende des Projekts/Produkts oder gar von Unternehmen
  - akademischer Kontext: Verlust der Nachvollziehbarkeit
- kein Naturgesetz
  - Es gibt wirksame Strategien, um gegenzuarbeiten.
  - setzt permanentes aktives Gegensteuern voraus

Software-Entropie

Die Programmierer selbst sind die Verursacher

UNI FREIBURG

Broken-Window-Theorie

beschreibt, wie ein vergleichsweise harmloses Phänomen, beispielsweise ein zerbrochenes Fenster in einem leer stehenden Haus, später zu völliger Verwahrlosung führen kann

- Wie verschlechtert sich die Qualität des Codes?
  - Antwort: Schritt für Schritt für Schritt...
- Wehret den Anfängen!
  - Harmlose Kleinigkeiten haben oft schwerwiegende Folgen.
- Der Programmierer ist verantwortlich für den Code.
  - Es gibt keine Ausreden... schon gar nicht Zeitmangel.

James Q. Wilson & George L. Kelling, *The Atlantic Monthly* **249**:29, 1982

Software-Entropie

Gegenmaßnahmen: die Pfadfinderregel

UNI FREIBURG

> *Hinterlasse einen Ort immer in einem besseren Zustand als du ihn vorgefunden hast.*
>
> – Pfadfinderregel

- Aufräumen ist eine Dauerangelegenheit.
  - Für große Aufräumaktionen fehlt oft die Zeit.
  - Aufräumen muss tägliche Praxis sein.
  - Fokus auf dem gerade bearbeiteten Codeblock
- Wer verantwortlich war, spielt keine Rolle.
  - Die Gründe sind genauso unerheblich.
  - oft Unwissen oder mangelnde Fähigkeit
- Verbessern – nicht perfekt machen (wollen)
  - Für Perfektion fehlt den meisten von uns die Fähigkeit.

Beispiele für kleine Verbesserungen

- Ändern eines (Variablen-)Namens
  - besser verständlich
  - lesbarerer Code
- Aufteilen einer (zu langen) Funktion
  - verbesserte Übersicht
  - verringerte Fehleranfälligkeit
- Entfernen einer Doppelung im Code
  - im jeweiligen direkten Kontext

☞ Voraussetzung: ausreichende Testabdeckung
  - Sicherheitsnetz, um korrekte Funktionalität zu erhalten

Vorteile kleiner Änderungen gegenüber dem Neuentwurf

- intellektuell beherrschbar
  - Programme sind in der Regel zu komplex.
  - Ziel der Programmierung: Zerlegung in beherrschbare Teile
- in die Programmierroutine integrierbar
  - erfordert Kenntnis, Bewusstsein und Disziplin
  - Zeit ist kein Argument: Später kostet es viel mehr Zeit.
- wenig zeitaufwendig
  - Aufwand skaliert mit der Erfahrung.
- umsetzbar und dauerhaft wirksam
  - Neuentwurf mag verlockend sein – meist fehlt die Zeit.
  - Neuentwurf behebt nicht die Ursache schlechten Codes.

Zentrale Aspekte

UNI FREIBURG

- Ziel der Programmierung:
  intellektuelle Beherrschung komplexer Zusammenhänge
- Programmierer kommunizieren durch ihren Code.
  Man kann nicht nicht, nur unzureichend kommunizieren.
- Die Qualität von Code nimmt über die Dauer der Entwicklung eines Projekts meist ab (Software-Entropie).
- Die Lösung ist nicht der große Neuentwurf,
  sondern die kontinuierliche Verbesserung im Kleinen.
- Code-Qualität ist eine Frage der persönlichen Einstellung und Wertschätzung des Lesers/Nutzers.