Programmierkonzepte in den Naturwissenschaften

3. Infrastruktur

PD Dr. Till Biskup
Physikalische Chemie und Didaktik der Chemie
Universität des Saarlandes
Sommersemester 2020

Zentrale Aspekte

- Eine minimale Infrastruktur ist zwingende Voraussetzung zur Erfüllung der Kriterien für Auswertungssoftware.
- Richtig eingesetzt kann Infrastruktur 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.

Standortbestimmung

Grobgliederung der Vorlesung „Programmierkonzepte“

1. Motivation
2. **Infrastruktur**
3. Sauberer Code
4. Architektur
5. Datenauswertung in den Naturwissenschaften

☛ Infrastruktur ist eine notwendige Voraussetzung.
- Beschränkung auf wesentliche Aspekte
- überwiegend Konzepte statt konkrete Programme

Übersicht

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

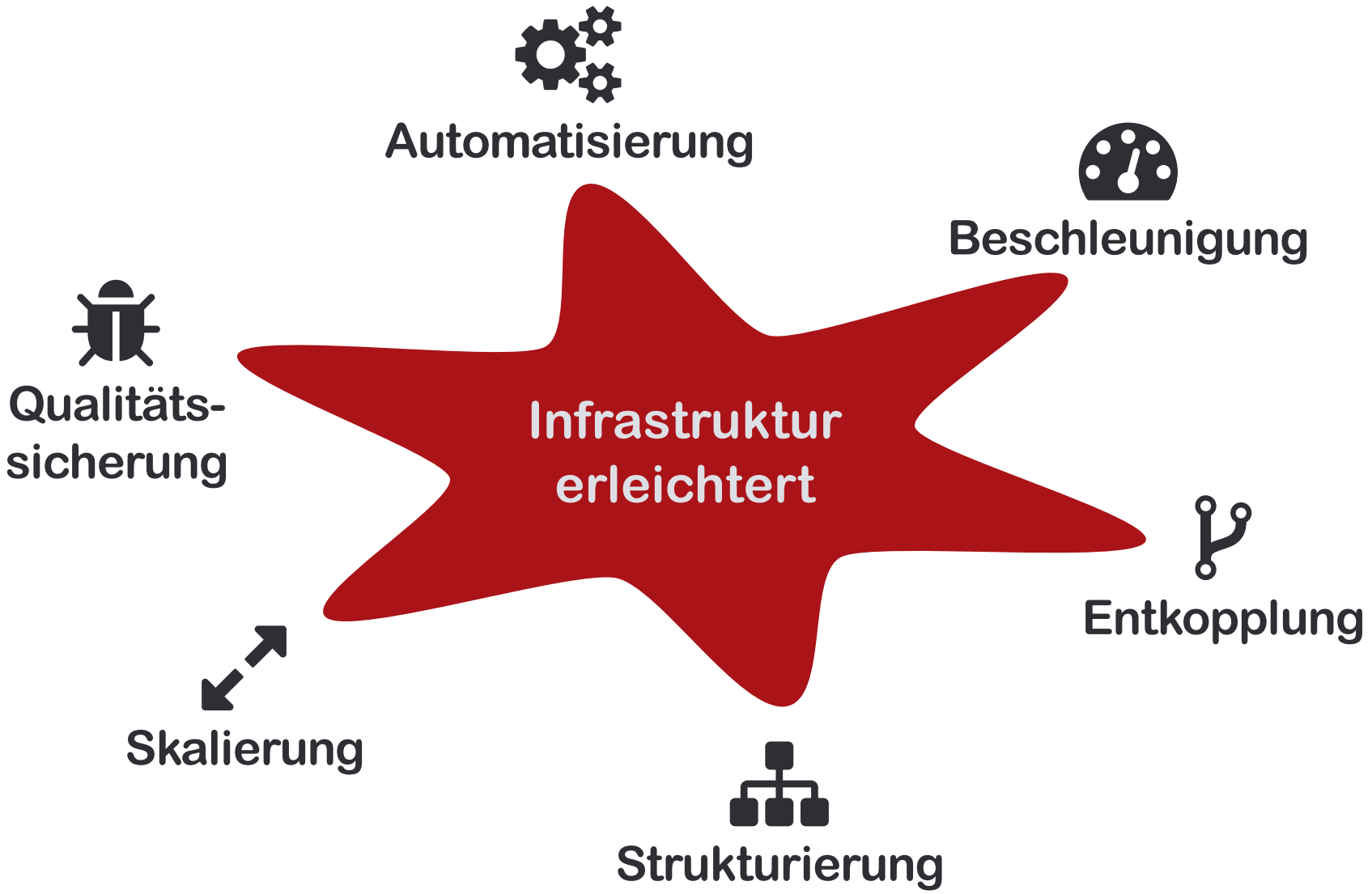
Größeres Projekt

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:
  Verwendung über längere Zeit bzw. von anderen
- ☛ 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
- eine Motivation hinter Programmierung als solcher

Beschleunigung

- Routineaufgaben automatisierbar und so beschleunigbar
- Formalisierung sorgt für Konsistenz und Fokussierung.
- Der Kopf ist frei für das Wesentliche.

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

- Infrastruktur skaliert, Lebenszeit eines Entwicklers nicht.
- essentiell für größere Projekte/bei mehreren Entwicklern
- Entwickler müssen nicht 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?

Automatisierung ermöglicht Fokussierung auf das Wesentliche.

> *It is a profoundly erroneous truism, repeated by all copybooks and by eminent people when they are making speeches, that we should cultivate the habit of thinking of what we are doing. The precise opposite is the case. Civilization advances by extending the number of important operations which we can perform without thinking about them. Operations of thought are like cavalry charges in a battle—they are strictly limited in number, they require fresh horses, and must only be made at decisive moments.*
>
> – Alfred North Whitehead

- Das ist *kein* Plädoyer, nicht zu denken...
- wiederkehrende Abläufe durchdenken und formalisieren, um den Kopf für die wichtigen Dinge frei zu haben

Whitehead: *An Introduction to Mathematics*, Dover, Mineola 2017 (orig. 1911)

Minimale Infrastruktur für jede Auswertungsroutine

- Versionskontrolle
- eindeutige Versionsnummer

Hilfreich und dringend angeraten

- Dokumentation
- Lizenz

☛ Grund für die *minimale* Infrastruktur: Gewährleistung von Wiederverwendbarkeit, Zuverlässigkeit, Überprüfbarkeit

☛ 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 schränkt die individuelle Freiheit ein.
  - Studenten sollen selbst geeignete Konzepte entwickeln.
- Infrastruktur macht Manches erst möglich, u.a.:
  - 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

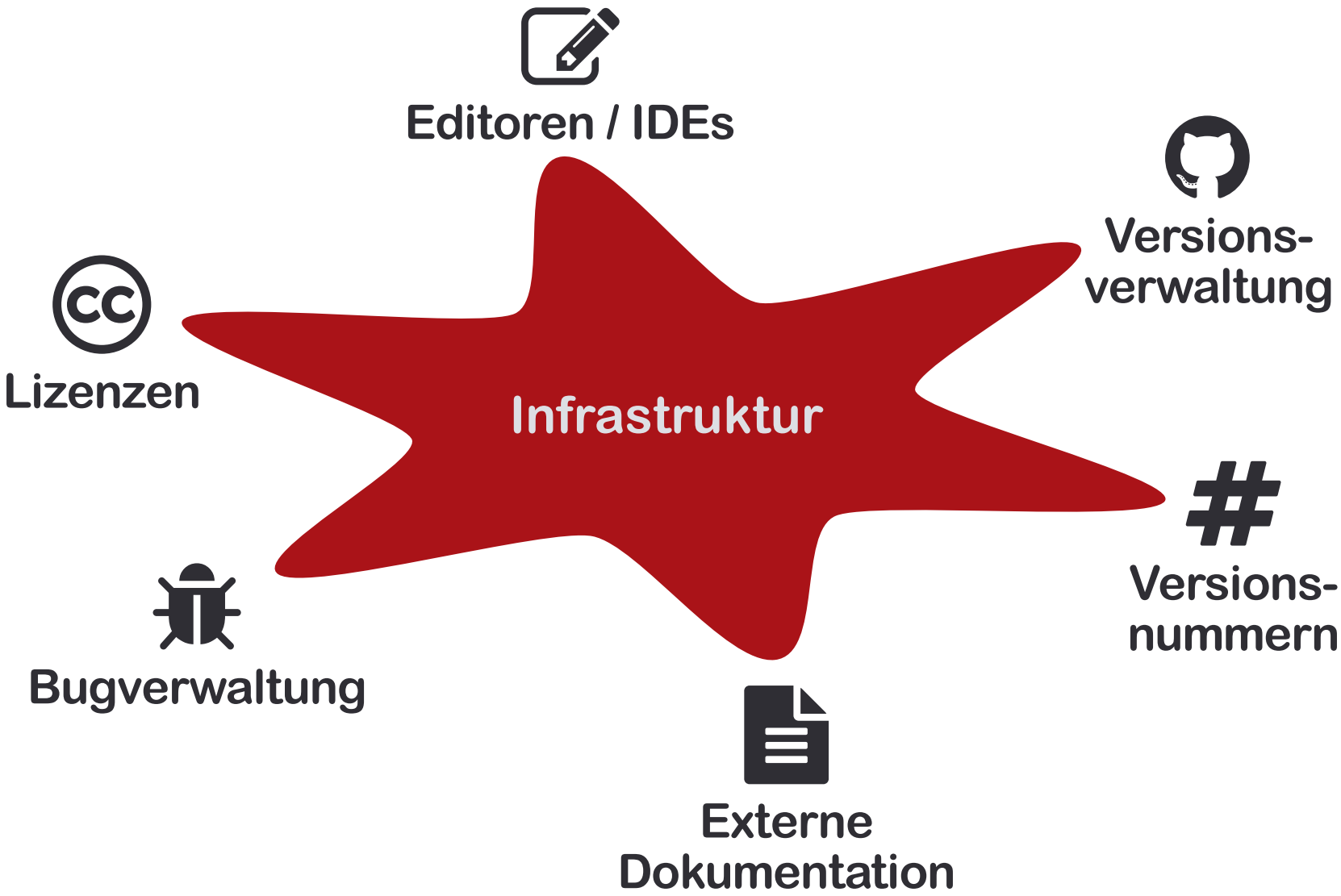
- Arten möglicher Kosten für Infrastruktur
  - Software
  - Hardware (Rechner, zentrale Server)
  - Wartung und Pflege (Personalkosten)
- viele Lösungen als freie Software verfügbar
  - alle hier vorgestellten Aspekte über freie Software lösbar
- integrierte Lösungen über kostenlose Online-Services
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

Infrastruktur schränkt die individuelle Freiheit ein

- Freiheit ist essentiell für die Wissenschaft
  - gilt auch für Datenaufnahme, -ablage, -auswertung
- Problem: Freiheit kollidiert mit Wissenschaftlichkeit
  - oft fehlen Wissen und der notwendige zeitliche Horizont
- Vorgaben sind noch keine Einschränkung der Freiheit
  - immer nur Empfehlungen verknüpft mit der Aufforderung, kritisch zu hinterfragen und weiterzuentwickeln
- eigentliche kreative Aufgabe: Konzeption guter Software
  - entsprechend den Kriterien für Wissenschaftlichkeit
  - erfordert tiefes Verständnis und Abstraktionsvermögen
  - reine Implementierung von Vorgaben ist vergleichsweise einfach

Übersicht über die Infrastruktur

Jeder Aspekt wird separat im Detail behandelt werden



Einzelne Aspekte einer Projekt-Infrastruktur (I)

Editoren/IDEs

- IDEs können die Produktivität immens steigern.
- zwei Gründe für IDEs: Refactoring, automatisierte Tests
- Einarbeitung oft zeitintensiv

Versionsverwaltung (VCS)

- erlaubt jederzeit den Rückgriff auf beliebige vorherige Versionen einer einzelnen Routine/des ganzen Projektes
- befreiend für den Entwickler (Sicherheitsnetz!)
- immenser Einfluss auf die Qualität von Software

Versionsnummern

- eineindeutig für jede Version einer Routine
- *per se* unabhängig von der Versionsverwaltung
- essentiell für Reproduzierbarkeit und damit Wissenschaftlichkeit

Einzelne Aspekte einer Projekt-Infrastruktur (II)

- (externe) Dokumentation
  - Konzepte, Anforderungsanalysen, Coding Conventions etc.
  - Alles, was nicht sinnvoll direkt im Quellcode ablegbar ist
  - Minimum: Warum sollte ich das Programm wofür einsetzen?
- Bugverwaltung
  - formalisiert Fehlerberichte an die Entwickler
  - eindeutige Bezeichner erleichtern Bezugnahme
  - räumliche und zeitliche Trennung von Entwicklern und Anwendern
- Lizenzen und Urheberrechte
  - Quellcode unterliegt *per se* dem Urheberrecht.
  - Wiederverwendbarkeit formal nur bei klarer Rechtslage/Lizenz
  - im akademischen Kontext oft vernachlässigt

Infrastruktur und wissenschaftliche Datenanalyse

Welcher Aspekt benötigt welche Infrastruktur?

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

– Editor/IDE, – Versionsverwaltung, # – Versionsnummern, – Dokumentation, – Bugverwaltung, – Lizenzen

☛ Zahl der Haken ist *kein* Maß für Wichtigkeit oder Rangfolge!

Infrastruktur und wissenschaftliche Datenanalyse

Welcher Aspekt benötigt welche Infrastruktur?

- Wiederverwendbarkeit
  - Refactoring erhöht die Lesbarkeit und Qualität
  - Versionsverwaltung organisiert
  - Konzepte und Zielstellung sollten dokumentiert sein
  - Nutzungsrechte sollten klar sein
- Selbstdokumentation
  - ohne Versionsnummern keine Selbstdokumentation
- Zuverlässigkeit
  - Refactoring erhöht die Lesbarkeit und Qualität
  - Fehlerbehebung sollte strukturiert stattfinden
- Überprüfbarkeit
  - Refactoring erhöht die Lesbarkeit
  - Konzepte und Zielstellung sollten dokumentiert sein

Infrastruktur und wissenschaftliche Datenanalyse

Welcher Aspekt benötigt welche Infrastruktur?

- Nutzerfreundlichkeit
  - Zielstellung und Nutzung sollten lebensnah dokumentiert sein
  - strukturierter Umgang mit Fehlern wirkt vertrauensbildend
  - eindeutige Lizenzen schaffen Klarheit
- Erweiterbarkeit
  - Refactoring erhöht die Lesbarkeit und Modularität
  - Versionsverwaltung befreit und sichert Bestehendes ab
  - Schnittstellen und Konzepte sollten dokumentiert sein
  - eindeutige Lizenzen schaffen Klarheit
- Reproduzierbarkeit
  - Jede verwendete Version der Software muss rekonstruierbar sein
  - Versionsnummern müssen automatisiert mitgeschrieben werden

Weitere Arten von Infrastruktur

Eine unvollständige Übersicht

- Automatisierte Tests
  - zentraler Aspekt für zuverlässigen Code
- Codeanalyse-Werkzeuge
  - statische und dynamische Codeanalyse
  - stark von der verwendeten Programmiersprache abhängig
- Generatoren für Dokumentation aus dem Quellcode
  - für Entwicklerdokumentation geeignet
  - setzt Disziplin bei der Codedokumentation voraus
  - ersetzt nicht die konzeptionelle und Nutzerdokumentation
- 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, . . .
  - Teilnahme sollte als „Investition in die Zukunft“ gesehen werden
- „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.
- Richtig eingesetzt kann Infrastruktur 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.