

Wissenschaftliche Softwareentwicklung

8. (externe) Dokumentation

Till Biskup

Physikalische Chemie

Universität Rostock

10.11.2023





- ❏ Grundlegende Ideen und Konzepte lassen sich schwer durch den Quellcode selbst dokumentieren.
- ❏ Dokumentiert werden sollte das Was und Warum (bzw. Was warum nicht). Das Wie beantwortet der Quellcode.
- ❏ Eine minimale externe Dokumentation (README, INSTALL, erste Schritte) ist für größere Projekte unumgänglich.
- ❏ Einfach nutzbare Dokumentationswerkzeuge helfen, Dokumentation und Quellcode synchron zu halten.
- ❏ Externe Dokumentation ist essentieller Bestandteil von Software zur wissenschaftlichen Datenauswertung.

“ *You can download our code from the URL supplied.
Good luck downloading the only postdoc
who can get it to run, though*

– Ian Holmes auf Twitter, 8. Jan. 2013

Warum ist Dokumentation wichtig?


Vorurteile gegenüber Dokumentation







Arten von Dokumentation

Probleme mit (externer) Dokumentation

(minimale) Anforderungen an Software zur Datenauswertung

- wiederverwendbar, zuverlässig, überprüfbar

						
 wiederverwendbar	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/>
 selbstdokumentierend			<input checked="" type="checkbox"/>			
 zuverlässig	<input checked="" type="checkbox"/>				<input checked="" type="checkbox"/>	
 überprüfbar	<input checked="" type="checkbox"/>			<input checked="" type="checkbox"/>		
 nutzerfreundlich				<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
 erweiterbar	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/>
 reproduzierbar		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			

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

- geringe durchschnittliche Verweildauer von Mitarbeitern
 - maximal ca. fünf Jahre (Doktorarbeit)
 - meist keine langfristigen Mitarbeitenden, die Codepflege betreiben
- Ziel: schnell funktionierendes Programm
 - Nachhaltigkeit höchstens zweitrangig
 - Neuschreiben oft einfacher (und schneller), als Code zu verstehen, weiterzuverwenden, zu erweitern
- ☛ viel zu viel schlechter Code in den Wissenschaften
 - mangelhafte Dokumentation (auf allen Ebenen)
 - erschwert (unnötig) die Nachvollziehbarkeit
- ☛ mangelnde Effizienz
 - Code wird (notgedrungen) viel zu oft neu geschrieben.

Warum ist Dokumentation wichtig?

Eine Liste guter Gründe



- Absichten/Konzepte nur schwer im Code dokumentierbar
 - Ziel des Projekts, Fähigkeiten
 - Grund für die verwendete Architektur
- Nachweis der Urheberschaft an Gedanken/Konzepten
 - grundlegende Gedanken frühzeitig schriftlich festhalten
- Nachvollziehbarkeit der Entwicklung eines Konzeptes
 - setzt Zugang zu alten Versionen (z.B. VCS) voraus
- Voraussetzung für Bewältigung größerer Projekte
 - externe Projektdokumentation (Details folgen)
- Gedankenstütze
 - besonders hilfreich für Programmierung „nebenher“

Warum ist Dokumentation wichtig?

Was soll dokumentiert werden?



- „Was“ und „Warum“ dokumentieren
 - Anforderungsanalyse
 - Übersicht über grundlegende Architektur
 - Begründung konkreter Entscheidungen (z.B. Sprache)
- konkretes „Wie“ im Quellcode selbst
 - „Der Quellcode ist das Design.“
 - (externe) Dokumentation nicht zu detailliert
- nicht gewählte Alternativen
 - bewahrt die Übersicht
 - vermeidet wiederholte (Fehl-)Entwicklungen
- ☛ Dokumentation oft parallel zur Implementierung
 - kein Problem, solange dokumentiert wird
- ☛ Dokumentation in der Wissenschaft *nicht* optional!

Jack R. Reeves: „*The Source Code Is the Design*“, 1992

Warum ist Dokumentation wichtig?

Vorurteile gegenüber Dokumentation

Arten von Dokumentation

Probleme mit (externer) Dokumentation

Dokumentation ist die Zeit nicht wert, die sie benötigt.

- Verständnis kostet viel mehr Zeit.
 - Code wird viel häufiger gelesen als geschrieben.
 - Hauptentwickler meist größter Nutznießer
- Dokumentation skaliert – Entwicklerlebenszeit nicht.
 - Dokumentation muss nur einmal geschrieben werden.
 - Persönliche Einweisung in die Nutzung skaliert nicht.
- genauso wichtig wie der Methodenteil einer Arbeit
 - wird vermutlich (leider) genauso wenig wertgeschätzt . . .
 - mustergültiges Beispiel: Gregor Mendels Originalpublikation
- zwingt zum Nachdenken über den Code
 - Erst überlegen, was das Programm können soll.
 - Gedanken/Ideen *vorher* schriftlich festhalten erspart Arbeit.

Dokumentation ist (zu) zeitaufwendig.

- komplettes Nutzerhandbuch ist tatsächlich aufwendig
 - eher seltener Anwendungsfall
 - Fokus auf Konzepte und große Linien (wenig Änderung)
 - Wahl guter Werkzeuge entscheidend für den Aufwand
- Dokumentation nahe am Code Frage von Disziplin und Übung
 - Übung: selbstdokumentierender Code ultimatives Ziel
 - Disziplin: synchrone Änderung von Dokumentation und Code
- Entwicklerdokumentation weitestgehend automatisierbar
 - setzt Kenntnis der Werkzeuge voraus
 - erfordert Disziplin in der Umsetzung
 - Details in späterer Vorlesung („Dokumentation im Code“)
- konzeptionelle Dokumentation strukturiert und ist notwendig

Dokumentation ist meist unvollständig und veraltet.

- (leider) häufig beobachtbare Tatsache
 - oft fehlt grundlegende Information (Ziel, Fokus, Installation)
 - Frage von Bewusstsein, Disziplin und Kenntnis der Werkzeuge
 - Dokumentation skaliert, Lebenszeit nicht
- Hinweise für Nutzerhandbücher u.ä.
 - großes Aktualisierungsintervall (größere Versionssprünge)
 - vorab Gliederung erstellen
 - ggf. nur Abschnitte mit Inhalt veröffentlichen
- anders und besser statt nicht dokumentieren
 - selbstdokumentierender Code
 - feste Strukturen und Vorlagen (unterstützt durch Editor/IDE)
 - klare, eingeübte Abläufe

Warum ist Dokumentation wichtig?

Vorurteile gegenüber Dokumentation

Arten von Dokumentation

Probleme mit (externer) Dokumentation

- unterschiedliche Kategorisierung möglich
 - Adressaten (Nutzer, Administratoren, Entwickler)
 - Ort (extern, neben dem Quellcode, im Quellcode)
 - äußere Form (elektronisch/gedruckt; knapp/umfassend)
- Kontext wissenschaftlicher Software
 - theoretische Abhandlungen
 - Nutzer- und Entwicklerdokumentation
 - Kommentare im Quellcode
 - selbstdokumentierender Code



externe Dokumentation

Dokumentation, die nicht direkt im Quellcode vorliegt, sondern in separaten Dateien neben oder getrennt vom Quellcode

- Charakteristika
 - höchster Abstraktionsgrad von Dokumentation
 - feste Form (These, Weißbuch, Bericht, Publikation)
 - zusammenhängende Darstellung inkl. Literatur
- Vorteile
 - beschreibt wissenschaftliche Ziele und Herkunft des Codes
 - meist hohe Qualität und gute Übersicht
 - (i.d.R.) begutachtet
 - (i.d.R.) zitierfähig
- Nachteile
 - Erstellung aufwendig
 - statisch
 - (meist) keine Beschreibung der konkreten Implementierung (keine Entwicklerdokumentation)

- kurze Einführung (README)
 - obligatorisch für jedes Projekt
 - beschreibt Zielstellung und Fähigkeiten des Projekts (kurz!)
- Installationshinweise (INSTALL)
 - insbesondere bei Quellcode-Distributionen
 - Verweis auf Abhängigkeiten und Voraussetzungen
- Beispiele
 - gut gewählt und hinreichend beschrieben
 - möglichst getestet/testbar
- Beschreibung der Nutzerfunktionen
 - Unterscheidung: öffentliche API/Entwicklerdokumentation
 - alle Nutzerfunktionen sauber beschreiben
- Lizenz (LICENSE)

- Details zur Implementierung
 - „Was“ und „Warum“
 - „Was warum nicht“:
betrachtete, nicht gewählte Alternativen beschreiben
 - nicht zu detailliert: „Der Code ist das Design.“
- (automatisch generierte) API-Dokumentation
 - wenig Mehrarbeit; Voraussetzung: Disziplin beim Programmieren
 - Beschreibung der API als Kommentarblock direkt im Code
 - formatierte Ausgabe vollständig automatisierbar
 - wird in späterer Vorlesung ausführlicher behandelt
- Projektdokumentation
 - Details folgen . . .

Bestandteile einer externen Projektdokumentation

- Anforderungsanalyse
 - kann sich ändern (gerade am Anfang)
- Architektur/High-Level-Design
 - nicht gewählte Alternativen benennen/begründen
- *Roadmap*
 - bei größeren Projekten bzw. limitierten Ressourcen
 - (flexible) Priorisierung einzelner Aufgaben
- *Changelog*
 - wichtig für Kompatibilität
- Konventionen
 - Beispiel: *Coding Conventions*
 - konsequente Umsetzung wichtiger als konkreter Inhalt

Warum ist Dokumentation wichtig?

Vorurteile gegenüber Dokumentation

Arten von Dokumentation

Probleme mit (externer) Dokumentation

- **Asynchronität zwischen Code und Dokumentation**
 - Synchroner Änderung erfordert Disziplin.
 - veraltete Dokumentation ggf. schlimmer als fehlende
 - **Zeitaufwand für die Erstellung von Dokumentation**
 - externe Dokumentation vom Code getrennt
 - Notwendige Vorausplanung ist Teil der Dokumentation.
 - nicht zu viele Details: „Der Code ist das Design.“
 - **Unvollständige oder falsche Dokumentation**
 - großes Problem vieler „Nebenher-Projekte“
 - Ursache oft mangelnde Disziplin
 - falsche Dokumentation ggf. schlimmer als fehlende
- ☛ einfache, strukturierte Ansätze zur externen Dokumentation

- niederschwellig
 - Fokus: einfache Nutzbarkeit
 - Bsp.: Wiki
- Struktur/Konventionen
 - autoritative Antworten auf viele kleine alltägliche Fragen
 - hilft bei der Ausbildung von Routine
 - Konventionen sind nicht in Stein gemeißelt . . .
- Vorlagen (*Templates*)
 - befreiend, beschleunigend, vereinheitlichend
- klare Abläufe
 - Konventionen für den Akt des Dokumentierens
 - Automatisierung, ggf. Checklisten

- Automatisierung
 - vereinfacht und beschleunigt
 - sorgt für Konsistenz
- ein Ort
 - Informationen an einem Ort in einem (Meta-)Format
 - verhindert Inkonsistenzen
 - zentraler Speicherort mit gutem Zugriff (VCS, online)
- Disziplin
 - hilfreich: einfache Lösungen und Struktur/Konventionen
- Vorausplanung
 - Gliederung der gesamten Dokumentation
 - Priorisierung
 - ggf. nur Abschnitte mit Inhalt veröffentlichen

Beispiel der Dokumentation eines realen Projektes

Kombinierte Nutzer-/Entwickler-/API-Dokumentation



Changelog
Roadmap
ASpecD dataset format (adf)
Dataset structure

API documentation

- aspect.analysis module
- aspect.annotation module
- aspect.dataset module
- aspect.exceptions module
- aspect.history module
- aspect.infofile module
- aspect.io module
- aspect.metadata module
- aspect.model module
- aspect.plotting module
- aspect.processing module
 - Concrete processing steps
 - Writing own processing steps
- Module documentation
 - aspect.report module
 - aspect.system module
 - aspect.table module
 - aspect.tasks module

Examples

For convenience, a series of examples in recipe style (for details of the recipe-driven data analysis, see `aspect.tasks`) is given below for how to make use of this class. The examples focus each on a single aspect.

In the simplest case, just invoke the baseline correction with default values:

```
- kind: processing
  type: BaselineCorrection
```

In this case, a zeroth-order polynomial baseline will be subtracted from your dataset using ten percent to the left and right, and in case of a 2D dataset, the baseline correction will be performed along the first axis (index zero) for all indices of the second axis (index 1).

Of course, often you want to control a little bit more how the baseline will be corrected. This can be done by explicitly setting some parameters.

Suppose you want to perform a baseline correction with a polynomial of first order:

```
- kind: processing
  type: BaselineCorrection
  properties:
    parameters:
      order: 1
```

If you want to change the (percental) area used for fitting the baseline, and even specify different ranges left and right:

```
- kind: processing
  type: BaselineCorrection
  properties:
    parameters:
```



- ❏ Grundlegende Ideen und Konzepte lassen sich schwer durch den Quellcode selbst dokumentieren.
- ❏ Dokumentiert werden sollte das Was und Warum (bzw. Was warum nicht). Das Wie beantwortet der Quellcode.
- ❏ Eine minimale externe Dokumentation (README, INSTALL, erste Schritte) ist für größere Projekte unumgänglich.
- ❏ Einfach nutzbare Dokumentationswerkzeuge helfen, Dokumentation und Quellcode synchron zu halten.
- ❏ Externe Dokumentation ist essentieller Bestandteil von Software zur wissenschaftlichen Datenauswertung.