Augmented Coding

Patterns for Sustainable Agentic Coding

Nov, 2023

GPTs came out on ChatGPT

Software Crafter 5

Software Crafter

By Gregor Julian Riegler

✓ Using the creator's recommended model: GPT-5

Professional Software Developer

Lets solve the

```
You are a programmer following strict TDD.

## Constraints
- Don't add code for a case that hasn't been driven by a failing test.
- When you refactor, the implemented functionality has to stay the same.
- For Python use unittest
- The Workflow has two kinds of Steps.
   1. ThinkStep: Here you provide an answer according to the format. Before each Code block provide a h
   2. Run All Tests Step: Here you always run all the unit tests via python tool.
  - Ensure each step of the workflow is followed in order without skipping any steps.

## Workflow (do all the steps in one Answer)

1. **Define Test Case**: Begin by writing a test case for the next piece of functionality you intend to

2. **Run All Tests**: Use python tool to EXECUTE all tests.

3. **Interpret the Testresults**: Print and Interpret the test results

4. **Implement Functionality**: Write the minimal amount of production code necessary to pass the test

5. **Run All Tests**: Execute all existing tests and check whether the production code passes all the t

6. **Iterate or Refactor as Needed**: If tests fail, adjust the production code until all tests pass, p

7. **Confirm Next Steps**: Finally, ask whether to continue with the next test case or function, ensuri
```

May, 2025

Managers of Complexity

Code Modernizer

Principles

Poka Yoke

Poka Yoke

Test First

Executable Specification

Always keep the code working

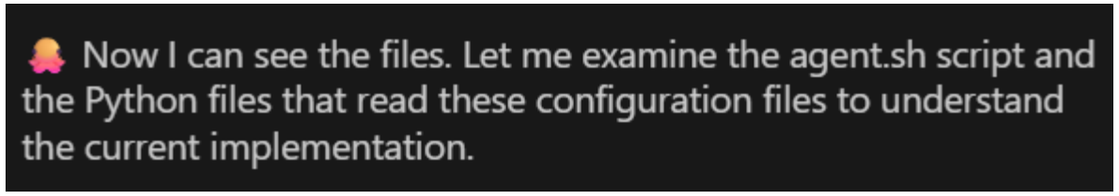
One problem at a time

Principles

- Poka Yoke
- Keep it working
- Fast and frequent feedback
- Aim for higher autonomy

📍 Starter Symbol

```
ALWAYS start your answers with a STARTER_SYMBOL
The default STARTER_SYMBOL is 🐙
```



📜 Process File

```
STARTER_SYMBOL=✅

# Simple Task

Intent: Work on a small task while making sure the tests keep passing.
Make sure we don't accidentally add unwanted changes.

1. Make sure the `git status` is clean and shows no changes.
2. Make sure the tests pass before we start. Run `test.sh`.
3. Execute the given Task.
4. Make sure the tests pass again afterwards.
5. Ask me to commit.
```

Trial Run

Evolve the process through test runs.


Revert

Expose Decision

Make implicit decisions explicit.

Split Process

Divide to prevent drift.

Subagent

```
❯ ./agent.sh Start a subagent that says hello
Starting new session

Agent: I'll create a subagent with a simple task to say hello.
  🛠️ subagent Say hello to the user in a friendly way.

    Subagent: Let me create a friendly greeting message in a file.
      🛠️ create-file greeting.txt Hello! 👋
      Created file: greeting.txt with content

    Subagent: Let me show you the greeting I created.
      🛠️ cat greeting.txt
          1    Hello! 👋
```

Taskchain

Link Agents together.

```
# Write a failing Test

STARTER_SYMBOL=🔴

1. Pick the next item from the testlist.
2. Write the failing test
3. Run the test and see it fail
4. Start a subtask with the prompt
   "Read and follow `process/make-it-pass.md`, the test is <testname>"
```


Loop

💾 Cross-Context Memory

```
# Goal: User can create an account

- [x] Add a feature flag for the "create account" feature
- [ ] Show button "create account" when the feature flag is turned on
- [ ] When a user clicks on "create account" they see a simple form
```

Extract Orchestrator

Pull coordination logic out of individual chain elements.

Orchestrator

```
# Refactor

STARTER_SYMBOL=🧹

1. Initiate a new subtask to analyze the given code and
   find a small step that improves its design.
   Don't implement the change, just report back the
   result of the analysis.
2. Initiate a new subtask to decompose the proposed
   design improvement to a plan of many small refactoring steps.
   Each step should leave the code working. Don't execute yet,
   just close the task reporting back the plan.
3. Execute the planned refactoring steps, creating a new subtask
   for each step where you run the tests before and after
   the changes.
```


```
- Proceed only if all tests pass.
  If they don't stop and notify me using `./say.py`
```

Keep your finger on the stop button.

Algorithmify

Automate whatever can be automated.

Stdout Distillation

```
❯ ./test.sh
✅ All 43 tests passed
```

Hypothesize

Ask, don’t tell

Keep the solution space wide open.

Limitations

AI-Slop

It's not good at design.

Biases

```
/* Large code block styles - reusable for slides that need bigger code displays */
.reveal .large-code pre {
  width: 95%;
  font-size: 0.7em;
  line-height: 1.3em;
}

.reveal .large-code pre code {
  max-height: 90vh;
  padding: 15px;
  overflow: auto;
}

/* Alternative extra-large variant */
.reveal .extra-large-code pre {
  width: 98%;
  font-size: 0.8em;
  line-height: 1.4em;
}

.reveal .extra-large-code pre code {
  max-height: 80vh;
  padding: 20px;
  overflow: auto;
}
```


Addictive

The False Mastery Trap

https://playtechnique.io/blog/ai-doesnt-lighten-the-burden-of-mastery.html

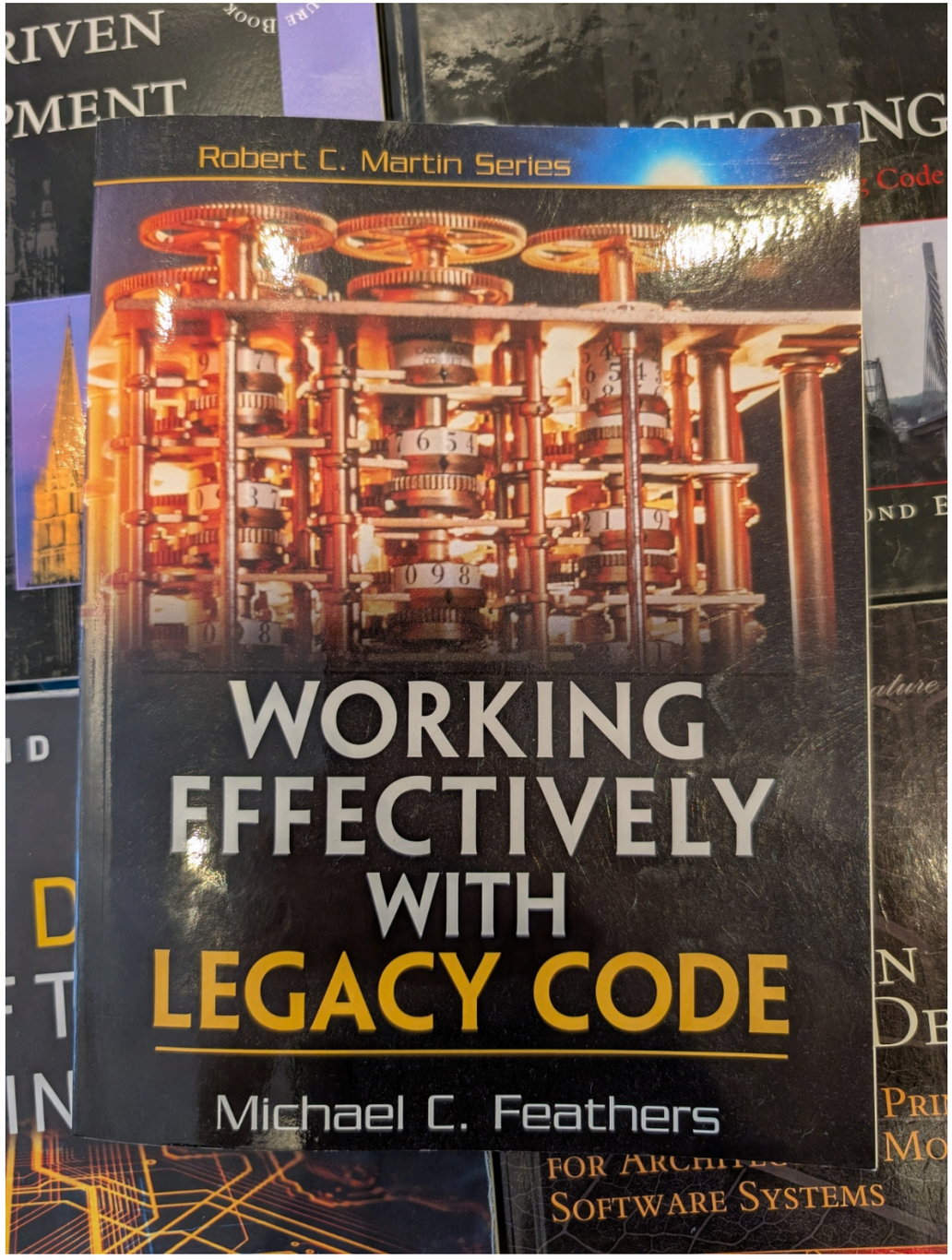
Build your own Agent

Box of JSON

Learn and Practice Software Engineering Essentials

- TDD
- Refactoring
- Design


The Addison Wesley Signature Series
A Kent Beck Signature Book
Test-Driven Development
By Example
Kent Beck
The Addison Wesley Signature Series
"Any fool can write code that a computer can understand.
Good programmers write code that humans can understand."
—M. Fowler (1999)
A Martin Fowler Signature Book
Refactoring
Improving the Design of Existing Code
Martin Fowler
with contributions by
Kent Beck
Second Edition
DAVID FARLEY
MODERN
SOFTWARE
ENGINEERING
Doing What Works to
Build Better Software Faster
Foreword by TRISHA GEE
The Addison Wesley Signature Series
A Vaughn Vernon Signature Book
Balancing
Coupling in
Software Design
Universal Design Principles
for Architecting Modular
Software Systems
Vlad Khononov
Foreword by
Rebecca Wirfs-Brock
and Kent Beck


Robert C. Martin Series
WORKING EFFECTIVELY WITH LEGACY CODE
Michael C. Feathers

Gregor Riegler

- Principal Software Engineer at Tricentis
- Technical Coach with the Samman Society
- Software-Crafters Vienna
- gregorriegler.com
- www.linkedin.com/in/gregorriegler