



# Software-Entwicklung mit AI-Unterstützung?

Dr. Ralf Schlatterbeck  
Open Source Consulting

Email: [office@runtux.com](mailto:office@runtux.com)  
Web: <http://www.runtux.com>  
Tel. +43/650/621 40 17



## Contents

Aider	4
Sicherheit und Datenschutz	7
Beispiel: Meta-Kommando /add	8
Beispiel: Prompt mit Antwort	9
API Rate Limit	10
Coding conventions: AI cannot count	11
No real understanding of the code	12
Komplexe Änderung	13
Linear-time median finding algorithm	15
Steiniger Weg	16



## Contents

Python, erster Aufruf	17
Neuer Check aus einem Buch	21
Writing a Test	23
Bibliography	26



## Aider

- Python-Programm **Aider**
- Installationsscript installiert eigenen Python-Interpreter
- ... ich habe bessere Erfahrungen mit
 

```
pip install aider-chat
```

 statt
 

```
pip install aider-install
```

```
./aider-install
```
- Debian Trixie: 2. Variante (Python-Version)
- kommt mit Plugins für diverse AIs
- ... unter anderem für **OpenRouter**
- hier: direktes Interface zu Anthropic **Claude**



- Wird in einem Git-Repo gestartet und macht direkt Git commits
- Das Kommando `git rebase` wird zu einem guten Bekannten
- Programm öffnet ein Chat-Fenster
- Interaktion mit der AI läuft in diesem Fenster
- Meta-Kommandos um die Auswahl der Dateien, die für die AI sichtbar sind, zu definieren
- Längere Prompts schreibt man im Editor der Wahl
- Switch zwischen Architektur-Diskussion und direkter Code-Erzeugung



- Experimente mit Claude Sonnet 3.7
- ... und mit Claude Sonnet 4, noch nicht 4.1
- „Weak model“ 3.5 Haiku (for non-reasoning tasks like git commit)
- Schwerpunkt liegt auf *Änderung von bestehender Software*
- Aider schreibt Logs in `.aider.input.history` und `.aider.chat.history.md`
- Die folgenden Ausführungen sind Anekdoten, ein konsistentes Bild zeichnet sich für mich noch nicht ab



- Alle Experimente betreffen Open Source Software die ich selbst maintainen
- Daher habe ich keine Angst dass die AI Dinge lernt die sie nicht wissen sollte
- Alle Programme sind sowieso auf Github und wurden vermutlich schon verwendet um die AI von Microsoft zu trainieren
- Ich würde nicht empfehlen, proprietäre Software mit AI-Unterstützung zu schreiben
- Wir können vermuten dass die Eingaben weiterverwertet werden



```
>>> add include/pgapack.h
<<< Add file to the chat?
    (Y)es/(N)o/(D)on't ask again [Yes]: y
<<< I've added the include/pgapack.h file to the chat. This file
contains all the constant and structure definitions for PGAPack, as well
as all function declarations. It includes definitions for data types, error
printing constants, boolean values, debug levels, optimization direc-
tions, and many other constants and structures used throughout the
library. The file also contains all the function prototypes organized by
source file.

>>> /add source/pop.c
<<< > Added source/pop.c to the chat
```



## Beispiel: Prompt mit Antwort

```
>>> The file test/unit/pop_test.c... It currently has a test of the
function ranking but doesn't test the resulting rank variable in the indi-
viduals starting with start. We want to add this to the test. The idea is
to replace the current O(n**2) ranking algorithm with something better
(we do that later). Before that we need to establish a baseline with a
test of the ranking. ...
<<< I understand you want to add a test for the ranking function in
test/unit/pop_test.c to verify the rank values assigned to indi-
viduals. This will establish a baseline before you replace the current
O(n^2) algorithm with something better.
I'll need to see the contents of test/unit/pop_test.data to pro-
perly implement this test. Could you please add that file to the chat?
```



## API Rate Limit

```
> litellm.RateLimitError: AnthropicException ...
> The API provider has rate limited you. Try again later o
> Retrying in 0.2 seconds...

> litellm.RateLimitError: AnthropicException ...
> The API provider has rate limited you. Try again later o
> Retrying in 0.5 seconds...

1.0 ... 2.0 ... 4.0 ... 8.0 ... 16.0

I see the test data file now. Let me add a test for the 'ranking' function
to verify the rank values assigned to individuals.
```



## Coding conventions: AI cannot count

```
>>> We need to establish coding conventions first, please extract co-
ding conventions from the .c files you know into CONVENTIONS.rst
<<< Based on the provided code files, here are the coding conventi-
ons I've extracted:
```

```
Coding Conventions
=====
```

```
Indentation and Formatting
-----
...
Naming Conventions
-----
```



## No real understanding of the code

```
>>> There are still some hard-coded "3" in test_pop that should use
the new dim parameter. Please observe coding guidelines regarding
new paragraph on spacing before parentheses and brackets.
<<< I'll fix the remaining hard-coded "3" values in the test_pop func-
tion and ensure proper spacing before parentheses and brackets ac-
cording to the coding guidelines. Here's the corrected version:
```

```
...
    double (*pop) [3] = p;
    double (*extreme) [3];
    assert (sizeof (pop2) / 3 / sizeof (double) == npop);
...
```

```
>>> There are still some hard-coded 3 values in test_pop
```



## Komplexe Änderung

- Neues „Non-dominated sorting“ mit mehreren langen Prompts
- Beschreibung aus einem Paper [Jen03] 3 bzw. 4 Bildschirmseiten
- Mit C-artigem pseudo-code
- Ziemlich gutes Ergebnis, aber:

```
/* Sort individuals by first objective .. */
for (i = 0; i < n; i++) {
    for (j = i + 1; j < n; j++) {
        ...
    }
}
```

>>> The initial sorting is  $O(n^2)$ , please use a compare function and a call to qsort instead



## Komplexe Änderung

- Verbessertes non-dominated sorting „Jensen’s Algorithm“ [Jen03]
- Die komplexe Änderung hätte ich ohne Aider vermutlich nicht angefangen
- Das erzeugte Framework ist ziemlich gut
- Einige C-Macros wurden falsch verwendet (Sortierreihenfolge)
- Der Code brauchte noch *einiges* an Debugging
- Aber das Problem ist komplex, es gibt ein eigenes Paper 10 Jahre später das eine Implementierung beschreibt [FGP13]



## Linear-time median finding algorithm

>>> /web <https://rcoh.me/posts/linear-time-median-finding/>  
 <<< I’ll implement a linear-time median finding algorithm to replace the current implementation in the `find_median` function. This will significantly improve the performance of the code.

- Link ist Code in Python, Zielcode ist C
- Code enthält zwei Funktionen die sich *gegenseitig* rekursiv aufrufen
- Das hat Claude absolut nicht kapiert
- Ein Versuch die Aufrufe aufzurollen endet in vier Funktionen die sich dubios gegenseitig aufrufen
- Ich hab dann auf Claude verzichtet



## Steiniger Weg

- Coding-style violations (immer wieder)
- Variablen-Deklarationen in der Mitte des codes
- Mehrere Fälle wo das Ding einen  $O(n^2)$  Algorithmus produziert
- Immer wieder mehrere Anläufe
- Vieles davon habe ich dann von Hand korrigiert
- Es ist unklar ob das an der Programmiersprache C liegt oder an der Claude Version (3.7 vs. 4)
- ... oder einfach an der Komplexität des Problems
- Spätere Versuche mit Claude 4 und Python Code waren erfolgreicher



## Python, erster Aufruf

```
>>> /add contrapunctus/checks.py
<<< I notice that in the checks.py file, there are references to two classes that are used in the [...] list but are not defined in the file:
```

1. Check\_Melody\_laMotte\_Jump
2. Check\_Melody\_Avoid\_Eighth\_Jump
3. Check\_Melody\_Quarter\_Jump
4. Check\_Passing\_Tone

I'll need to implement these classes to complete the file. Let me create the implementations for these missing classes.



## Python, erster Aufruf

- Kommentare waren in Deutsch und wurde automatisch – und ausführlicher – als Docstrings in Englisch übernommen
- Implementierung war gut
- Leider waren einige Features als *Ausnahmen* der bestehenden Checks gedacht
- Das war nicht fertig implementiert weil es noch kein Framework für solche Ausnahmen gab
- Implementierung eines Frameworks für Ausnahmen mit Hilfe von Aider
- Weitgehende Übernahme des generierten Codes



## Neuer Check aus einem Buch

### 18. Durchgänge und Wechselnoten

Durchgänge und Wechselnoten besitzen eine melodische und eine satztechnische Seite: Als melodische Ereignisse bilden sie ‚Nebennoten‘, die jeweils zwischen zwei ‚Hauptnoten‘ liegen: der Durchgang als Ausfüllung eines Terzsprungs, die Wechselnote als Einfügung in eine Tonwiederholung. Satztechnisch wird man sie vor allem dann registrieren, wenn sie dissonieren. Im 16. Jh. sind diese Arten der Ergänzung typischerweise ‚flüchtige‘ Viertelnoten, gehören also in den Bereich der Ornamente, z.B. als einzelne Viertel nach punktierter Halber:

Abbildung 1: Buchseite-Ausschnitt mit Handy fotografiert



## Neuer Check aus einem Buch

18 Durchgänge und Wechselnoten

Lasso: Das Quartett... (Musical notation)

Durchgänge (d) sind die einzelne Viertelnoten allgegenwärtig, Wechselnoten (w) weniger. Diese gehören weit über zu Viertelgruppen, Wechselnoten (w) über dem beliebigen Kolnoten mit dem Viertel, nach dem mit nicht lange nicht sind, am wenigsten nach demjenigen mit einer anderen Wechselnote zum Beispiel der betreffenden Gruppe wie im ersten der folgenden Ausschnitte.

Lasso: Das Quartett... (Musical notation)

Dieser Quartett Lasso die gleiche Wendung mit entsprechender Verschiebung in ihm. Der zweite Abschnitt präsentiert die Verschiebung in der Umkehrung d.h. steigend mit oberen Wechselnoten. Um auf diese zu achten, muss in die entsprechenden Wiederholung, bedarf es schon einiger Geduld bei der Schöpfung der musikalischen Literatur. Offenbar bestand eine gewisse Scheu die Systeme allen heillosig zu setzen. Darum ist generell zu beobachten:

Obere Wechselnoten sind erheblich seltener als untere.

Untere Wechselnoten können zu jedem Zeitpunkt in Kolnoten eingetriggert werden, wie das folgende Alt-Ende eines Maß-Biciniums von Lasso eindrucksvoll zeigt:

Lasso: Das Quartett... (Musical notation)

Sie können isoliert stehen sowie Anfang, Mitte und Ende einer Kolnote bilden. Während die obere Version zwei gewichtige Einschränkungen erfüllt:

Obere Wechselnoten bilden das Ende von Viertelgruppen, die aus mindestens drei steigenden Tönen bestehen.

Abbildung 2: Ganze Buchseite



## Neuer Check aus einem Buch

>>> The scan pictures scan0.png to scan7.png contain a description of a `Harmony_Exception` and should be implemented as a new derived class from `Harmony_Exception`.

<<< Based on the scanned text, I can see this describes „Durchgänge und Wechselnoten“ (passing tones and changing tones). The text explains that these are melodic ornaments that create dissonances but are allowed under specific conditions. Let me implement a new `Harmony_Exception` class for these ornamental dissonances.

- Beschreibung als *gescannte Bilder* mit minimalem Prompt Text
- Daraus ist ein neuer Harmonie-Check zu implementieren
- Sprache des Buches: Deutsch
- Auf Anhieb korrekter, funktionierender Code



## Writing a Test

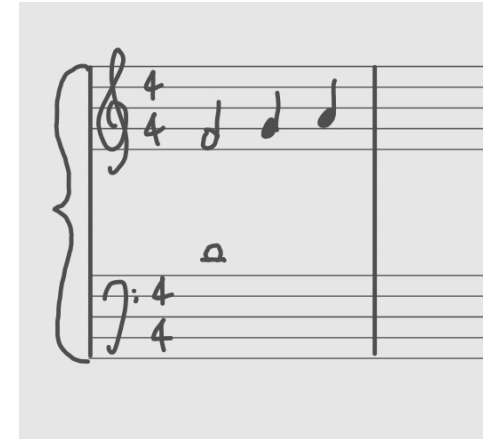


Abbildung 3: Exportierte Grafik



## Writing a Test

>>> The music notation in [pic] should become a regression test for the new `Exception_Harmony_Passing_Tone` class. We need a derived class from `Check_Harmony_Interval` for testing. This should just call the exception classes. For the tests we pass only a single exception class (the one to test). The music notation should be in ABC-notation and use the `Tune.from_iterator` method to parse the ABC. Example ABC notation can be found in `test/de-1.abc`.

<<< Looking at the musical notation in the image, I can see two measures in 4/4 time with treble and bass clefs. I'll create a regression test for the `Exception_Harmony_Passing_Tone` class by:

- Hat fast funktioniert, Noten werden nicht erkannt
- Noten in ABC-Notation manuell ergänzt



## Versuch eines Fazits

- Größere Schwierigkeiten im C-Projekt
- ... mit *deutlich* komplexerer Aufgabenstellung
- Trotzdem eine große Hilfe
- Erzeugtes Framework war gut
- Auch andere Personen hatten Probleme mit dem Algorithmus
- Seltsamkeiten bei rekursivem Code: Aufgabe den in eine andere Sprache zu übersetzen
- Viel Code auf Anhieb korrekt im Python-Projekt
- Unterschied wegen Komplexität? Claude-Version?



## Versuch eines Fazits

- Beschreibung als sehr verzerrtes Foto
  - Deutschsprachiger Input
  - Volles Verständnis von Fachjargon (Musik)!
  - Matchen der Fachjargon-Beschreibung zum Code
  - Noten lesen funktioniert (noch?) nicht
  - Schwer vorherzusagen was funktionieren wird und was nicht
  - Mühsame wiederholte Korrekturen, → Coding-Style
  - Einiges würd' ich dem Teil noch nicht anvertrauen
- Beispiel Kalender (oder Adress-) Synchronisation



## Bibliography

- [FGP13] Félix-Antoine Fortin, Simon Grenier, and Marc Parizeau. Generalizing the improved run-time complexity algorithm for non-dominated sorting. In Christian Blum, editor, *Genetic and Evolutionary Computation GECCO 2013*, pages 615–622. ACM, Amsterdam, The Netherlands, July 2013.
- [Jen03] Mikkil T. Jensen. Reducing the run-time complexity of multiobjective EAs: The NSGA-II and other algorithms. *IEEE*



## Bibliography

*Transactions on Evolutionary Computation*,  
7(5):503–515, October 2003.