

Programmierkonzepte in der Physikalischen Chemie

3. Infrastruktur

Albert-Ludwigs-Universität Freiburg



**UNI
FREIBURG**

Dr. Till Biskup

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



Zentrale Aspekte

- ❖ Eine minimale Infrastruktur ist zwingende Voraussetzung zur Erfüllung der Kriterien für Auswertungssoftware.
- ❖ Infrastruktur kann die Softwareentwicklung vereinfachen, disziplinieren und strukturieren.
- ❖ Klare Abläufe sorgen für Konsistenz und Routine und erleichtern die Nutzung.
- ❖ Nur eine funktionierende und möglichst einfach bedienbare Infrastruktur wird auch regelmäßig genutzt werden.
- ❖ Entscheidend ist nicht der Einsatz konkreter Produkte, sondern der Werkzeuge als solcher.

Warum ist Infrastruktur wichtig?

Kosten-Nutzen-Abwägung

Übersicht über die nachfolgend behandelte Infrastruktur

Weitere Arten von Infrastruktur

Ausgangspunkt der Vorlesung

- ▶ Programmierung größerer Projekte
 - Beispiel: Datenauswertung
- ▶ Klar definierte Ansprüche an die entwickelte Software
 - Wiederverwendbarkeit, Zuverlässigkeit, Überprüfbarkeit
- ▶ Software für die wissenschaftliche Datenanalyse
 - Ansprüche wurden entsprechend definiert
 - führt (fast) zwangsläufig zu „größerem Projekt“
- ▶ Softwareentwicklung ist Mittel zum Zweck
 - Physikalische Chemie:
Verständnis von Zusammenhängen, Auswertung von Daten
 - Softwareentwicklung so einfach und effizient wie möglich

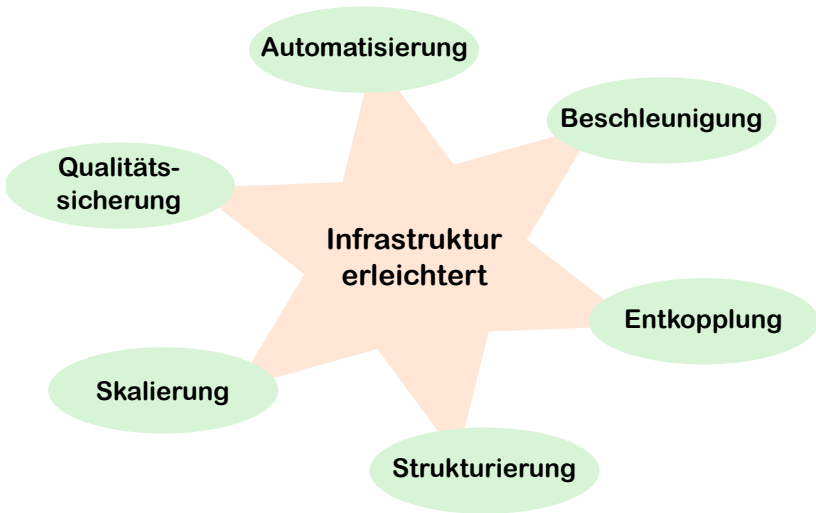
Größeres Projekt

Alles, was mehr als zwei Wochen Arbeit kostet und deutlich mehr als zweihundert Zeilen (reinen) Quellcode bzw. mehr als eine Handvoll Unterfunktionen umfasst.

- ▶ Genaue Zahlen sind immer problematisch.
- ▶ Wichtig ist der Fokus.
- ☞ Qualitativ hochwertige Software zur wissenschaftlichen Datenauswertung fällt in diese Kategorie.

Warum ist Infrastruktur wichtig?

Gründe für die Nutzung



Gründe für die Nutzung von Infrastruktur (I)

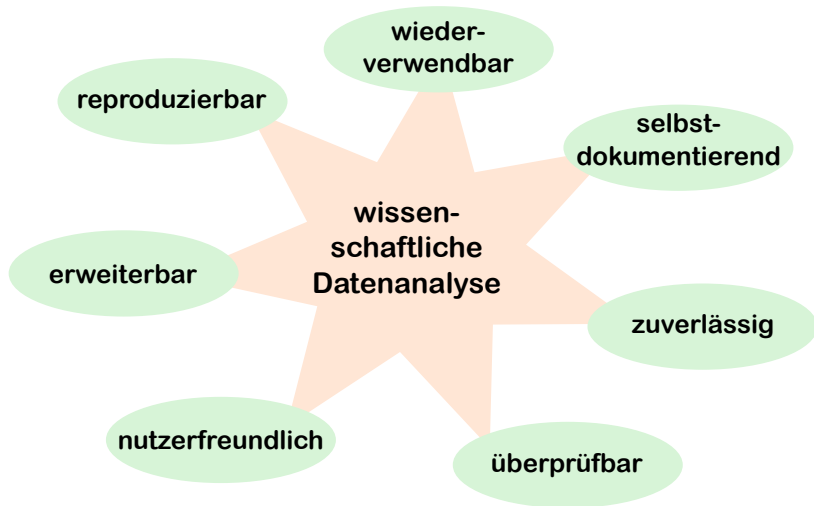
- ▶ Automatisierung
 - nimmt dem Programmierer/Entwickler Arbeit ab
 - Automatisierung von Routineaufgaben
- ▶ Beschleunigung
 - ermöglicht ein zügigeres/einfacheres Arbeiten
 - Routineaufgaben automatisierbar und so beschleunigbar
 - Formalisierung sorgt für Konsistenz und Fokussierung.
- ▶ Entkopplung
 - entkoppelt den einzelnen Entwickler von der Software
 - erleichtert die Übernahme durch andere Entwickler
 - im akademischen Kontext stark unterschätzter Aspekt
 - einer der Hauptantriebe für diese Vorlesung

Gründe für die Nutzung von Infrastruktur (II)

- ▶ **Strukturierung**
 - diszipliniert und strukturiert die Softwareentwicklung
 - Projektmanagement als Antwort auf die Softwarekrise
 - für jedes größere Projekt unumgänglich
 - Nutzen für die Entwickler steht im Vordergrund
- ▶ **Skalierung**
 - essentiell für größere Projekte/bei mehreren Entwicklern
 - Entwickler müssen nicht notwendigerweise gleichzeitig am Projekt arbeiten.
- ▶ **Qualitätssicherung**
 - Voraussetzung für hochwertige Auswertungssoftware
 - Kriterien wurden eingangs aufgestellt.

Warum ist Infrastruktur wichtig?

Wiederholung: Kriterien für Auswertungssoftware



Warum ist Infrastruktur wichtig?

Eine minimale Infrastruktur für wissenschaftliche Datenauswertung

Minimale Infrastruktur für jede Auswertungsroutine

- ▶ Versionskontrolle
- ▶ eindeutige Versionsnummer
- ▶ Urheberrechtshinweis

Hilfreich und dringend angeraten

- ▶ Dokumentation
 - ▶ Lizenz
- ☞ Für größere Projekte werden alle nachfolgend genannten Aspekte von Infrastruktur benötigt.

Aspekte einer Kosten-Nutzen-Abwägung

- ▶ Umgang mit Infrastruktur muss erlernt werden.
 - Programmierung ist bereits Mittel zum Zweck.
 - Infrastruktur erschwert initial die Arbeit.
- ▶ Betrieb der Infrastruktur bindet Ressourcen.
 - Infrastruktur sollte langlebig sein (>10 Jahre).
 - Pflege muss entsprechend sichergestellt werden.
- ▶ Infrastruktur macht Manches erst möglich.
 - verteiltes Arbeiten an einem Projekt
 - Versionsverwaltung
 - ...
- ☞ Nutzung möglichst komfortabel gestalten (Akzeptanz!)

Umgang mit Infrastruktur muss erlernt werden

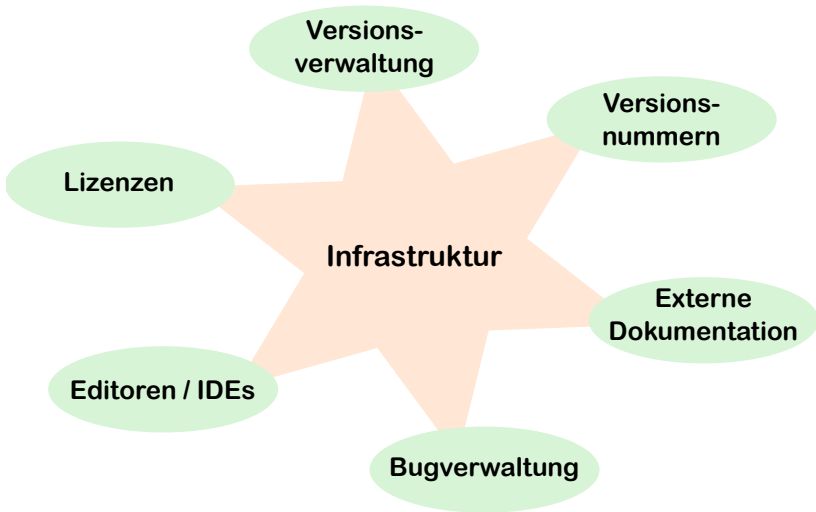
- ▶ Programmierung ist nur Mittel zum Zweck.
 - Jede weitere (Einstiegs-)Hürde will gut motiviert sein.
- ▶ Lösungsansatz: Reduktion auf das Notwendige
 - vorgestellte Aspekte praxisnah und auf den akademischen Kontext zugeschnitten
 - größere Projekte nicht anders sinnvoll beherrschbar
- ▶ Fokus der Vorlesung: Konzepte statt konkrete Programme
 - konkrete Lösungen oft von persönlichen Vorlieben und vom gegebenen Umfeld abhängig
- ☛ Kenntnis im Umgang mit der vorgestellten Infrastruktur ist eine gute Zusatzqualifikation.

Betrieb der Infrastruktur bindet Ressourcen

- ▶ Arten möglicher Kosten für Infrastruktur
 - Software
 - Hardware (Rechner, zentrale Server)
 - Wartung und Pflege (Personalkosten)
- ▶ viele Lösungen als freie Software verfügbar
 - alle hier vorgestellten Aspekte über freie Software lösbar
- ▶ integrierte Lösungen über kostenlose Online-Services
 - vorher ggf. rechtliche Aspekte klären
- ▶ Hardware
 - In kleinen Gruppen reicht ein Arbeitsplatzrechner.
 - Ein dedizierter Server pro Gruppe ist relativ günstig.

Übersicht über die Infrastruktur

Jeder Aspekt wird separat im Detail behandelt werden



Einzelne Aspekte einer Projekt-Infrastruktur (I)

- ▶ Versionsverwaltung (VCS)
 - erlaubt jederzeit den Rückgriff auf beliebige vorherige Versionen einer einzelnen Routine/des ganzen Projektes
 - befreiend für den Entwickler (Sicherheitsnetz!)
- ▶ Versionsnummern
 - eineindeutig für jede Version einer Routine
 - *per se* unabhängig von der Versionsverwaltung
- ▶ (externe) Dokumentation
 - Konzepte, Anforderungsanalysen, Coding Conventions etc.
 - Alles, was nicht sinnvoll direkt im Quellcode ablegbar ist

Einzelne Aspekte einer Projekt-Infrastruktur (II)

- ▶ Bugverwaltung
 - formalisiert Fehlerberichte an die Entwickler
 - eindeutige Bezeichner erleichtern Bezugnahme
 - räumliche und zeitliche Trennung von Entwicklern und Anwendern
- ▶ Editoren/IDEs
 - IDEs können die Produktivität immens steigern.
 - Einarbeitung oft zeitintensiv
- ▶ Lizenzen und Urheberrechte
 - Quellcode unterliegt *per se* dem Urheberrecht.
 - Wiederverwendbarkeit nur bei klarer Rechtslage/Lizenz
 - im akademischen Kontext oft vernachlässigt

		#			
wiederverwendbar	☒		☒		☒
selbstdokumentierend		☒			
zuverlässig				☒	
überprüfbar			☒	☒	
nutzerfreundlich			☒	☒	
erweiterbar	☒		☒		☒
reproduzierbar	☒	☒	☒		

 – VCS, **#** – Versionsnummern,  – Dokumentation,
 – Bugverwaltung,  – Lizenzen

- ▶ Automatisierte Tests
 - zentraler Aspekt für zuverlässigen Code
- ▶ Codeanalyse-Werkzeuge
 - statische und dynamische Codeanalyse
 - stark von der verwendeten Programmiersprache abhängig
- ▶ Generierung von Dokumentation
 - für Entwicklerdokumentation geeignet
 - setzt Disziplin bei der Codedokumentation voraus
- ▶ Build-Umgebungen
 - für komplexere Projekte (z.B. plattformübergreifend)
- ▶ Projektmanagement-Software
 - meist nur für große Projekte relevant

Aspekte einer „weichen“ Infrastruktur

- ▶ regelmäßige Projekttreffen
 - klein und groß, formal und informell
 - Code-Review, Strategiediskussionen, ...
- ▶ Kommunikation
 - auf allen Ebenen und Kanälen
 - Konventionen/Entscheidungen schriftlich fixieren
- ▶ Weiterbildungsangebote
 - ausgewählte Literatur
 - Kurse zu spezifischen Themen
- ▶ „Pair Programming“
 - Zweierteams gemeinsam an einem Rechner
 - kaum zu überschätzende Wirkung



Zentrale Aspekte

- ❏ Eine minimale Infrastruktur ist zwingende Voraussetzung zur Erfüllung der Kriterien für Auswertungssoftware.
- ❏ Infrastruktur kann die Softwareentwicklung vereinfachen, disziplinieren und strukturieren.
- ❏ Klare Abläufe sorgen für Konsistenz und Routine und erleichtern die Nutzung.
- ❏ Nur eine funktionierende und möglichst einfach bedienbare Infrastruktur wird auch regelmäßig genutzt werden.
- ❏ Entscheidend ist nicht der Einsatz konkreter Produkte, sondern der Werkzeuge als solcher.