Programmierkonzepte in der Physikalischen Chemie

2. Motivation (Programmierung)

UNI FREIBURG

Albert-Ludwigs-Universität Freiburg

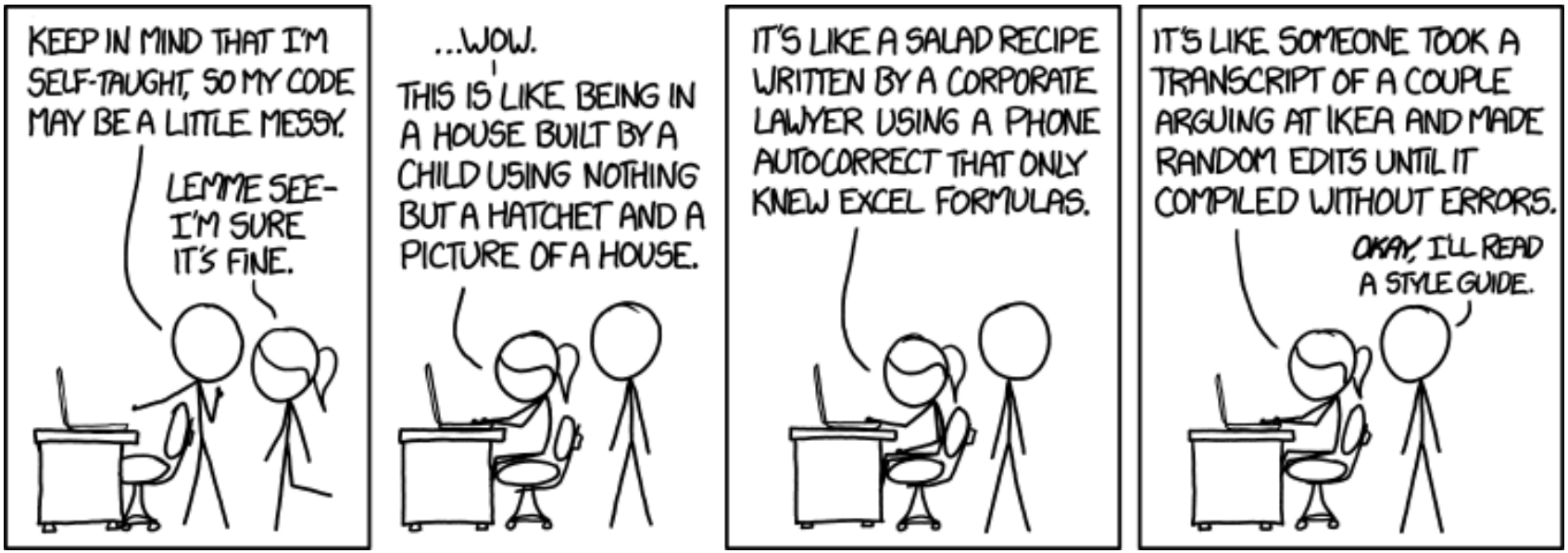
Dr. Till Biskup

Institut für Physikalische Chemie
Albert-Ludwigs-Universität Freiburg
Wintersemester 2016/17

Zentrale Aspekte

- Abstraktion ist der Schlüssel zur Beherrschung komplexer Fragestellungen.
- Intellektuelle Beherrschbarkeit erfordert Disziplin bei der Programmierung.
- Wiederverwertbarkeit, Zuverlässigkeit und Überprüfbarkeit erfordern modularen, getesteten und *lesbaren* Quellcode.
- Codequalität ist nicht optional für wissenschaftliche Auswertungssoftware.
- Programmierung hat in den Wissenschaften einen zu geringen Stellenwert.

Code Quality



> *Any fool can write code that a computer can understand. Good programmers write code that humans can understand.*
>
> – Martin Fowler

Codequalität ist direkt mit drei Aspekten verknüpft:

- Wiederverwendbarkeit
- Zuverlässigkeit
- Überprüfbarkeit

☛ Mangelnde Codequalität ist ein lange bekanntes Problem...

Fowler: *Refactoring*, Addison-Wesley Longman, Boston 1999

Übersicht

UNI FREIBURG

Ausgangspunkt: Die Softwarekrise

Lösungsansatz: Softwaretechnik (Software Engineering)

Problem der Wissenschaft: Programmierung nur Randthema

Schlüssel zum Erfolg: Konzepte und persönliche Disziplin

Ausgangspunkt: Die Softwarekrise

Der Bericht eines Augenzeugen

UNI FREIBURG

> *...so 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. [...]*
>
> *The increased power of the hardware, together with the perhaps even more dramatic increase in its reliability, made solutions feasible that the programmer had not dared to dream about a few years before. And now, a few years later, he had to dream about them and, even worse, he had to transform such dreams into reality!*
>
> – Edsger Dijkstra

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

Ausgangspunkt: Die Softwarekrise

Parallele zur Situation in den Naturwissenschaften

UNI FREIBURG

Ansprüche an Software

- viele Auswertungen software-basiert
- Komplexität mitunter sehr erheblich

Programmierkenntnisse und Fähigkeiten

- meist nur Grundkenntnisse in einer/wenigen Sprachen
- fehlende Kenntnisse über grundlegende Konzepte

☛ Qualität der Software häufig ungenügend:
nicht wiederverwendbar, zuverlässig, überprüfbar

Ausgangspunkt: Die Softwarekrise

Parallele zur Situation in den Naturwissenschaften

UNI FREIBURG

These

Die meisten Wissenschaftler haben nicht das Wissen
und die (mentalen) Werkzeuge, um die Möglichkeiten,
die ihnen heutige Computer und Programmiersprachen bieten,
für ihre Zwecke sinnvoll und gewinnbringend einzusetzen.

- Es liegt nicht an den grundlegenden Fähigkeiten, sondern am fehlenden Wissen.
- rührt an die Grundlagen wissenschaftlichen Arbeitens

Lösungsansatz: Software Engineering

Zum Begriff

UNI FREIBURG

Definition (Software engineering)

1. the systematic application of scientific and technological knowledge, methods, and experience to the design, implementation, testing, and documentation of software [...]

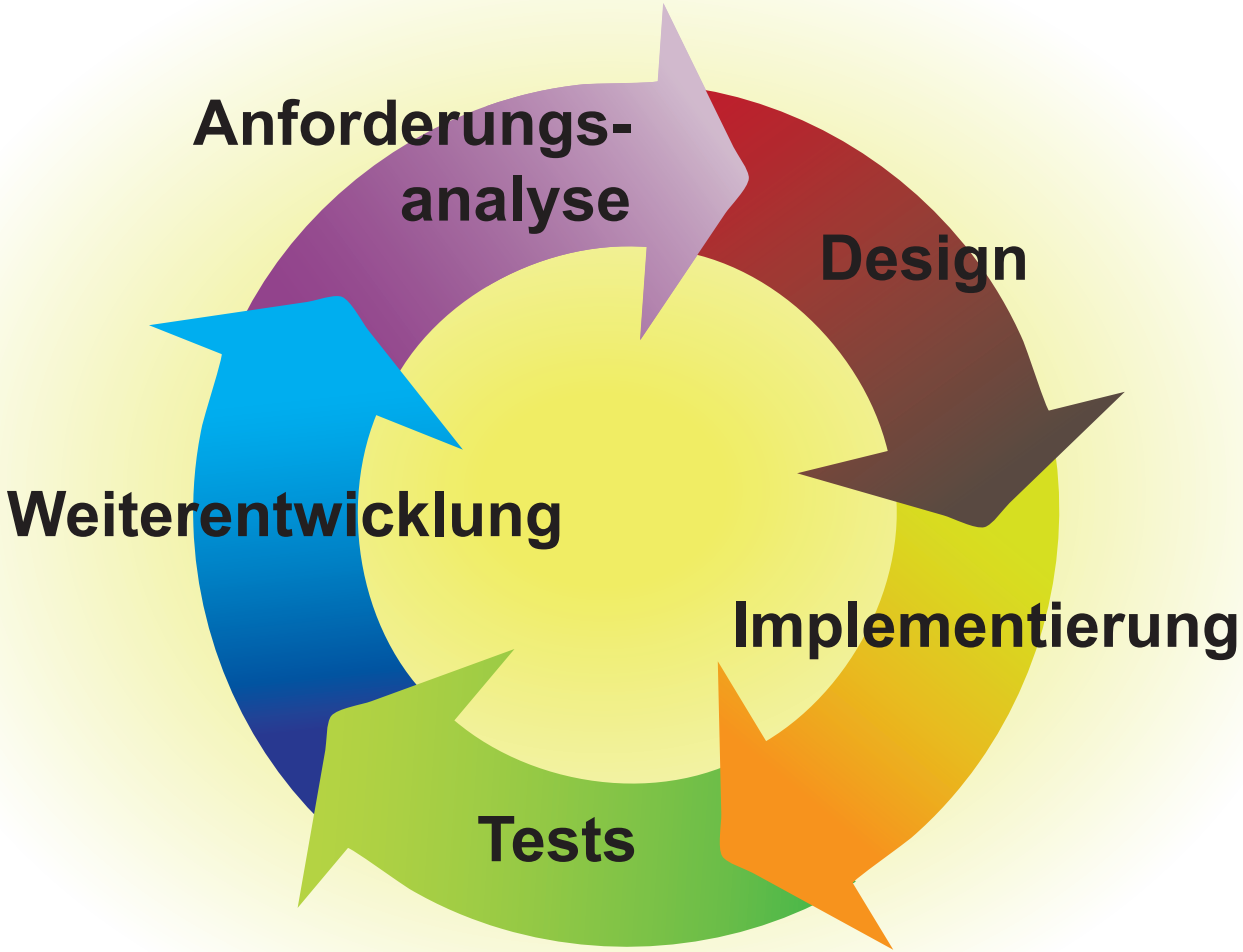
2. the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software

- Softwareentwicklung ist eher eine Ingenieursdisziplin.
- Es gibt etablierte Strategien für größere Projekte.

ISO/IEC/IEEE 24765:2010

Grundlegende Aspekte der Softwareentwicklung

- Projektmanagement
  - Parallele aus dem Ingenieurswesen
  - Formalisierung einiger Prozessabläufe
- Intellektuelle Beherrschbarkeit
  - Aufteilung in Unterprojekte
- Abstraktion
  - Abbildung der Problemstellung in Software
  - Entwicklung entsprechender Sprachkonzepte
- Muster
  - Wiederverwendung etablierter Lösungen
  - Kenntnis der Muster, Erkenntnis der Einsatzmöglichkeiten


Anforderungs-
analyse
Design
Implementierung
Tests
Weiterentwicklung

Softwarentwicklung braucht Planung und Formalisierung

- Unterteilung in kleinere Abschnitte
  - Projekte werden schnell komplex.
  - Voraussetzung für parallele Bearbeitung
- „Schnittstellen“ zwischen einzelnen Abschnitten
  - Einzelne Abschnitte werden parallel bearbeitet.
  - erhöht potentielle Wiederverwertbarkeit einzelner Bereiche
- Erfolgskriterien
  - Erreichen einzelner Ziele
  - Spezifikationen von Teilprojekten/Abschnitten

☛ Projektmanagement ist ein eigener Themenkomplex...

Projektmanagement

Umgang mit sich ändernden Anforderungen

UNI FREIBURG

> *We live in a world of changing requirements, and our job is to make sure that our software can survive those changes.*
>
> – Robert Martin

- Problem
  - Die Anforderungen an Software ändern sich häufig.
  - Starres Projektmanagement kann problematisch werden.
- (mögliche) Lösung
  - flexible Entwicklungskonzepte („Agile“)
  - Code von Anfang an auf Veränderungen auslegen.

Martin: *Agile Software Development*, Prentice Hall, Upper Saddle River 2003

Software wird schnell sehr komplex

- Problem komplexer Zusammenhänge
  - menschliche Fähigkeit, mehrere Aspekte gleichzeitig zu erfassen, ist sehr eingeschränkt
- Ziel der Softwareentwicklung
  - intellektuelle Beherrschbarkeit (*intellectual manageability*)
- Aspekte intellektueller Beherrschbarkeit
  - Anzahl der Aufgaben eines Programmabschnittes
  - Länge eines zusammenhängenden Programmabschnittes
  - Klarheit der Formulierungen

☛ Konsequenzen für die Programmierung

Bausteine intellektueller Beherrschbarkeit

- KISS
  - *Keep it simple, stupid*
  - Parallele: Sparsamkeitsprinzip in den Wissenschaften
- Modularität
  - Aufteilung der Gesamtaufgabe in kleine Abschnitte.
  - iterativer Prozess
- Unix-Philosophie
  - *Do one thing, and do it well.*
  - Jeder Programmabschnitt erfüllt genau eine Aufgabe.

Schlüssel zur intellektuellen Beherrschbarkeit: Abstraktion

> *the only mental tool by means of which a very finite piece of reasoning can cover a myriad of cases is called "abstraction" ...*
>
> *...the purpose of abstracting is not to be vague, but to create a new semantic level in which one can be absolutely precise.*
>
> – Edsger Dijkstra

- ☛ entscheidend für die Lösung eines realen Problems mit Hilfe von Software

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

Konsequenzen fehlender Abstraktion:

> *Cut and paste may be useful text-editing operations, but they can be disastrous code-editing operations. [...]*
>
> *When the same code appears over and over again, in slightly different forms, the developers are missing an abstraction. Finding all the repetition and eliminating it with an appropriate abstraction may not be high on their priority list, but it would go a long way toward making the system easier to understand and maintain.*
>
> – Robert Martin

- DRY: *Don't repeat yourself*

Martin: *Agile Software Development*, Prentice Hall, Upper Saddle River 2003

Das Rad wurde bereits erfunden...

- Viele Probleme sind nicht grundlegend neu:
  Es gibt allgemeine und erprobte Lösungen.
  - Es gibt ganze Kataloge mit Mustern.
- Parallele aus der Architektur
  - Christopher Alexander (1977):
    A Pattern Language. Towns, Buildings, Construction
- Muster erleichtern die Kommunikation.
  - Namenskonventionen für viele Muster

☛ Kenntnis der entsprechenden Entwurfsmuster
und korrekte Benennung

Ansprüche

Wiederverwendbarkeit, Zuverlässigkeit,
Überprüfbarkeit

Werkzeuge

Projektmanagement, intellektuelle Beherrschbarkeit,
Abstraktion, Muster

Ergebnis

modularer, lesbarer, getesteter Code

Zusammenhang mit den Themen der Vorlesung

- Infrastruktur
  - formalisiert Abläufe, sorgt für Struktur und Überprüfbarkeit
  - erleichtert die Arbeit des Programmierers
  - Voraussetzung: einfach bedienbar, schnell
- Code
  - intellektuelle Beherrschbarkeit, Abstraktion
  - „Clean Code“:
    KISS; DRY; Unix-Philosophie, Modularisierung...
- Architektur
  - Aufteilung in Unterprojekte
  - Definition von Schnittstellen/Anforderungen

Geschichte wiederholt sich...

> *Many scientists and engineers spend much of their lives writing, debugging, and maintaining software, but only a handful have ever been taught how to do this effectively: after a couple of introductory courses, they are left to rediscover (or reinvent) the rest of programming on their own.*
> *The result? Most spend far too much time wrestling with software, instead of doing research, but have no idea how reliable or efficient their programs are.*
>
> – Greg Wilson

in: Baxter *et al.*, *PLoS Comput. Biol.* **2**:e87, 2006

Problem der Wissenschaft

Mangelndes Wissen über Programmierkonzepte

UNI FREIBURG

Problem: mangelndes Wissen

- Programmierung wird selten gelehrt –
  (nicht nur) in der Physikalischen Chemie.
- Programmierung ist aber essentiell
  für viele Aspekte der Physikalischen Chemie.

Motivation für die Vorlesung

- Programmierung ist gar nicht so schwer zu lernen.
- Es gibt von Profis viele einfache und gute Hinweise.
- Hinweise und Tipps für guten Code

Problem: mangelnder Stellenwert

- Programmierung ist (fast) immer Mittel zum Zweck
  - kein hoher Stellenwert
  - hilft nicht beim Weiterkommen bzw. Einwerben von Geld
- Oftmals nur ein Hauptentwickler
  - Vorteile des Teams fehlen
  - keine Motivation, für andere nutzbaren Code zu schreiben
- Problembewusstsein fehlt
  - Mangelnde Softwarequalität hat direkten Einfluss auf die Qualität der Wissenschaft.
  - Reproduzierbarkeit rechnergestützter Datenauswertung im Regelfall nicht gegeben

Disziplin ist beim Programmieren essentiell

- Programmieren ist eine systematische Tätigkeit
  - Zusammenhacken von Quellcode und $\pm$ erratische Fehlerbehebung geben keinerlei Einblick und sind langfristig gefährlich.
- Erlernen und konsequentes Verwenden etablierter Strategien und Muster
  - nicht das Rad neu erfinden...
  - verwendete Muster und Strategien beim Namen nennen
- Befolgen von Konventionen
  - Der konkrete Inhalt ist weniger wichtig.
  - Wichtig ist die konsequente Umsetzung.

Programmieren ist eine Art der Kommunikation

- „Code for people, not computers“
  - Kommunikation mit denjenigen, die das Programm lesen (und verstehen wollen) – oft genug der Entwickler selbst
- *Lesbarkeit* von Code ist zentral.
  - Viele Ansätze seit den 1960er Jahre haben ein Ziel: Lesbarkeit/Kommunikationsfähigkeit von Quellcode
- Große Freiheiten erfordern entsprechende Disziplin.
  - (fast) alles ist möglich – aber nicht immer sinnvoll
  - Verantwortung des Programmierers

Schlüssel zum Erfolg

Persönliche Einstellung

UNI FREIBURG

These

Code-Qualität ist eine Frage der persönlichen Einstellung.

- Wer durch Code kommunizieren will...
  - ...wird sich mühen,
    sich das notwendige Wissen anzueignen.
  - ...wird immer wieder die notwendige Disziplin aufbringen,
    das Wissen anzuwenden.

- Ein Wort der Warnung:
  - Quellcode offenbart wie wenig sonst
    Herangehensweise und Charakter einer Person.

Zentrale Aspekte

UNI FREIBURG

- Abstraktion ist der Schlüssel zur Beherrschung komplexer Fragestellungen.
- Intellektuelle Beherrschbarkeit erfordert Disziplin bei der Programmierung.
- Wiederverwertbarkeit, Zuverlässigkeit und Überprüfbarkeit erfordern modularen, getesteten und *lesbaren* Quellcode.
- Codequalität ist nicht optional für wissenschaftliche Auswertungssoftware.
- Programmierung hat in den Wissenschaften einen zu geringen Stellenwert.