Wissenschaftliche Softwareentwicklung

Bausteine einer rechnergestützten Datenauswertung, die den Kriterien der Wissenschaftlichkeit genügt

8. (externe) Dokumentation

Till Biskup
Physikalische Chemie
Universität Rostock
07.11.2024

Zentrale Aspekte

- 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

https://twitter.com/ianholmes/status/288689712636493824 **(Zugriff am 18.10.2023)**

Übersicht

Warum ist Dokumentation wichtig?

Vorurteile gegenüber Dokumentation

Arten von Dokumentation

Probleme mit (externer) Dokumentation

(minimale) Anforderungen an Software zur Datenauswertung

- wiederverwendbar, zuverlässig, überprüfbar

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

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

Die Situation in den Wissenschaften

Dokumentation wird vernachlässigt – mit hohen Kosten

- 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

Übersicht

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

Übersicht

Warum ist Dokumentation wichtig?

Vorurteile gegenüber Dokumentation

Arten von Dokumentation

Probleme mit (externer) Dokumentation

Arten von Dokumentation

Eine erste Übersicht

- 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)

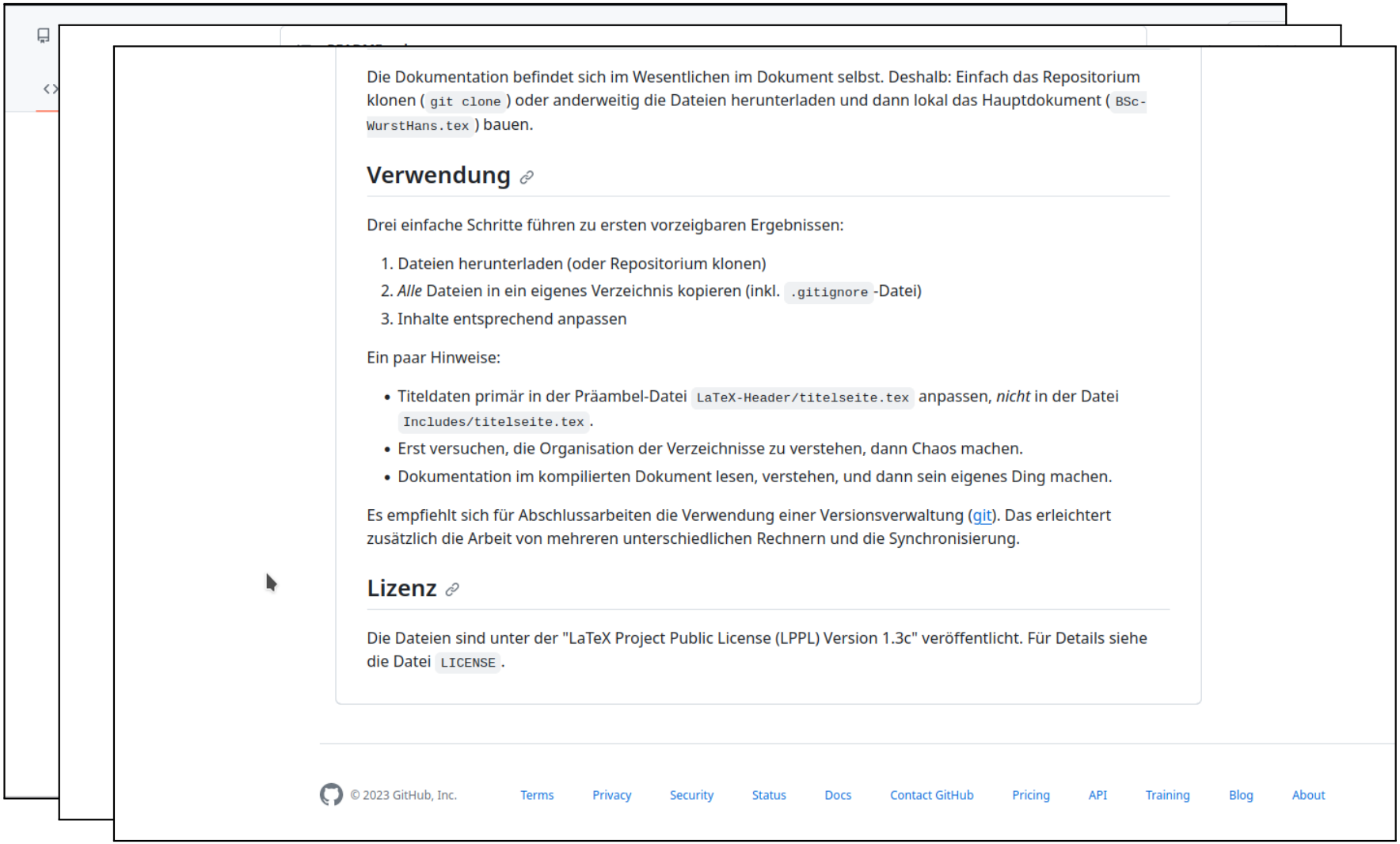
Nutzerdokumentation

Einfach zugängliche Informationen für den Anwender

- 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`)

Nutzerdokumentation

Die Vorteile strukturierter `README`-Dateien für *jedes* Projekt



Die Dokumentation befindet sich im Wesentlichen im Dokument selbst. Deshalb: Einfach das Repositorium klonen (`git clone`) oder anderweitig die Dateien herunterladen und dann lokal das Hauptdokument (`BSc-WurstHans.tex`) bauen.

Verwendung

Drei einfache Schritte führen zu ersten vorzeigbaren Ergebnissen:

1. Dateien herunterladen (oder Repositorium klonen)
2. *Alle* Dateien in ein eigenes Verzeichnis kopieren (inkl. `.gitignore`-Datei)
3. Inhalte entsprechend anpassen

Ein paar Hinweise:

- Titeldaten primär in der Präambel-Datei `LaTeX-Header/titelseite.tex` anpassen, *nicht* in der Datei `Includes/titelseite.tex`.
- Erst versuchen, die Organisation der Verzeichnisse zu verstehen, dann Chaos machen.
- Dokumentation im kompilierten Dokument lesen, verstehen, und dann sein eigenes Ding machen.

Es empfiehlt sich für Abschlussarbeiten die Verwendung einer Versionsverwaltung (git). Das erleichtert zusätzlich die Arbeit von mehreren unterschiedlichen Rechnern und die Synchronisierung.

Lizenz

Die Dateien sind unter der "LaTeX Project Public License (LPPL) Version 1.3c" veröffentlicht. Für Details siehe die Datei `LICENSE`.

© 2023 GitHub, Inc. Terms Privacy Security Status Docs Contact GitHub Pricing API Training Blog About

`https://github.com/tillbiskup/latex-template-abschlussarbeit/`

Entwicklerdokumentation

Strukturierte Information zu den Details der Implementierung

- 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 . . .

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

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

Übersicht

Warum ist Dokumentation wichtig?

Vorurteile gegenüber Dokumentation

Arten von Dokumentation

Probleme mit (externer) Dokumentation

Probleme mit (externer) Dokumentation

Gängige Probleme

- Asynchronität zwischen Code und Dokumentation
  - Synchrone 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

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

Probleme mit (externer) Dokumentation

Mögliche Lösungsansätze

- 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

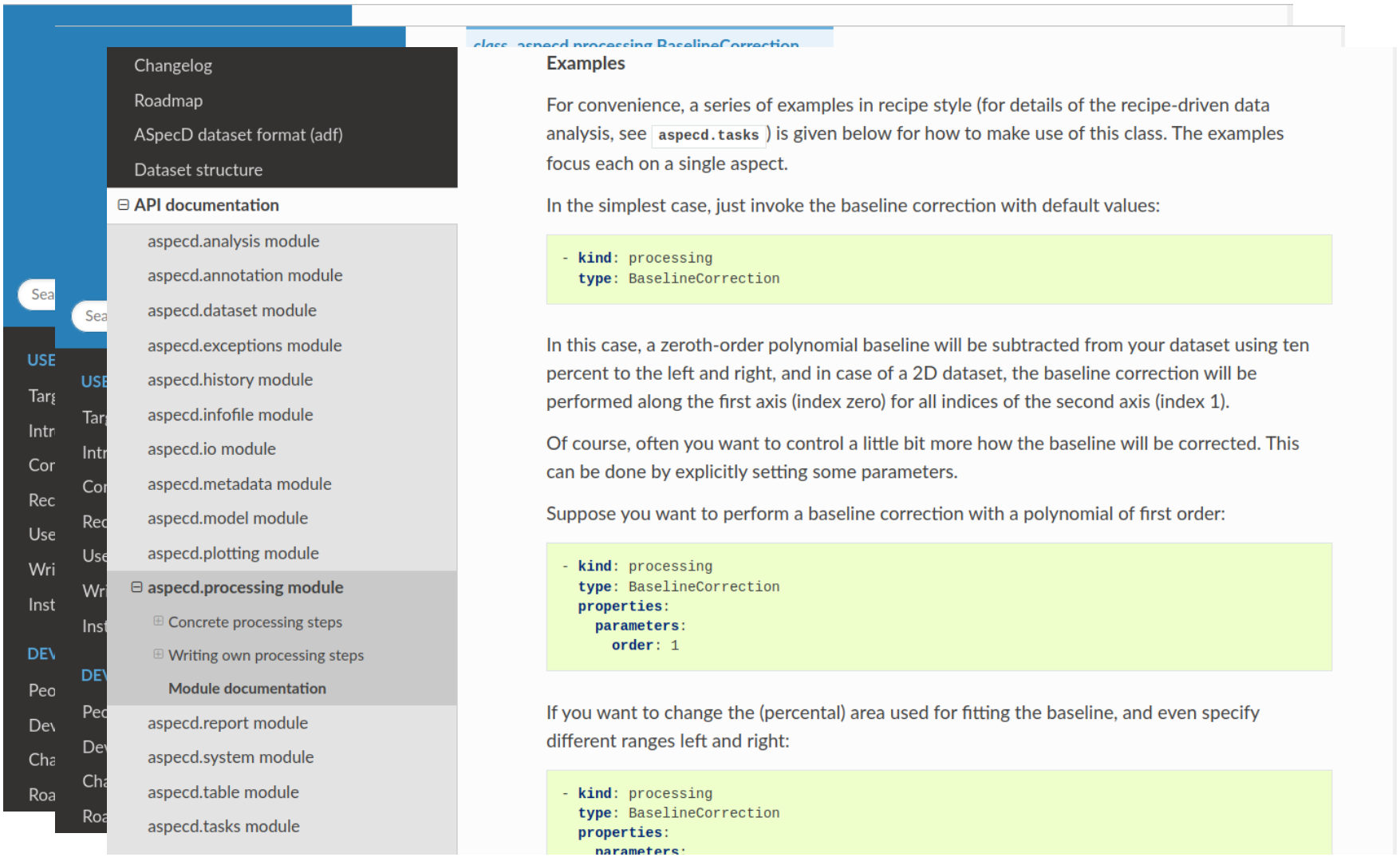
Probleme mit (externer) Dokumentation

Mögliche Lösungsansätze

- 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



https://docs.aspecd.de/

Zentrale Aspekte

- 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.