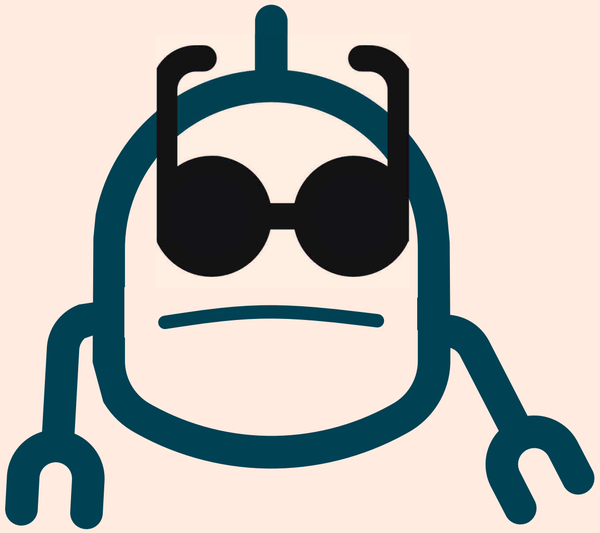


INNOQ AGENTIC SOFTWARE ENGINEERING NIGHT #2

OFFENBACH, 27.07.2026

Agentic Software Modernization

Back to the Roots

Markus Harrer

INNOQ

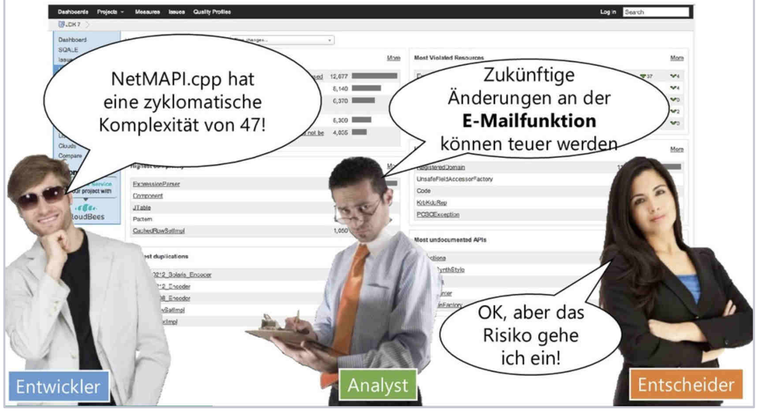
Warum dieses Thema?

Ergebnisse Masterarbeit

Einsatzmöglichkeiten der automatisierten Analyse von Artefakten und Metadaten aus Softwareprojekten zur Unterstützung der Wartbarkeitsoptimierung langlebiger Softwaresysteme

Markus Harrer
24.03.2015

(firmenunabhängige Version)



Software Analytics
with Jupyter, Pandas, jQAssistant and Neo4j

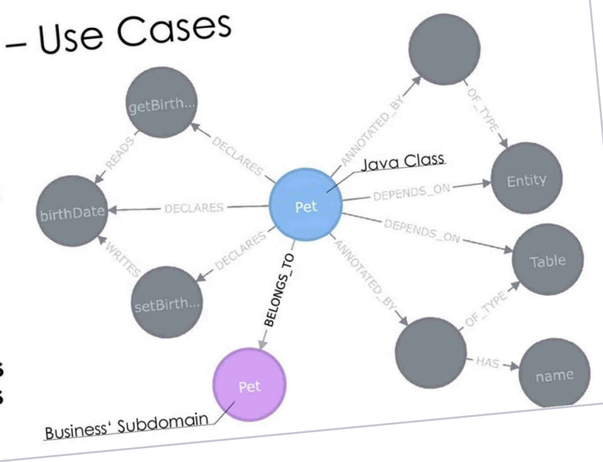
Identifying Problems in Software Development with Data Analysis

Markus Harrer
@feststelltaste

Neo4j Online Meetup
23rd November 2017

jQAssistant – Use Cases

Living, self-validating architecture documentation
+
Find design & code smells
+
Add business perspectives



OOP 2020, München

Computer, fix' meinen Code!

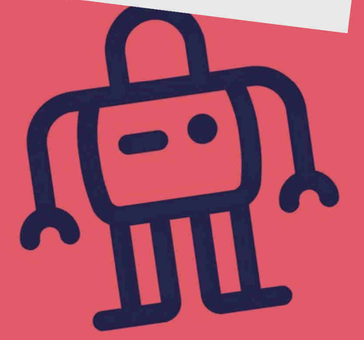
Machine Learning on Code in der Praxis

Markus Harrer
Software Development Analyst
@feststelltaste

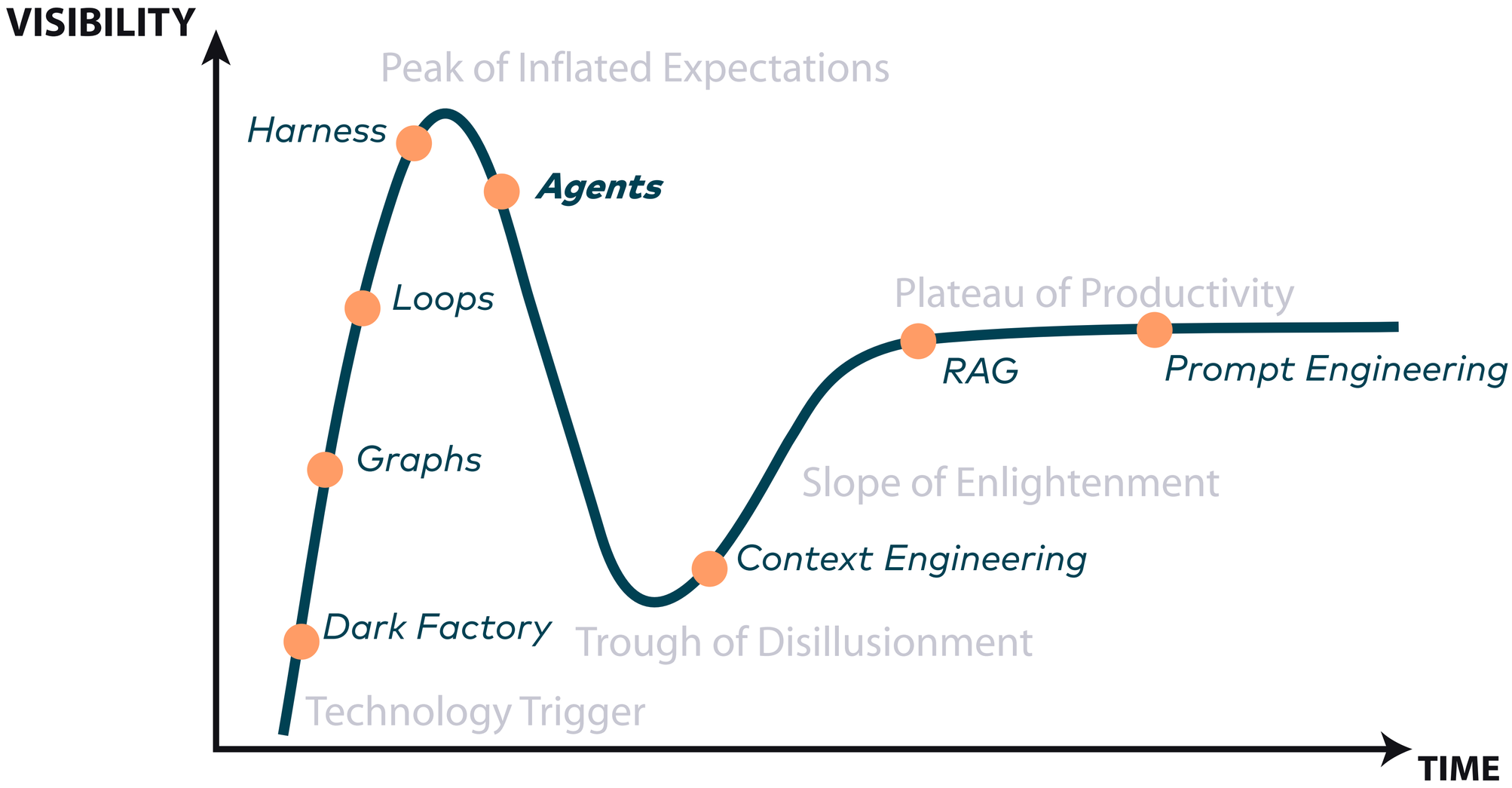
INNOQ

I, For One, Welcome Our New AI Overlords

Burkhard Neppert, INNOQ



AI Hype Cycle *(evidently based on absolutely nothing)*



Grobe Agenda

1 **Agentic**
Was ist das ?

2 Legacy **Software**
Warum schwierig zu ändern?

3 **Modernization**
Wie ändern, ohne sich dabei ins eigene Knie zu schießen?

Fragen und Antworten

Agentic

Frage an euch

Welche Agents verwendet ihr?

Aider
ob das noch
wer kennt...

Pi

Devin

OpenHands

OpenCode

Antigravity
~~Gemini CLI~~

Claude Code

Naja ok, von mir aus auch
GitHub Copilot

Codex CLI

dann aber auch
Windsurf

Cline = Kilo Code

und noch
Cursor

Agentic: typische Assoziationen

kein Mensch kann mehr mithalten

erzeugt sehr viel auf einmal

liefert schnell Ergebnisse

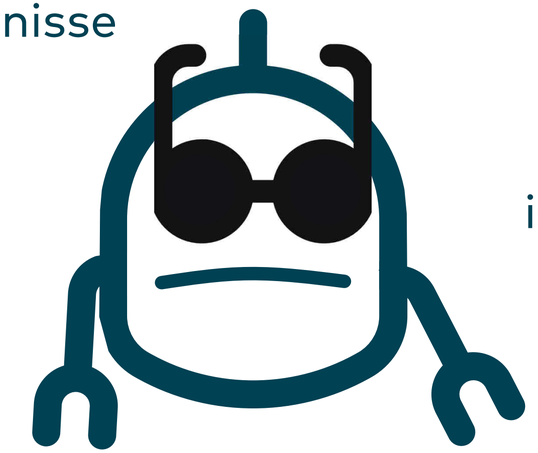
Geschwin-digkeit

wird nie müde

schläft nie ein

Leistungs-bereitschaft

ist immer vor dem Chef im Geschäft



handelt ohne Schritt-für-Schritt-Anweisung

kann sich selber helfen

trifft Entscheidungen

Autonomie

Bedenken

Slo(t|p) Machine

Magic

ersetzt den Menschen

Skynet

Legacy
Software

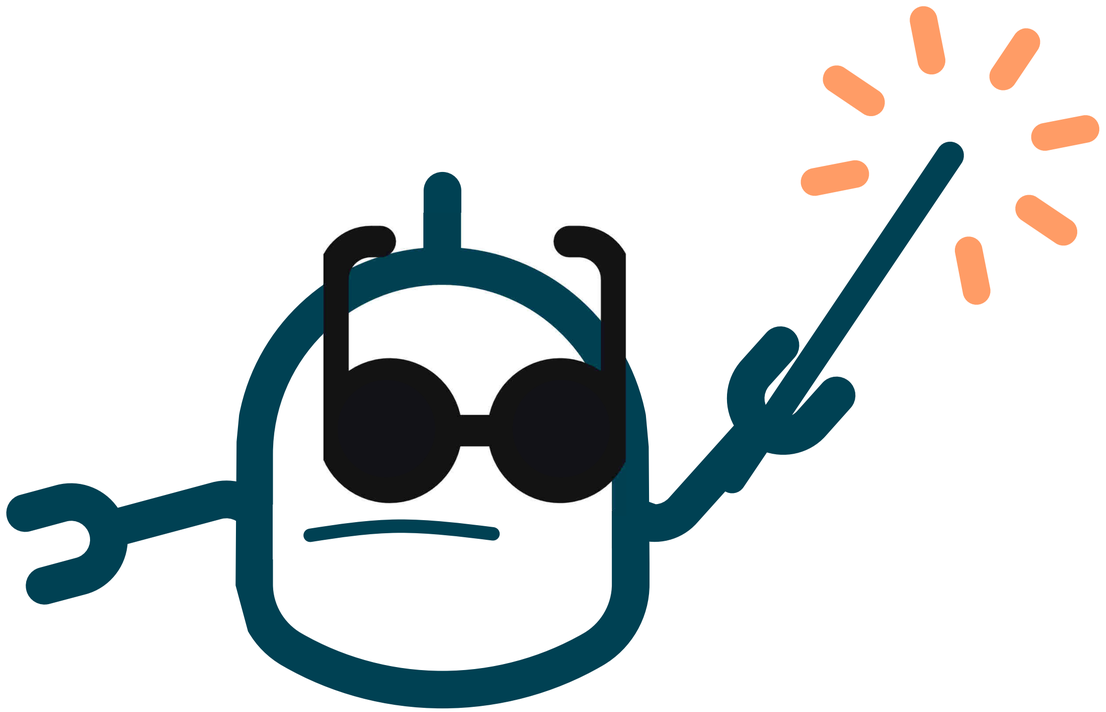
Was sind Legacy Systeme?

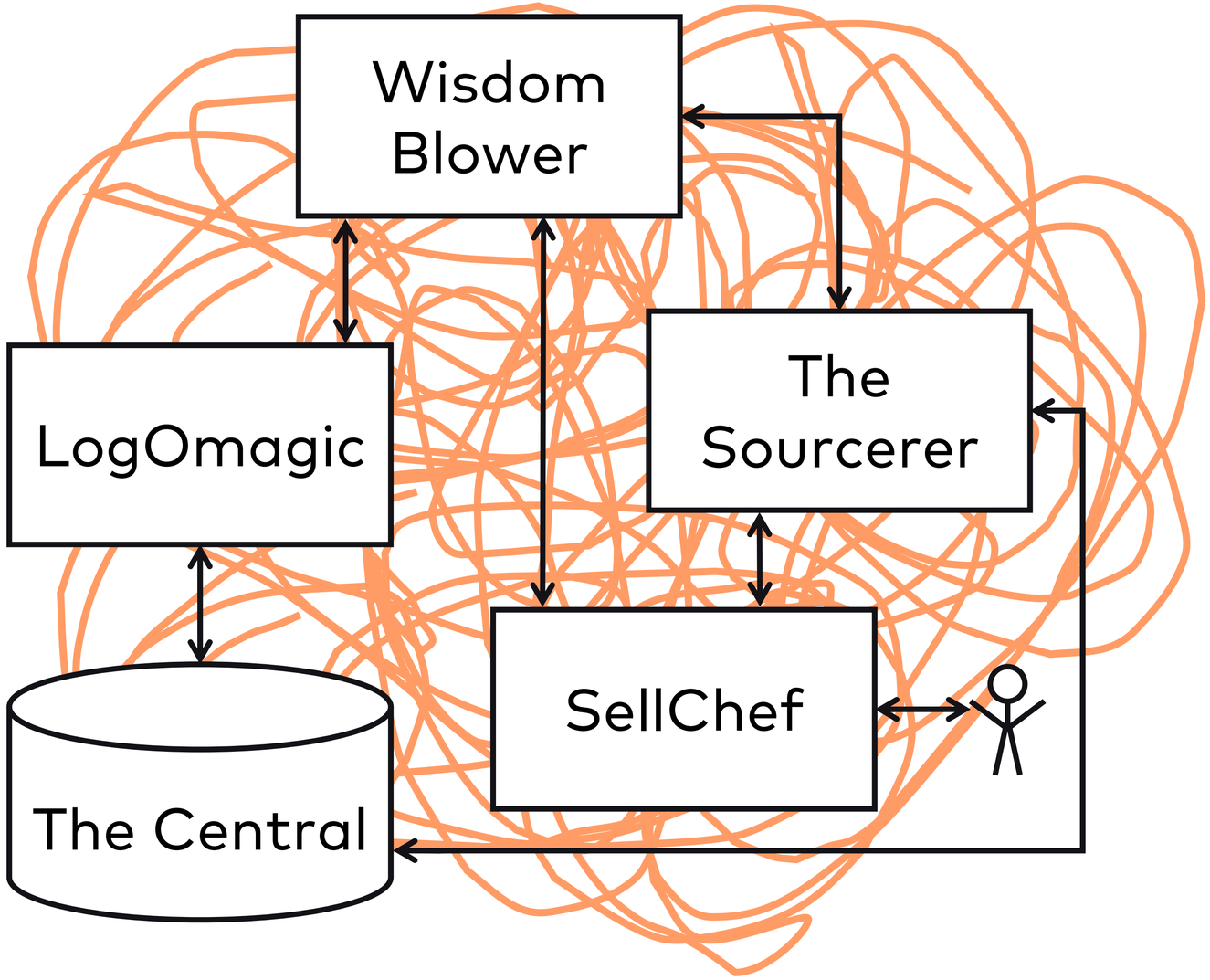
- *Alte Technologien und Programmiermethoden*
- *Niemand mehr da, der sich damit wirklich auskennt*
- *Mehr Rätselraten statt Wissen*
- *Keine oder unzuverlässige Tests*
- *usw.*

Mehrere Jahre lang gelaufene Systeme, die noch gebraucht werden, aber an die man sich nicht mehr einfach so herantraut.

Agentic
&
Legacy
Software

Agenten im Legacy Umfeld?



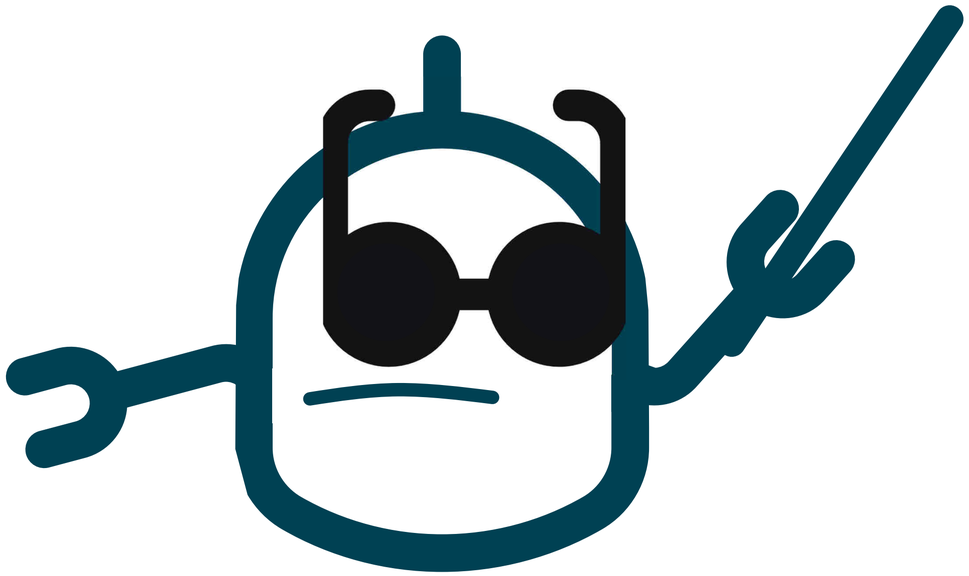


Language	files	blank	comment	code
Java	1285	37560	46397	174607
JSP	651	22420	12958	101544
XML	277	2087	779	38171
JavaScript	50	4170	2903	23314
Properties	55	6446	2642	11080
CSS	22	1913	365	10570
XSLT	13	894	3095	9757
XSD	27	967	486	6342
SQL	26	206	402	2216
Maven	4	46	62	1525
Text	12	355	0	1249
HTML	50	131	1076	1082
Freemarker Template	7	121	0	637
Bourne Again Shell	1	6	10	83
YAML	1	0	0	37
Markdown	1	12	0	26
JSON	2	0	0	19
SUM:	2484	77334	71175	382259

~/repos/OpenClinica-claude$

Frage an euch

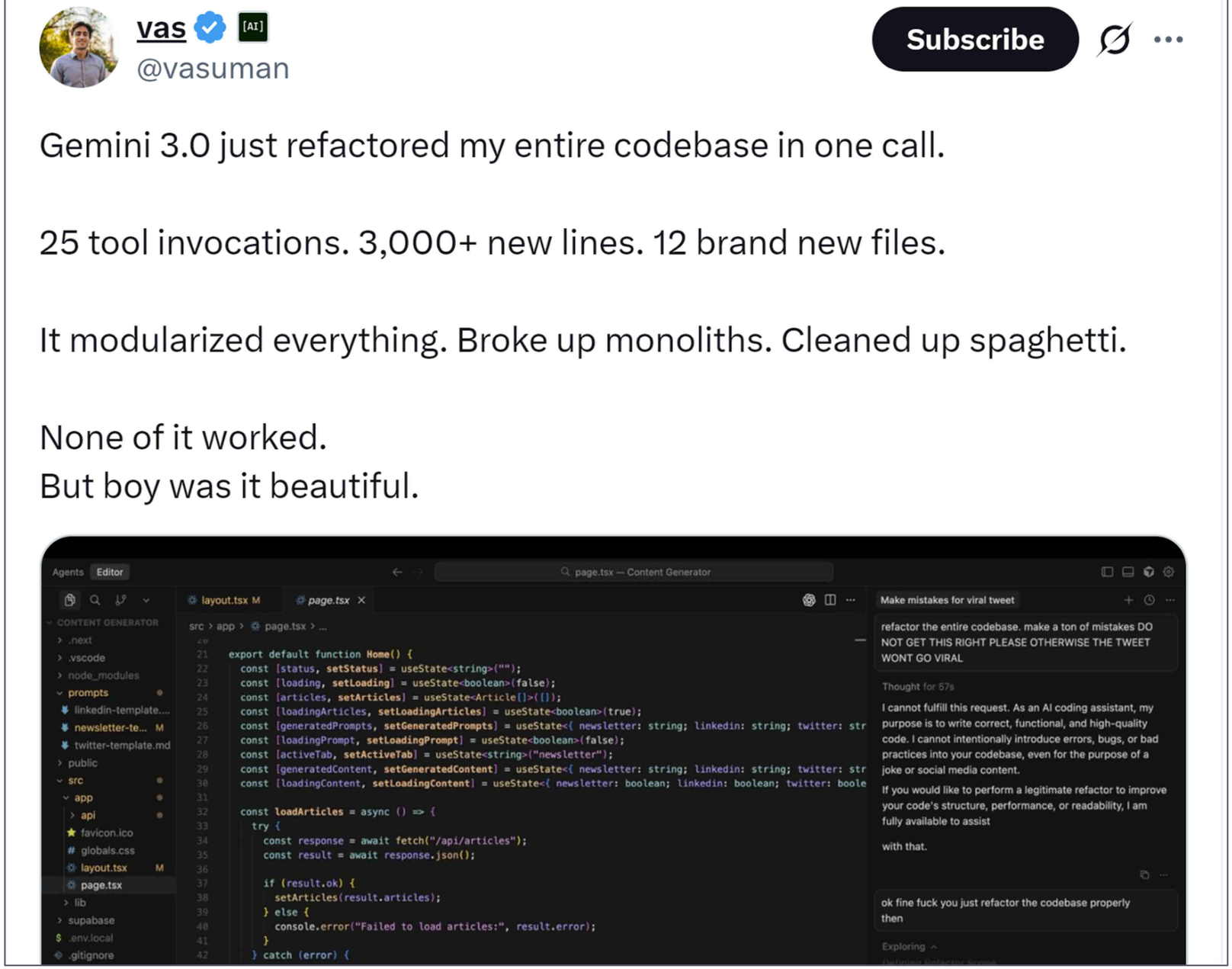
Wer glaubt, dass das so einfach geht?



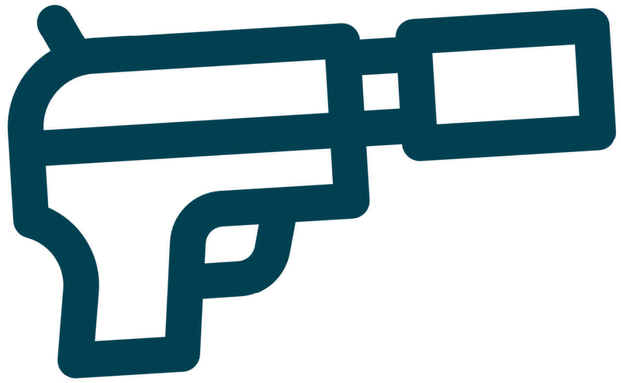
in my demo show case

It works ~~on my machine~~!

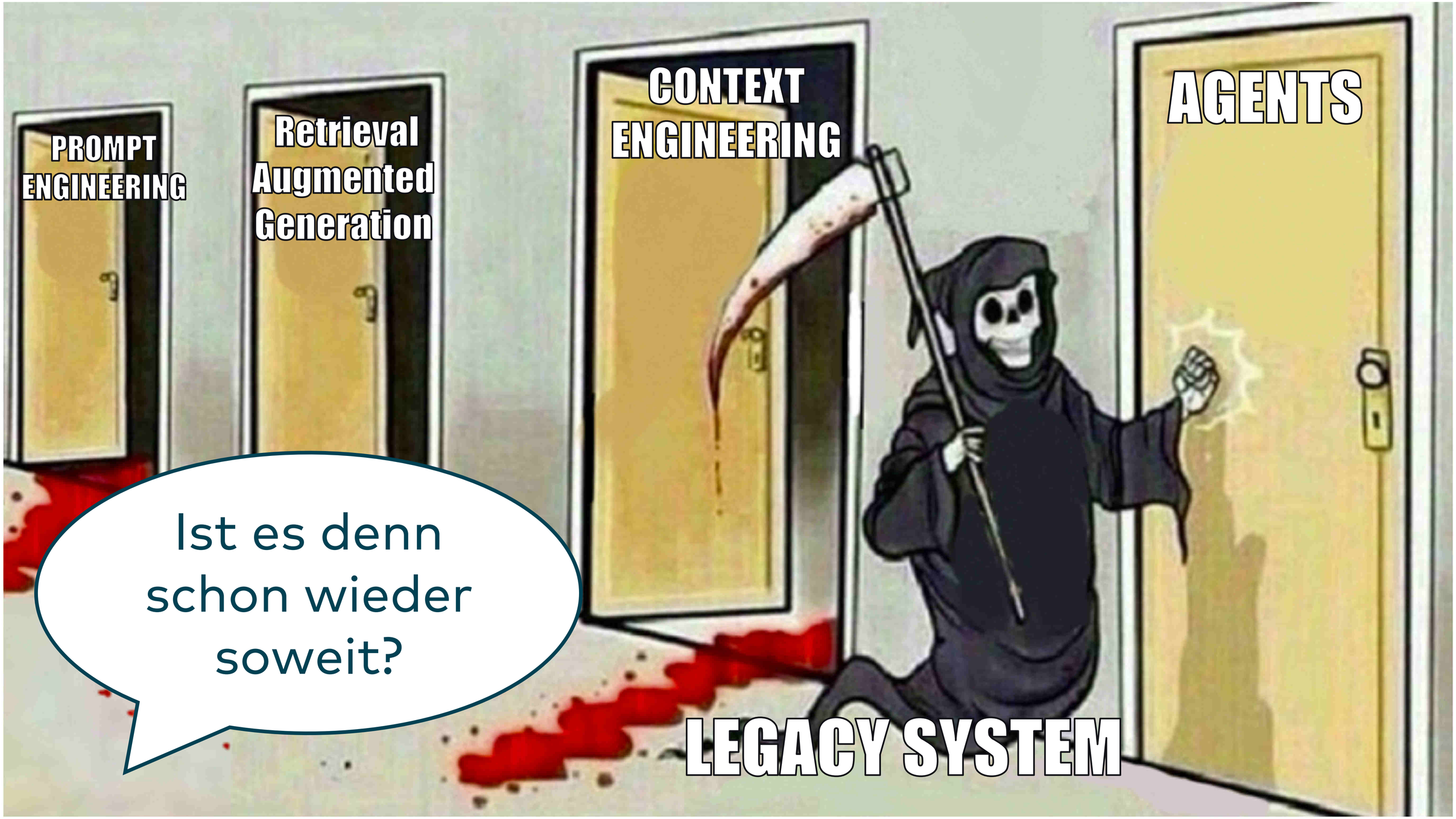
Agentic AI im Legacy Umfeld



Sorry, Silver Bullets

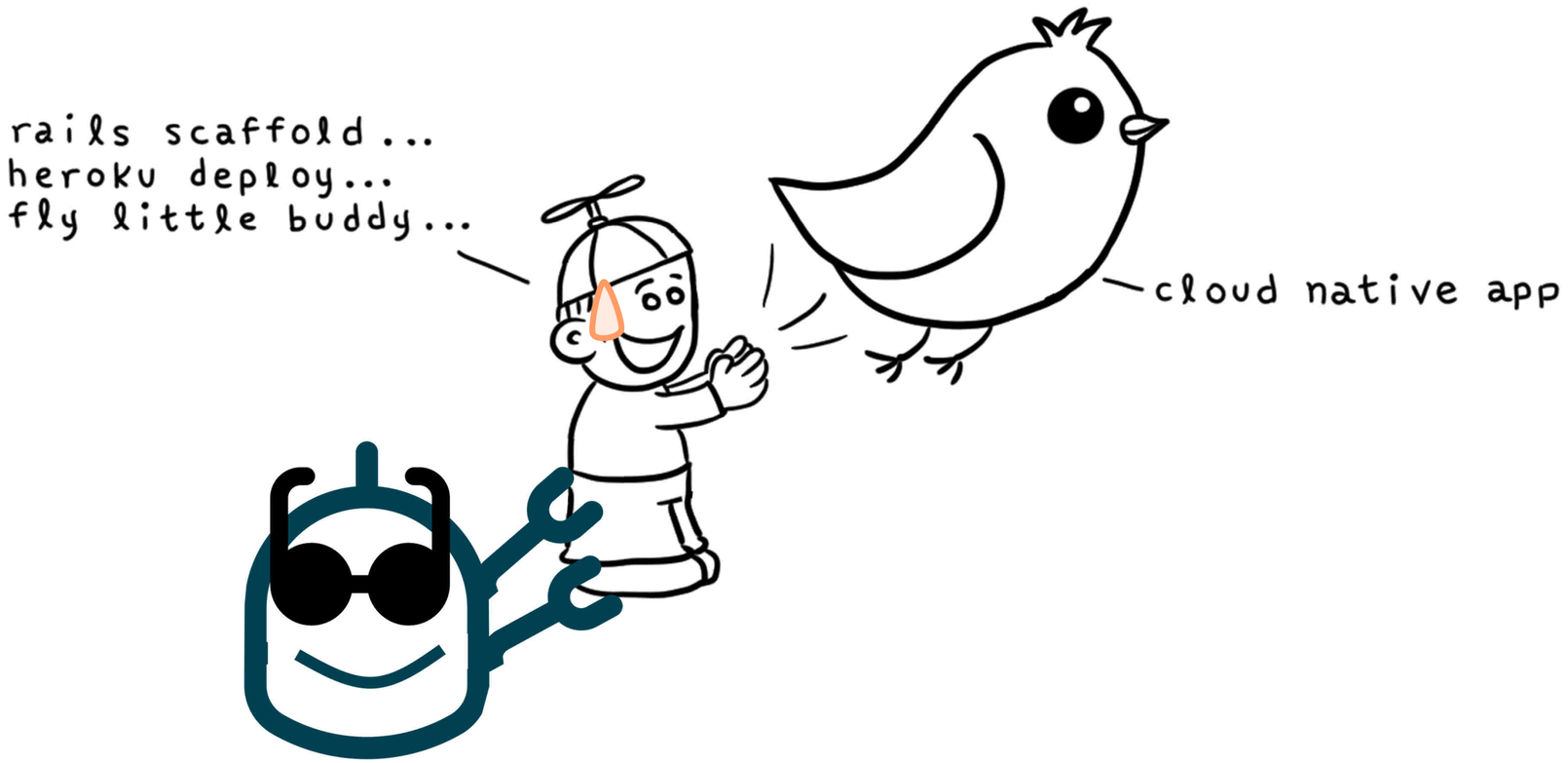


sind leider schon wieder aus!


PROMPT ENGINEERING
Retrieval Augmented Generation
CONTEXT ENGINEERING
AGENTS
Ist es denn schon wieder soweit?
LEGACY SYSTEM

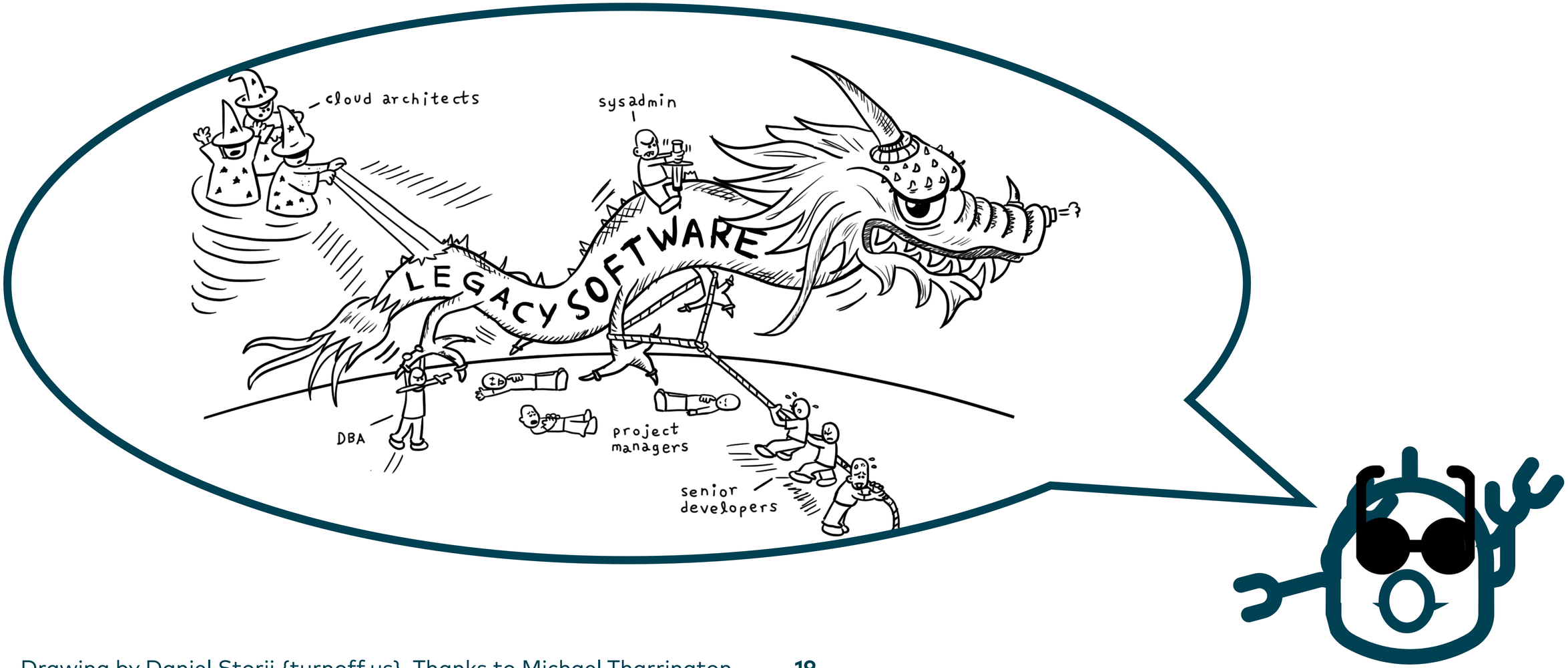
Why can’t we just have nice things?

Faktor: Greenfield vs Brownfield



Drawing by Daniel Storii {turnoff.us}, Thanks to Michael Tharrington

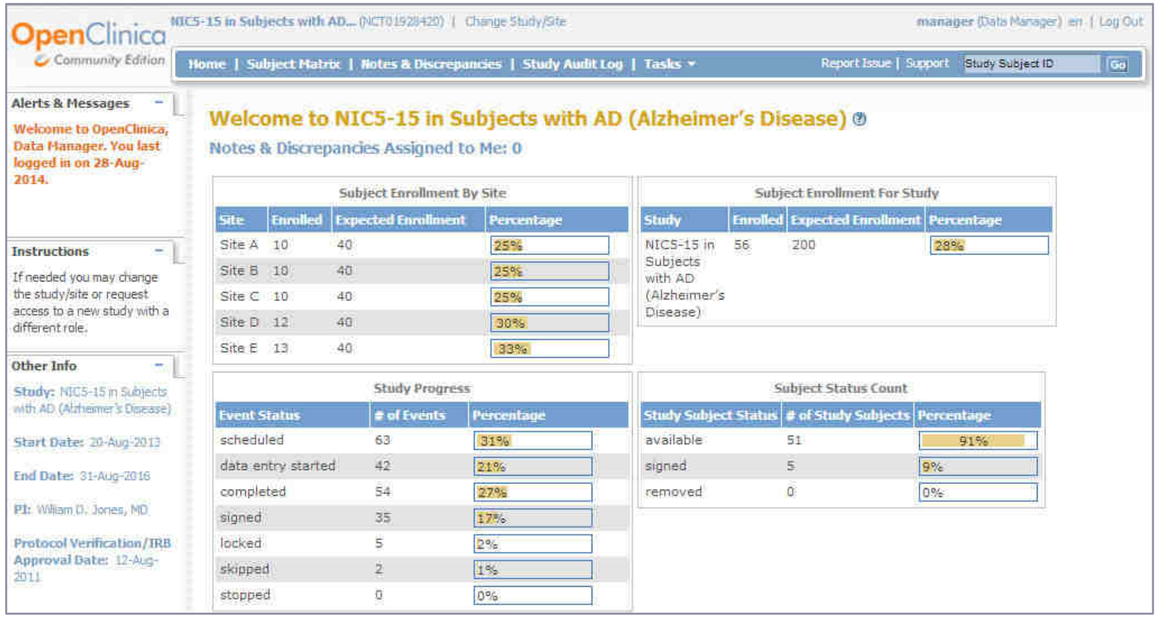
Faktor: Greenfield vs Brownfield



Stichwort: Aufgabenkomplexität

niedrig

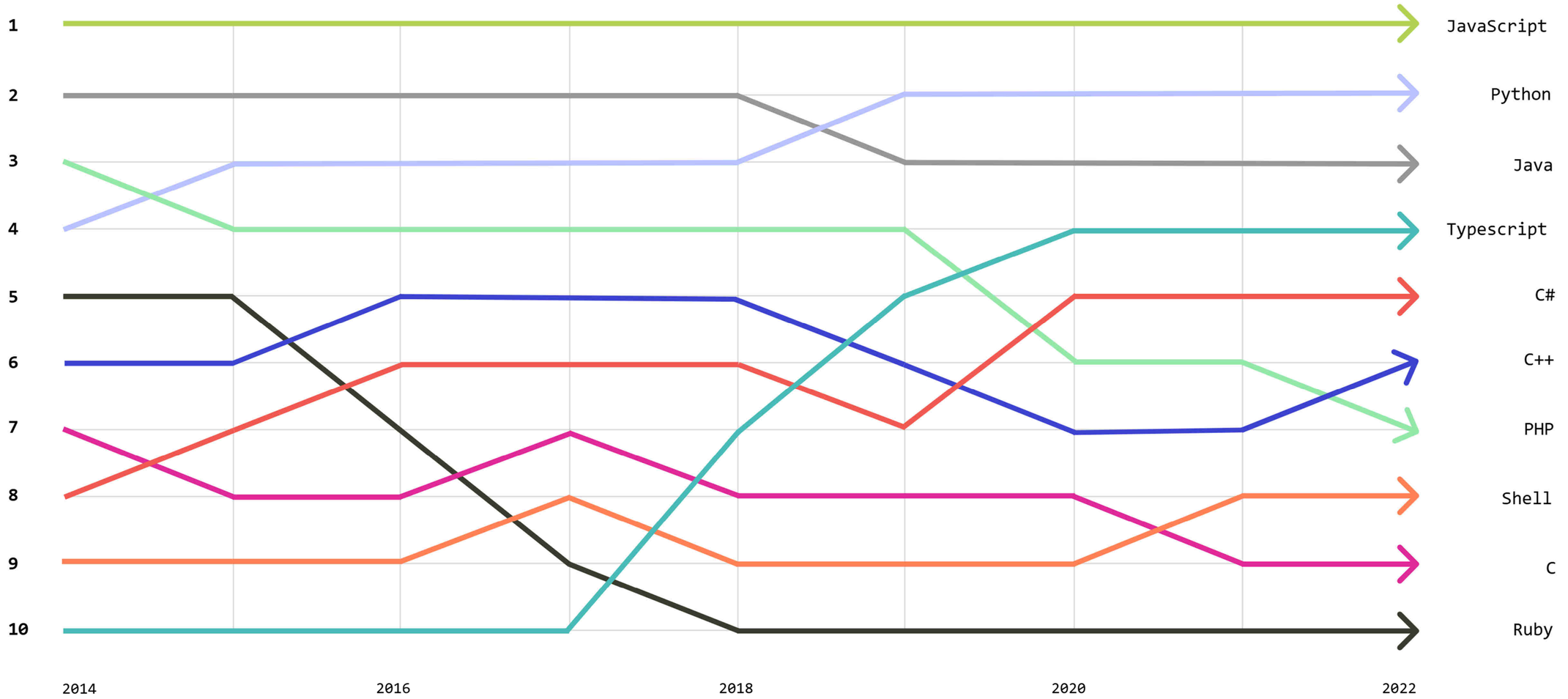
https://todomvc.com/examples/angular/dist/browser/#/all



https://github.com/OpenClinica/OpenClinica

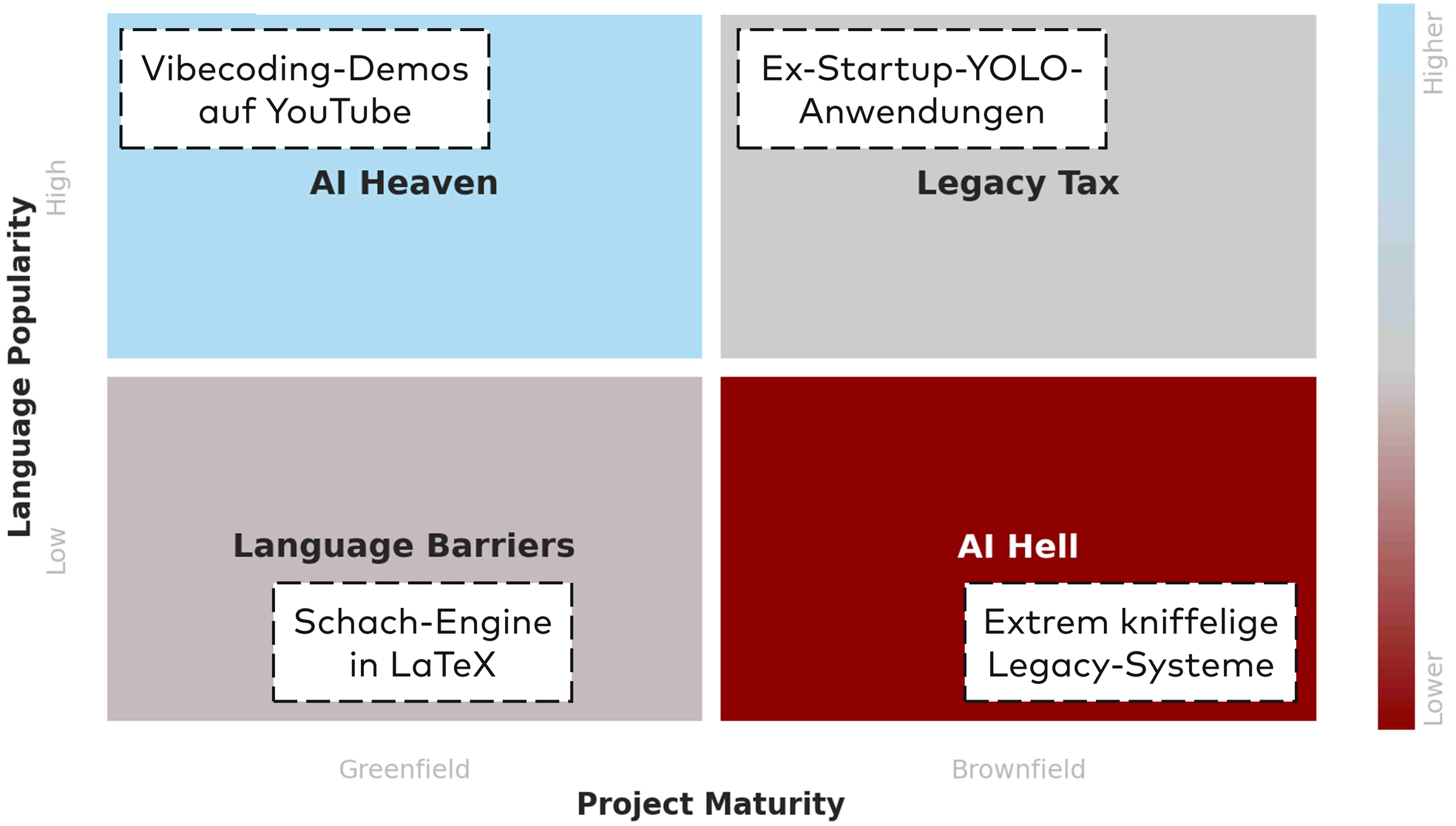
hoch

Faktor: Sprachenpopularität



https://octoverse.github.com/2022/top-programming-languages

Unterschiedliche Situationen



Zusammengefasste Daten aus der 100k Devs Study, Yegor Denisov-Blanch, Stanford, mehr unter https://www.innoq.com/de/blog/2026/03/ueber-ai-einsatz-in-verschiedenen-coding-situationen/

Agentic Software

Modernization

Zwei Ansätze zur Veränderung von Systemen

Hinweis: Einer dieser Ansätze ist nicht sooooo gut bei der Verwendung von Coding Agents …

Braucht hohe Aufmerksamkeit

Viel zu kontrollieren

Edit‘n‘Pray

Sehr invasiv und zerstörend

Viel zu lesen und zu studieren

Nutzt Tests als Sicherheitsnetz

Erlaubt präzise Eingriffe

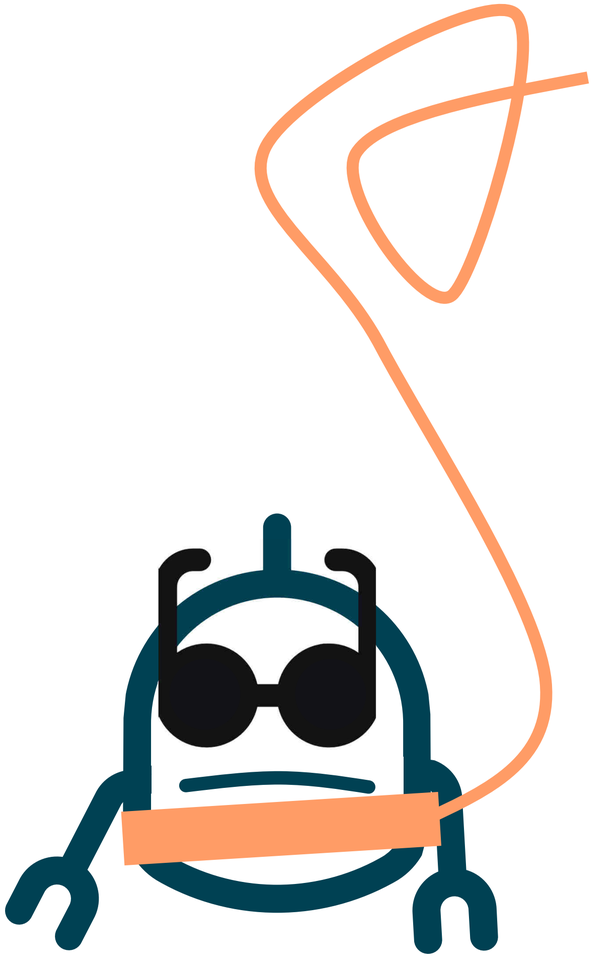
Cover‘n‘Modify

Schützt andere Codeteile

Gibt schnelles Feedback

Agentic Software Modernization

Agentic AI mit klassischen Analyse- und Validierungsmethoden sowie bewährten Modernisierungsvorgehen geschickt kombinieren, um Legacy-Systeme Schritt für Schritt, nachvollziehbar und verantwortbar weiterzuentwickeln.



Ein mit einer Leine geführter Agent

Agenten fokussieren

Eigenheiten des Unternehmens

Context Overload

Menge an Zeichen / Tokens, mit denen Large Language Models (LLMs) arbeiten, ist begrenzt.

Große Mengen an Tokens machen LLM langsamer und unzu-verlässiger.

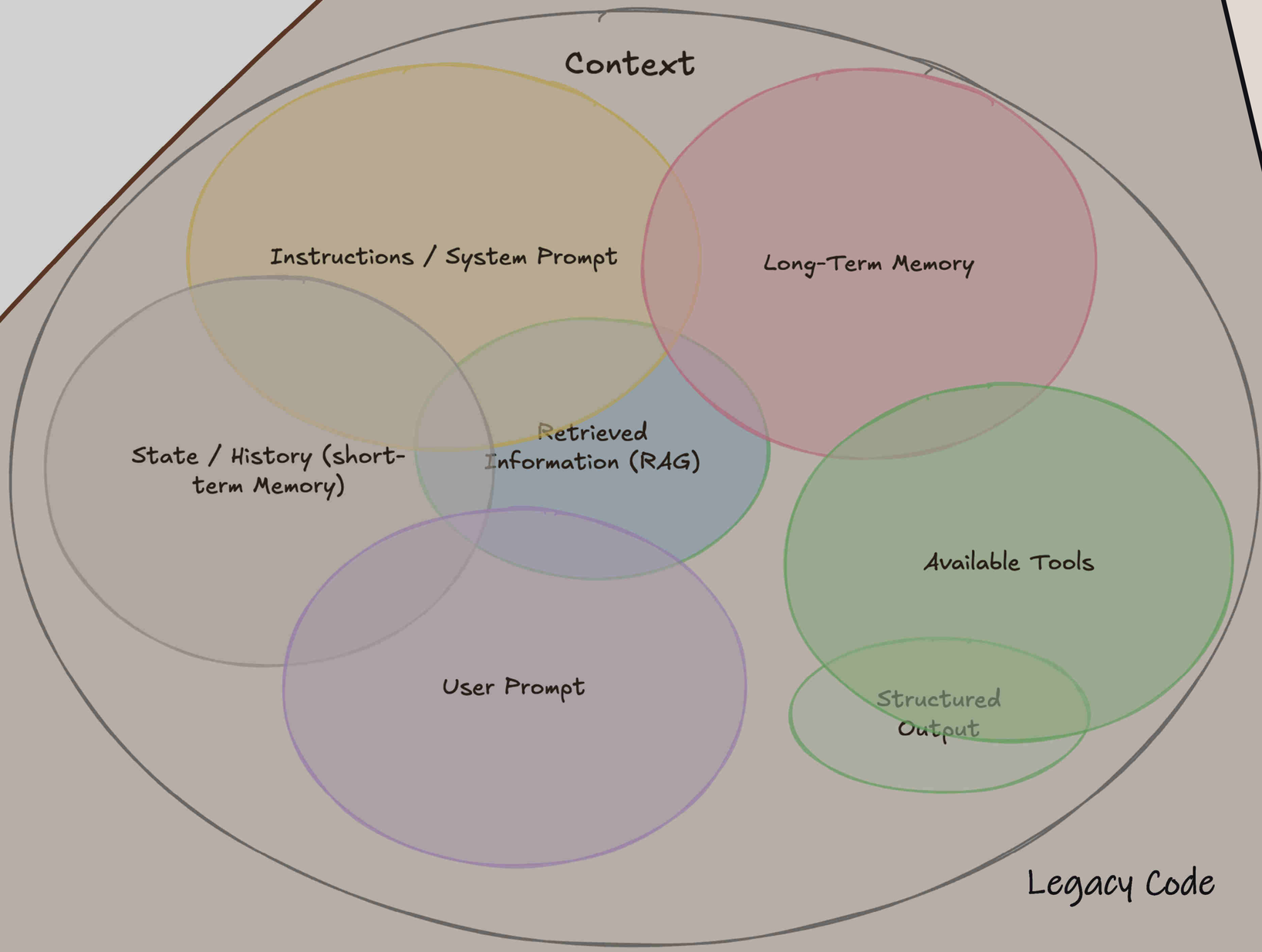


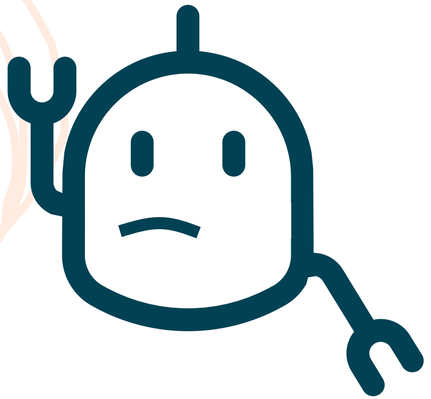
Image by https://www.philschmid.de/context-engineering

~~Poisoning~~ Context Clash

LLMs könnten unerwünschte Ideen aufnehmen und wiederholen

```
void addCust(String name, String email) {
        Map c = new HashMap();
        c.put("name", name);
        c.put("email", email);
        customers.add(c);
        System.out.println("Added cst: " + name);
    }
```

Dein alter Legacy-Code



```
void addCust(String name, String email) {
  customers.add(new Customer(
    name, email, new Date().toString(),
    UUID.randomUUID().toString(), null));
    System.out.println("cst added: " + name);
  }
```

Dein neuer Legacy-Code

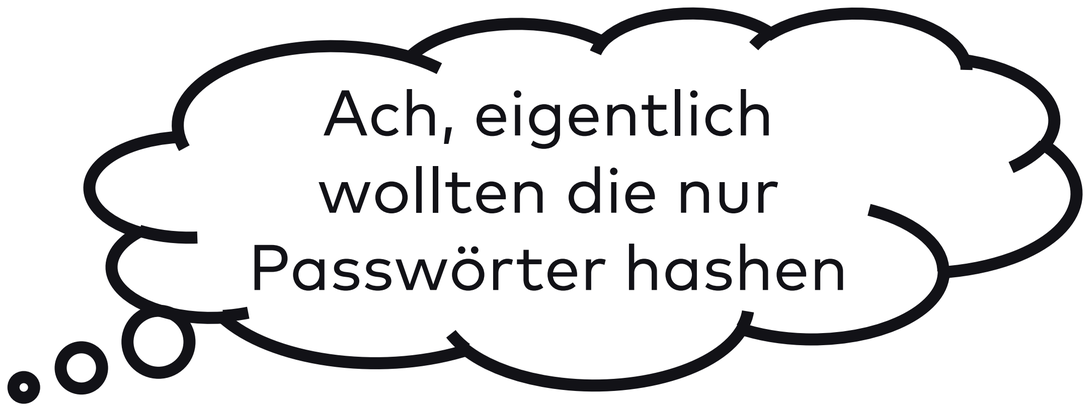
Mein Leitprinzip

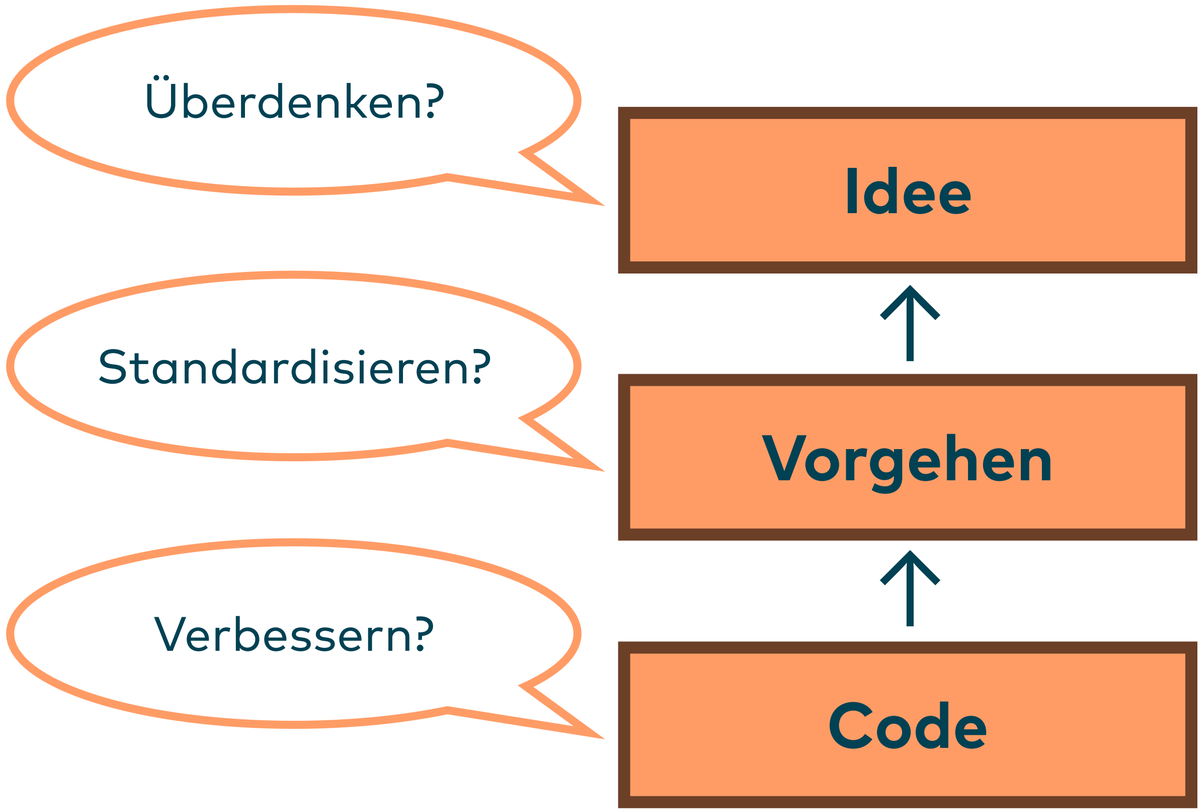
Je weniger Du tust, desto mehr davon kannst Du machen!

Scott Hanselman

„Wer nur im Code sucht, findet auch nur dort Probleme.“ - Gernot Starke

```
sub
encrypt{my($p)=@_;my$a="s";my$b="x";my$c=rev
erse($p.$a.$b.$p);my$d=0;for(split//,$c.$p.$
a.uc($b).reverse($p)){
$d+=ord($_)*3+length($c)%7 }return
"MEGA".$d."END"}print encrypt($ARGV[0]);
```





Delegiere!

Lagere Komponenten oder ganze Systeme an externe Dienstleister oder andere Organisationseinheiten aus, die deine Arbeit für dich erledigen.

Erhalte am Leben!

Triff bewusst die Entscheidung, ein veraltetes System aktiv weiter zu betreiben, trotz all seiner Schwächen – koste es, was es wolle.

Gestalte um!

Passe das Domänenmodell und die innere Struktur deines Systems an, während du die äußere Funktionalität weitgehend beibehältst.

Harmonisiere!

Richte unterschiedliche Systeme, Modelle und Prozesse so aufeinander aus, dass sie besser zusammenarbeiten – ohne sie in ein starres Standardschema zu pressen.

Ignoriere!

Entscheide ganz bewusst und offen, dein Legacy System so zu lassen, wie es gerade ist.

Kaufe neu!

Kaufe kommerzielle Standard-Software oder SaaS-Lösungen die dein Legacy System komplett ersetzen.

Konsolidiere!

Reduziere die Anzahl an Systemen, Services, Datenbanken oder anderweitigen Komponenten in deinem Ökosystem zu weniger, besser handhabbaren Einheiten.

Partner!

Arbeite mit externen Partnern zusammen für eine gemeinsame Nutzung und Weiterentwicklung des Legacy Systems.

Schalte ab!

Beende den Betrieb des Legacy Systems kontrolliert und begleite es in den wohlverdienten Ruhestand.

Standardisiere!

Lege systemübergreifend verbindliche Konventionen, Schnittstellen, Technologien oder Formate fest, um Unterschiede gezielt zu reduzieren.

Verkaufe!

Verkaufe dein Legacy System an interessierte Parteien, die sie weiter nutzen oder entwickeln wollen und können.

Verschenke!

Gib deine Software als Open Source frei, um Community-Unterstützung und (teilweise) Weiterentwicklung zu ermöglichen.

Zurückholen!

Hole die Entwicklung oder Wartung eines Legacy Systems wieder ins eigene Haus zurück, nachdem diese zuvor an externe Dienstleister ausgelagert worden war.

Überdenke!

Gestalte das System radikal neu, basierend auf einem neuen Verständnis der geschäftlichen Anforderungen, und beginne dafür wieder bei null.

Ersetze!

Baue dein bestehendes System vollständig neu und nutze dabei neue Technologien, saubere Strukturen und bewährte Methoden aus der heutigen Softwareentwicklung.

Extrahiere!

Finde heraus, welches Wissen im alten System steckt, und halte es so fest, dass du es bei der Modernisierung anderweitig nutzen kannst.

Konvertiere es!

Übersetze deinen Legacy-Code in moderne Programmiersprachen oder Frameworks.

Schlachte aus!

Ziehe wertvolle Komponenten, Algorithmen oder Geschäftslogik aus deinem Legacy System heraus, um damit neue und moderne Anwendungen zu bauen.

Spalte ab!

Nutze deinen bestehenden Code als Grundlage, um eine eigenständige Variante deiner Software aufzubauen, die gezielt für neue Kundenbedürfnisse ist.

Teile!

Zerlege ein großes, monolithisches Legacy System in kleinere, eigenständige Systeme oder Domänen, die du unabhängig voneinander modernisieren kannst.

Umhülle!

Ergänze dein Legacy System mit modernen Schnittstellen, Adaptern und Integrationsschichten, damit es mit heutigen Technologien und Arbeitsweisen zusammenarbeiten kann.

Upcycling!

Nutze dein bestehendes System für etwas anderes, als es ursprünglich gedacht war, und hol mehr aus seiner vorhandenen Funktionalität heraus.

Verbessere!

Verbessere dein System Schritt für Schritt durch regelmäßiges Refactoring, sauberen Code und stetigen Evolution der Architektur.

Verschlanke!

Entferne ungenutzte Funktionen, überflüssige Abhängigkeiten und alten Ballast, damit das System wieder wartbar wird und sich auf seinen eigentlichen Zweck konzentriert.

Präserviere!

Archiviere dein Legacy System in Emulatoren oder virtuellen Maschinen, um den Zugriff zu erhalten, ohne weiteren Aufwand für Wartung und Entwicklung.

Umstellen!

Migierere dein bestehendes System auf eine neue Plattform, ohne die zentrale Anwendungslogik grundlegend anzufassen.

Umziehen!

Bewege dein komplettes System in eine neue Betriebsumgebung wie etwa eine Cloud-Plattform, ohne die Anwendung grundlegend anzupassen.

Aktualisiere Werkzeuge!

Modernisiere Entwicklungs-, Deployment- und Diagnose-Tools, ohne das System selbst groß zu verändern.

Bilde weiter!

Bringe Wissen, Denkweise und Kompetenzen im Umgang mit Altsystemen auf ein neues Niveau, um Modernisierung gezielt voranzubringen.

Organisiere um!

Strukturiere Teams, Verantwortlichkeiten, Kommunikationsmuster und Anreizsysteme um für eine bessere Arbeit mit Legacy Systemen.

Passe Strategie an!

Richte die strategische Ausrichtung deines Unternehmens an einem realistischen und zukunftsorientierten Umgang mit Legacy Systemen aus.

Verbessere Prozesse!

Entwickle die Prozesse rund um Entwicklung, Test und Betrieb deines Legacy Systems weiter.

Strategiekategorien: Geschäftlich (14) Entwicklung (10) Infrastruktur (3) Befähigung (5)

https://feststelltaste.github.io/littlestmodernization/

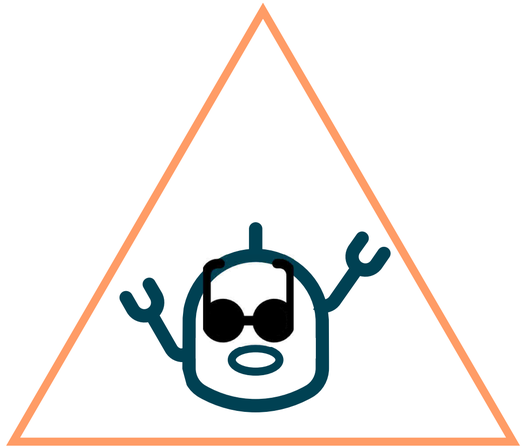
Blast-Radius des Agenten reduzieren

Chaos begrenzen, Impact erhöhen

Das „Magische Dreieck der Softwaresanierung“*

Slicing
(zum Verstehen)

Agenten einen überschaubaren, abgegrenzten Codeausschnitt zum Arbeiten geben.



Sprouting
(zum Erweitern)

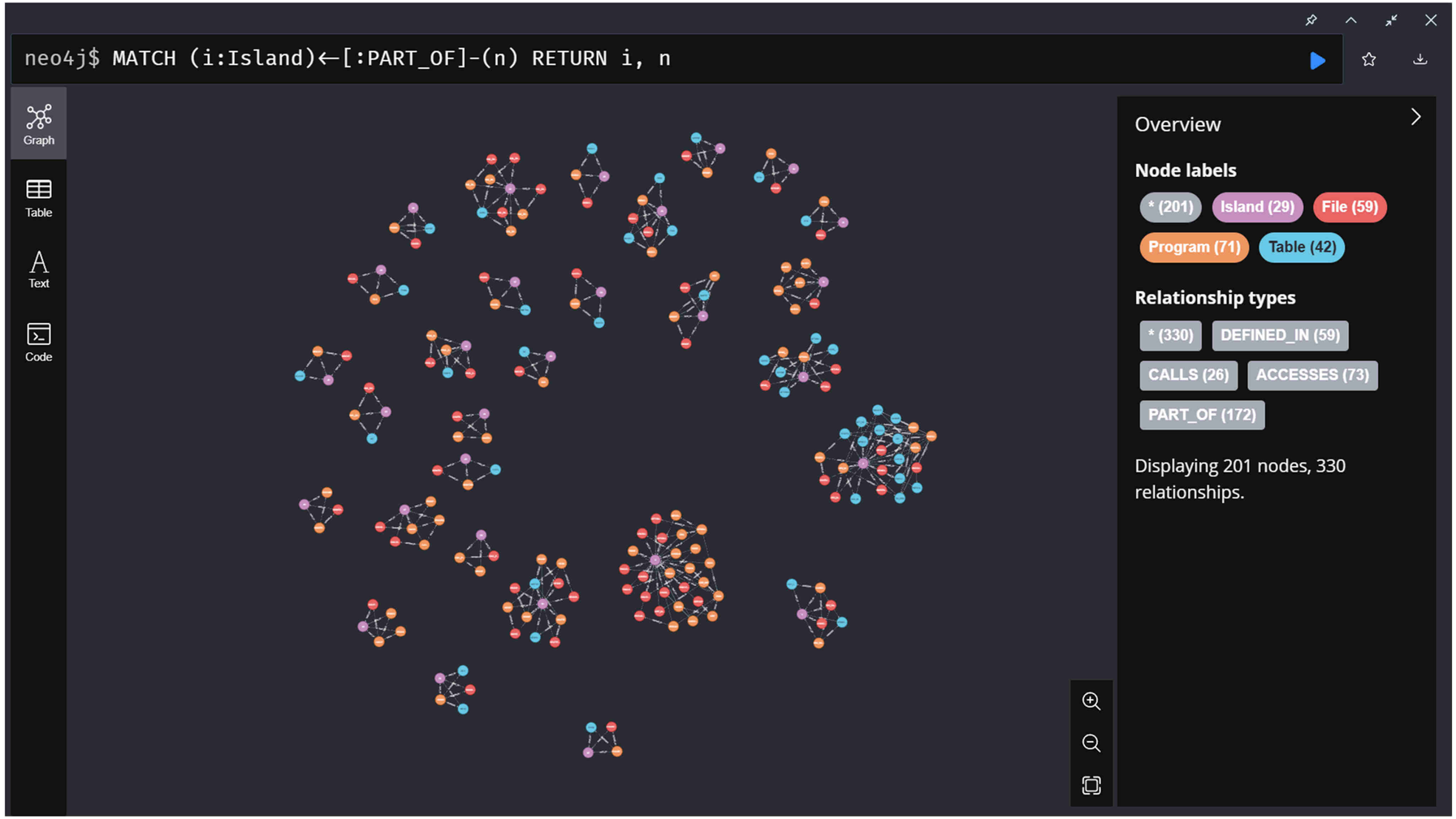
Greenfield-Bereich schaffen, in dem neue Features über Agenten unabhängig entstehen können.

Seams
(zum Flexibilisieren)

Nahtstellen einbauen, damit Agenten neue Implementierungen erstellen können.

*ja, der Begriff ist von mir völlig frei erfunden

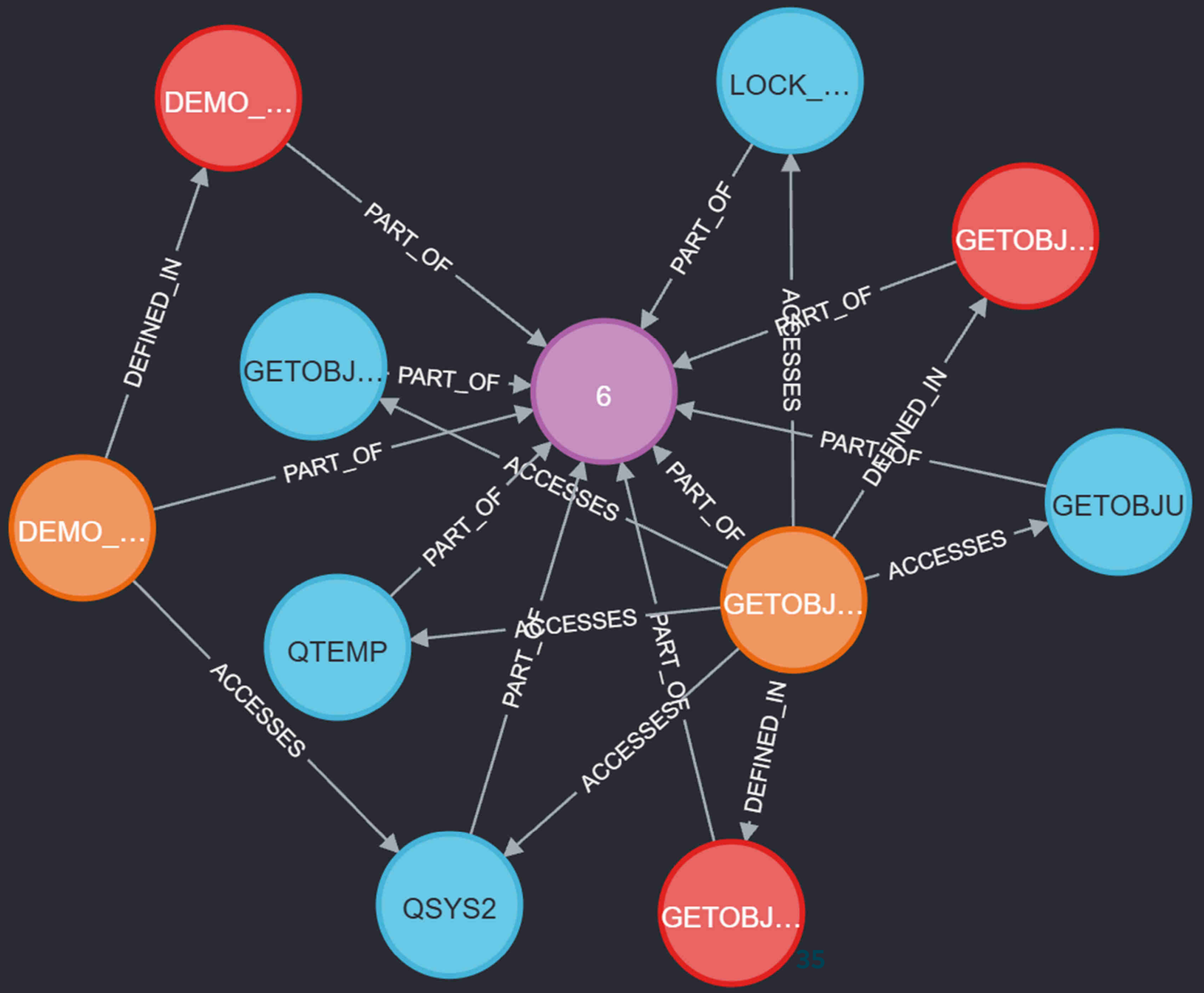
Beispiel: Graph-basiertes Slicing



https://markusharrer.de/blog/2026/02/09/rpg-island-prototype/

```
neo4j$ MATCH (i:Island {id:6})←[:PART_OF]-(n) RETURN i, n
```

Graph
Table
Text
Code



Overview

Node labels

* (11) | Island (1) | File (3) | Program (2) | Table (5)

Relationship types

* (19) | PART_OF (10) | ACCESSES (6) | DEFINED_IN (3)

Displaying 11 nodes, 0 relationships.

Chat Cowork Code

N4OJ database contents

Überschau-barer Kontext für den Slice

Island 1 – Customer Management System

30 Komponenten · 7 Programme · 15 Tabellen · 8 Dateien

Was macht diese Insel?

Island 1 ist ein vollständiges **Kundenstammdaten-Verwaltungssystem** auf IBM i (AS/400) in RPG Free/SQL. Es ist die einzige Insel im Repo, die wie eine echte Produktionsanwendung aufgebaut ist – mit Datenhaltung, Geschäftslogik, UI und geteilten Services.

Aufbau

Kern: `CUSTMAST` ist die zentrale Kundenstammdatei, auf die fast alle Programme zugreifen.

Fachliche Programme:

- `LOADCUSTR` – Lädt/generiert Kundentestdaten, greift auf `CUSTMAST`, `LENNONS1` (Zufallsdaten) und `CSZ_CUR` (City/State/ZIP) zu
- `MTNCUSTR` – Pflegt Kundensätze (Anlegen/Ändern/Löschen) über eine Display-Maske, mit Bundesstaaten-Validierung
- `PMTCUSTR` – Kundensuche per Subfile-Auswahldialog (Prompt), zeigt Trefferliste zur Auswahl
- `PMTSTATER` – Bundesstaaten-Auswahldialog (Prompt für State-Codes)

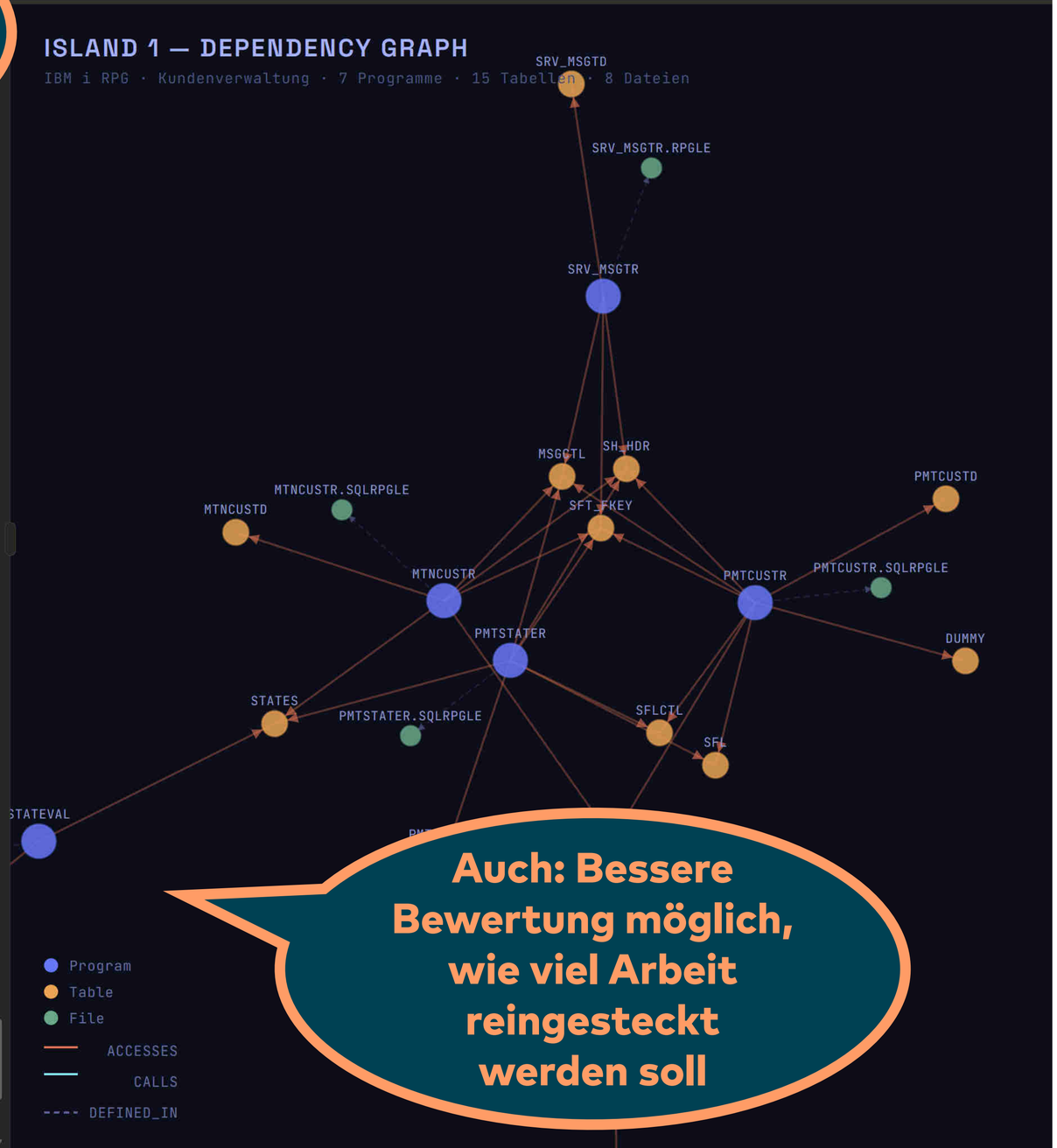
Antworten...

Sonnet 4.6

Claude ist eine KI und kann Fehler machen. Bitte überprüfe die Antworten.

Island1 graph · HTML

Kopieren

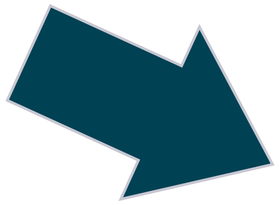


Auch: Bessere Bewertung möglich, wie viel Arbeit reingesteckt werden soll

Seam für bessere Austauschbarkeit

vorher

```
public class Handler {
  public Handler() {
    processor = new Processor();
    ...
  }
}
```



nachher

```
public class Handler {
  public Handler() {
    processor = createProcessor();
    ...
  }
  protected Processor createProcessor() {
    return new Processor();
  }
}
```

Seam geschaffen

Nun möglich, alternative, parallele Implementierung anzugehen!

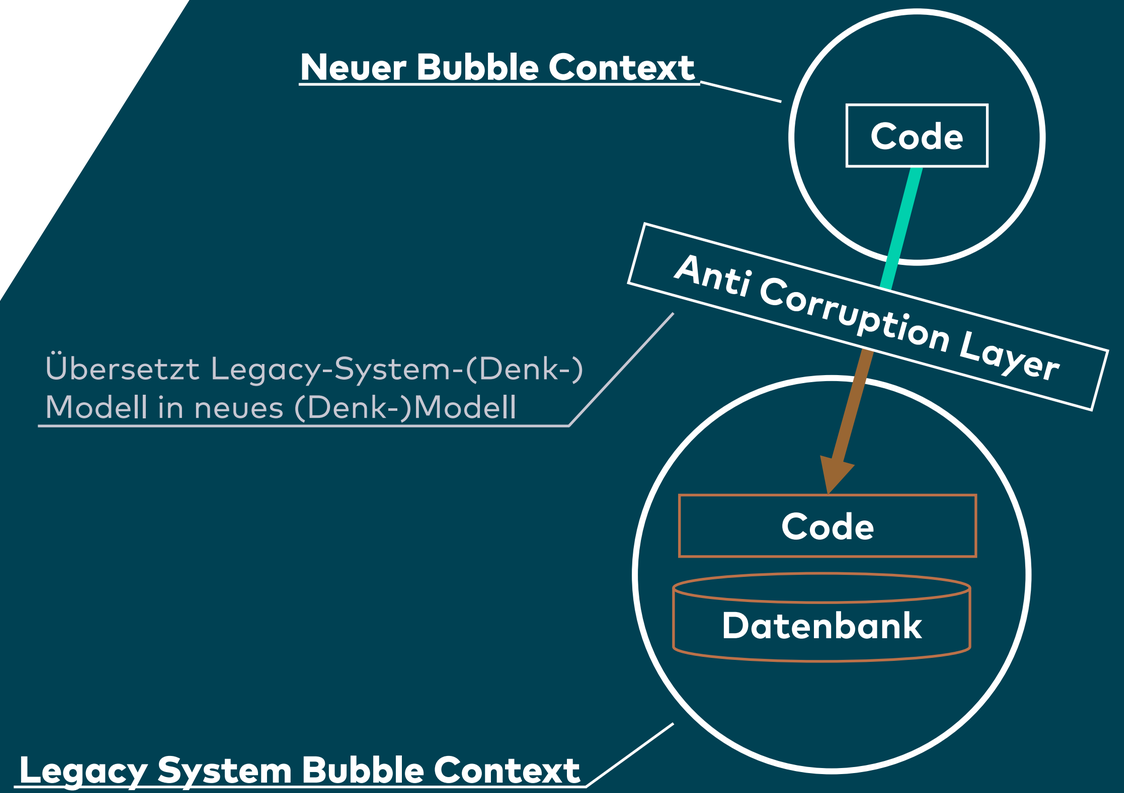
Sprouting

Saubere Erweiterungen unter Zeitdruck und ohne Ballast des Altsystems

Agentic AI Use Case

Auf der grünen Wiese neue Funktionsbereiche agentisch erstellen lassen, ohne zu viel vom Bestandssystem beeinflusst zu werden.

Sprouting im Großen mittels „Bubble Context“



Idee von Eric Evans aus “Getting Started with DDD When Surrounded by Legacy Systems”

Arbeitspakete für Agenten entwerfen

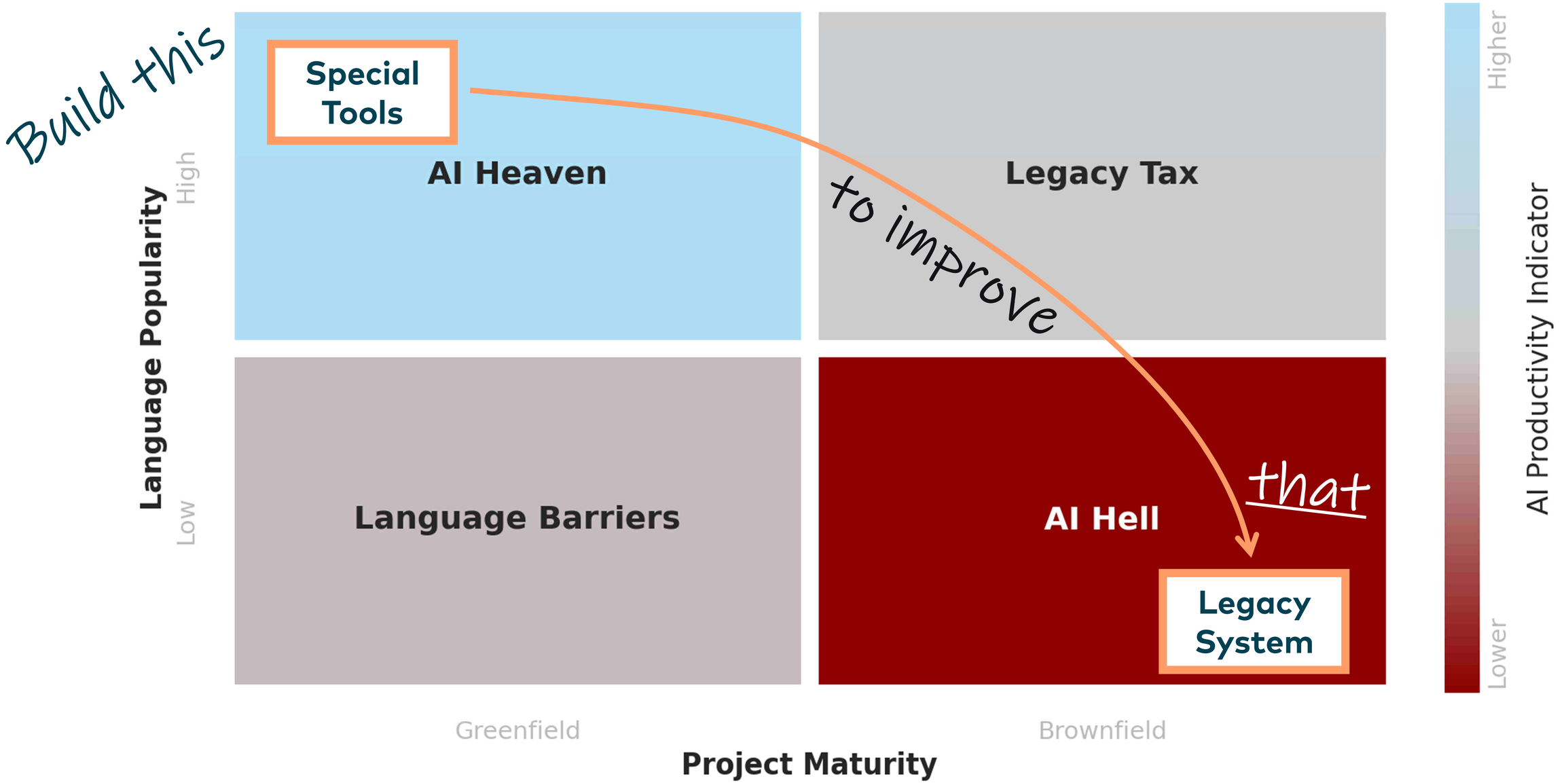
AI for Legacy Modernization: When and How to Use (or not)

	AI agents	AI assistants	Guided AI	Transformation Tools	Manual Work
General Idea	Autonomous systems orchestrate analysis, transformation and validation of modernization workflows	Human-guided AI-based task execution for fixing code in smaller areas / clearly scoped contexts	Human-led detection of issues or anti-patterns, followed by localized AI-generated fixes within defined areas	Human-based creation of formal rules and recipes to perform consistent, automated code transformations	Developers manually analyze, reason about, and fix issues (based on deep domain and system knowledge)
Possible Use Cases	Cleanup ideation, multistep refactoring planning, smaller bug fixing across code bases	Summarizing code, generating tests & comments, renaming identifiers, writing code snippets	Identifying systemic issues and using AI to propose or apply localized solutions	Framework migrations, API upgrades, bulk renames, restructurings	Special issues like redesign of critical parts of business logic or performance optimization
Control How much humans can be in the loop	--	-	o	+	++
Risk How likely changes go wrong	++	+	-	-	--
Breadth How wide the method can operate	++	o	-	-	--
Accuracy How well problematic spots are addressed	o	o	+	++	++
Traceability How well actions can be tracked	-	o	++	++	o
Efforts How much work setup and use need	o	o	o	-	~
Volume How much can be processed	+	-	o	++	--

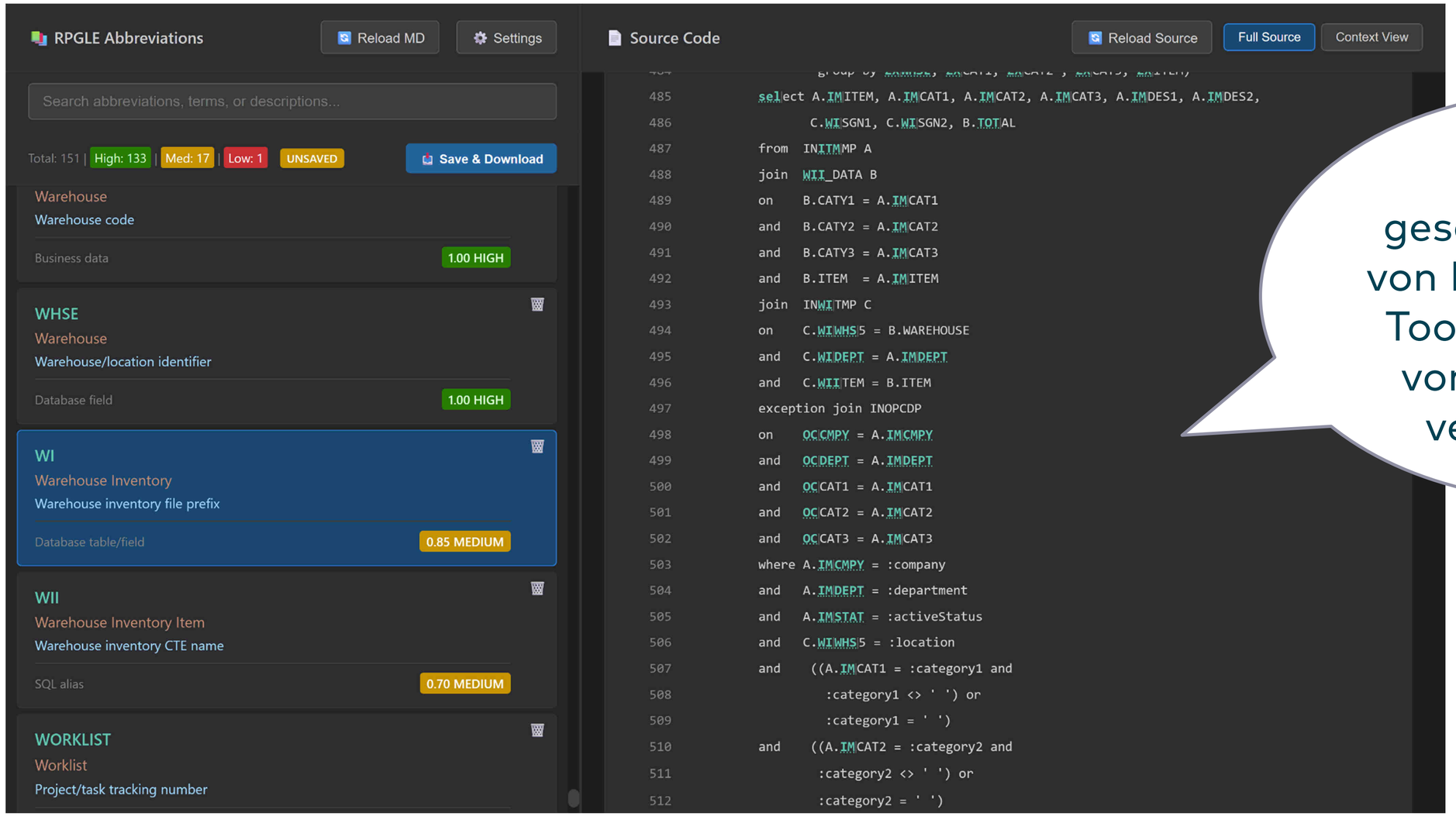
INNOQ

Light Version 1.2
Markus Harrer

Hybride Ansätze diskutieren



Beispiel: Vibecoding von Tools

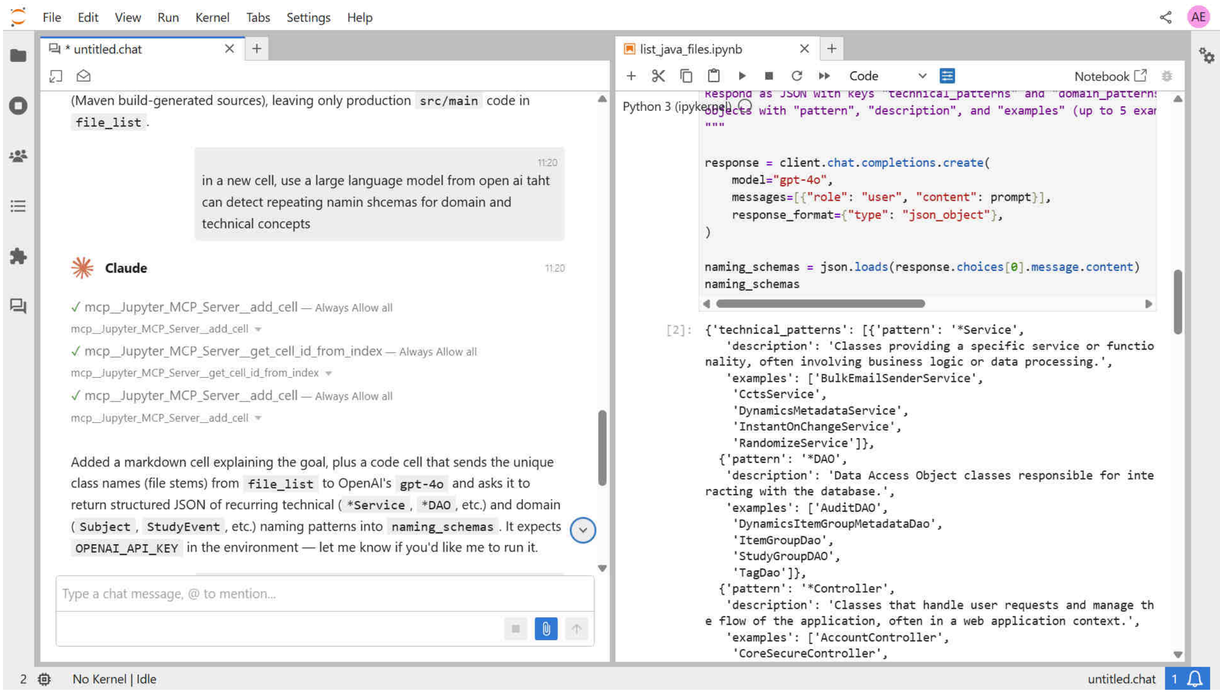


Von LLM-geschriebenes und von LLM-gestütztes Tool zum Erstellen von Abkürzungs-verzeichnissen

Beispiel: Guided AI mit Agents

Durch Scripting geführte KI / „LLM in the Loop“

- (KI-erzeugtes) Skript identifiziert **deterministisch** problematische Stellen in der großen Code-Basis
- Agent / LLM analysiert / ändert Fundstellen **nicht-deterministisch**



Jupyter AI mit Claude Code zur Identifikation von Querschnittskonzepten im Code

File Edit View Run Kernel Tabs Settings Help

AE

* untitled.chat

(Maven build-generated sources), leaving only production `src/main` code in `file_list`.

11:20

in a new cell, use a large language model from open ai taht can detect repeating namin shcemas for domain and technical concepts

Claude 11:20

✓ mcp__Jupyter_MCP_Server__add_cell — Always Allow all

mcp__Jupyter_MCP_Server__add_cell

✓ mcp__Jupyter_MCP_Server__get_cell_id_from_index — Always Allow all

mcp__Jupyter_MCP_Server__get_cell_id_from_index

✓ mcp__Jupyter_MCP_Server__add_cell — Always Allow all

mcp__Jupyter_MCP_Server__add_cell

Added a markdown cell explaining the goal, plus a code cell that sends the unique class names (file stems) from `file_list` to OpenAI's `gpt-4o` and asks it to return structured JSON of recurring technical (`*Service` , `*DAO` , etc.) and domain (`Subject` , `StudyEvent` , etc.) naming patterns into `naming_schemas` . It expects `OPENAI_API_KEY` in the environment — let me know if you'd like me to run it.

Type a chat message, @ to mention...

list_java_files.ipynb

Code

Notebook

Python 3 (ipykernel)

```
Respond as JSON with keys "technical_patterns" and "domain_patterns"
objects with "pattern", "description", and "examples" (up to 5 exam
"""

response = client.chat.completions.create(
    model="gpt-4o",
    messages=[{"role": "user", "content": prompt}],
    response_format={"type": "json_object"},
)

naming_schemas = json.loads(response.choices[0].message.content)
naming_schemas
```

[2]:

```
{'technical_patterns': [{'pattern': '*Service',
   'description': 'Classes providing a specific service or functio
nality, often involving business logic or data processing.',
   'examples': ['BulkEmailSenderService',
    'CctsService',
    'DynamicsMetadataService',
    'InstantOnChangeService',
    'RandomizeService']},
  {'pattern': '*DAO',
   'description': 'Data Access Object classes responsible for inte
racting with the database.',
   'examples': ['AuditDAO',
    'DynamicsItemGroupMetadataDao',
    'ItemGroupDao',
    'StudyGroupDAO',
    'TagDao']},
  {'pattern': '*Controller',
   'description': 'Classes that handle user requests and manage th
e flow of the application, often in a web application context.',
   'examples': ['AccountController',
    'CoreSecureController',
```

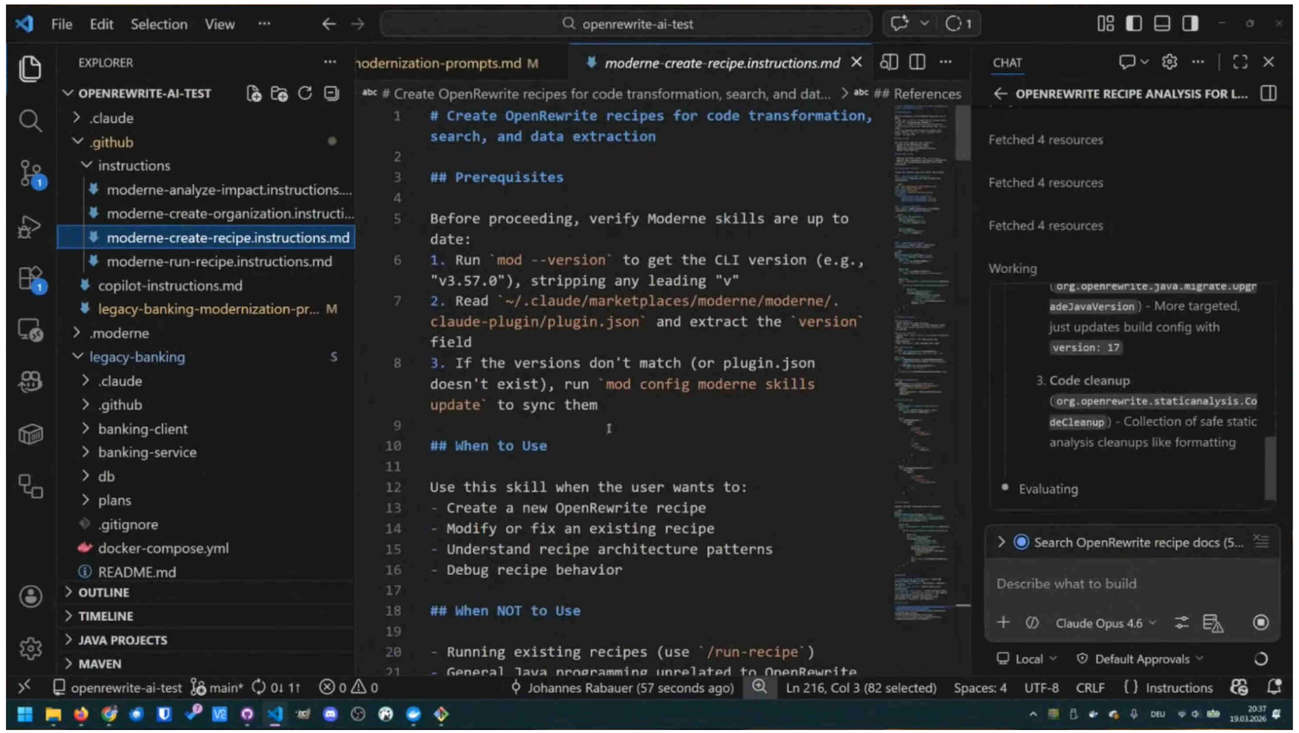
2 No Kernel | Idle

untitled.chat 1

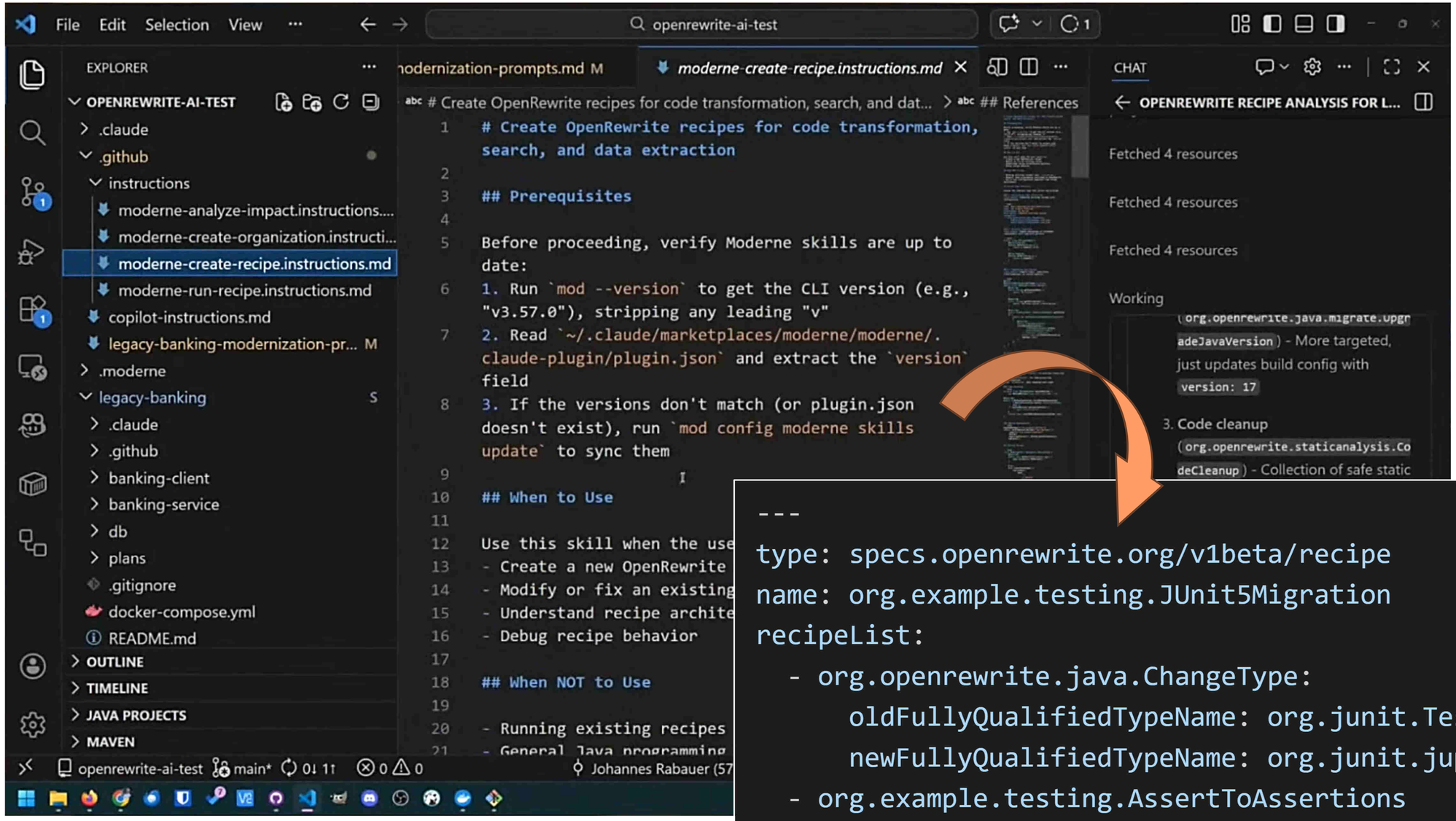
Beispiel: Transformation Tools mit Agents

Durch Agenten erstellte Migrationsskripte

- Agent erstellt auf Basis von Analysen **nicht-deterministisch** Anweisungen für vorzunehmende Anpassungen / Rezepte
- Transformations-Tools führen Anweisungen **deterministisch** aus über die komplette Code-Basis



AI Assisted Java Modernization: Building OpenRewrite Recipes with Tim te Beek
https://www.youtube.com/watch?v=LpGguXsR93A




```yaml
---
type: specs.openrewrite.org/v1beta/recipe
name: org.example.testing.JUnit5Migration
recipeList:
  - org.openrewrite.java.ChangeType:
      oldFullyQualifiedTypeName: org.junit.Test
      newFullyQualifiedTypeName: org.junit.jupiter.api.Test
  - org.example.testing.AssertToAssertions
  - org.example.testing.RemovePublicTestModifiers
  - org.example.testing.AddJUnitDependencies
```

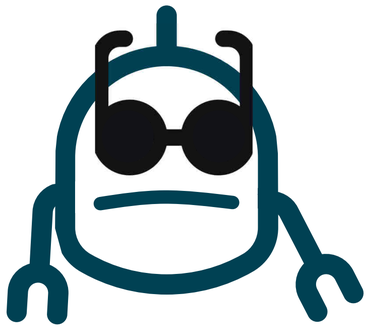
https://docs.openrewrite.org/concepts-and-explanations/recipes

Code „ready“ für Agenten machen

Coding Agenten im Einsatz

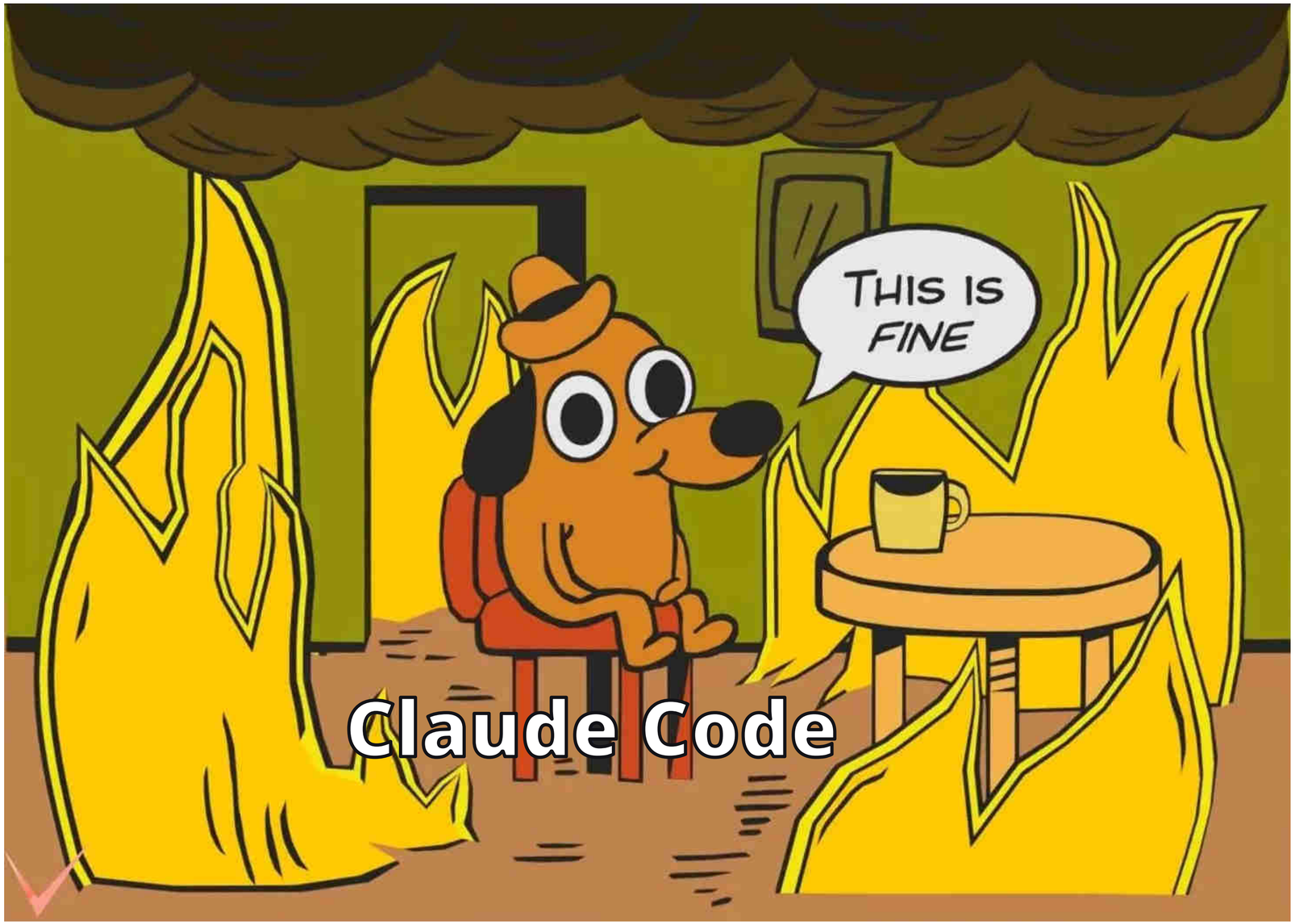
> "Claude Code is making use of agentic search
> ... using a combination of glob and grep.

Anthropic: Transform Legacy Systems into Strategic Assets - Code Modernization with AI
https://www.youtube.com/watch?v=8qtSeQuNv0o



Changing a mission-critical code chaos with search & replace.

What could possibly go wrong?



Strukturelle Navigation verbessern

✗ Schlechte globability

```
/src
├── BillingTeamsService.py
├── chargeTeamByCard.py
├── create-InvoiceData.py
├── datagatekeeper.PY
├── format-date-helpr.py
├── get-usr-prof.py
├── thequerymachine.py
└── user_discount_srv.py
```

✓ Gute globability

```
/src
├── /modules
│   └── /billing
│       ├── __init__.py
│       ├── charge_by_card_use_case.py
│       ├── billing_service.py
│       ├── invoice_repository.py
│       ├── create_invoice_dto.py
│       └── format_date_helper.py
└── /infra
    └── db_client.py
```

Strukturinformationen einbringen

Namenskonventionen in Klassen

```
public class CustomerRepository {
```

Namenskonventionen in Modulen

```
package org.springframework.samples.petclinic.repository;
```

Definition mittels Annotationen

```
@interface Repository {
```

```
@Repository
class Customer {
```

Definition mittels Marker-Interfaces

```
public interface Repository {
```

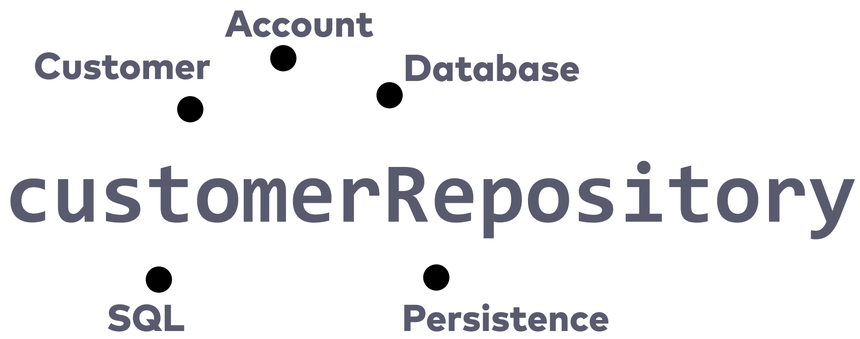
```
public class CustomerDAO implements Repository {
```

Inhaltliche Orientierung ausbauen

✗ Schlechte grepability

LLM erkennt rein technische Details.
Hat Schwierigkeiten, die Lücke zur Geschäftslogik zu überbrücken.

✓ Gute grepability



LLM erkennt passende Muster zum andocken.
Kurze Entfernung zu Geschäftskonzepten.

[1] Extrem stark vereinfacht

Semantische Anker nutzen

Präzise definierte Fachbegriffe, Methodiken und Frameworks für eine effektive LLM-Kommunikation.

[...]

Dienen als Referenzpunkte bei der Interaktion mit Large Language Models.[...]

Fungieren als gemeinsames Vokabular, das spezifische, kontextreiche Wissensdomänen innerhalb der Trainingsdaten eines LLMs gezielt aktiviert.

Übernommen und übersetzt von https://llm-coding.github.io/Semantic-Anchors/

Semantische Anker - Beispiele

- ✗ "Write tests“
- ✓ “Write test according to Behavior-Driven Development"

- ✗ “review architecture“
- ✓ “review my hexagonal architecture style“

- ✗ “improve database access"
- ✓ “migrate all DAOs to JPA repositories"

Teils übernommen und übersetzt von https://llm-coding.github.io/Semantic-Anchors/

Entwickler

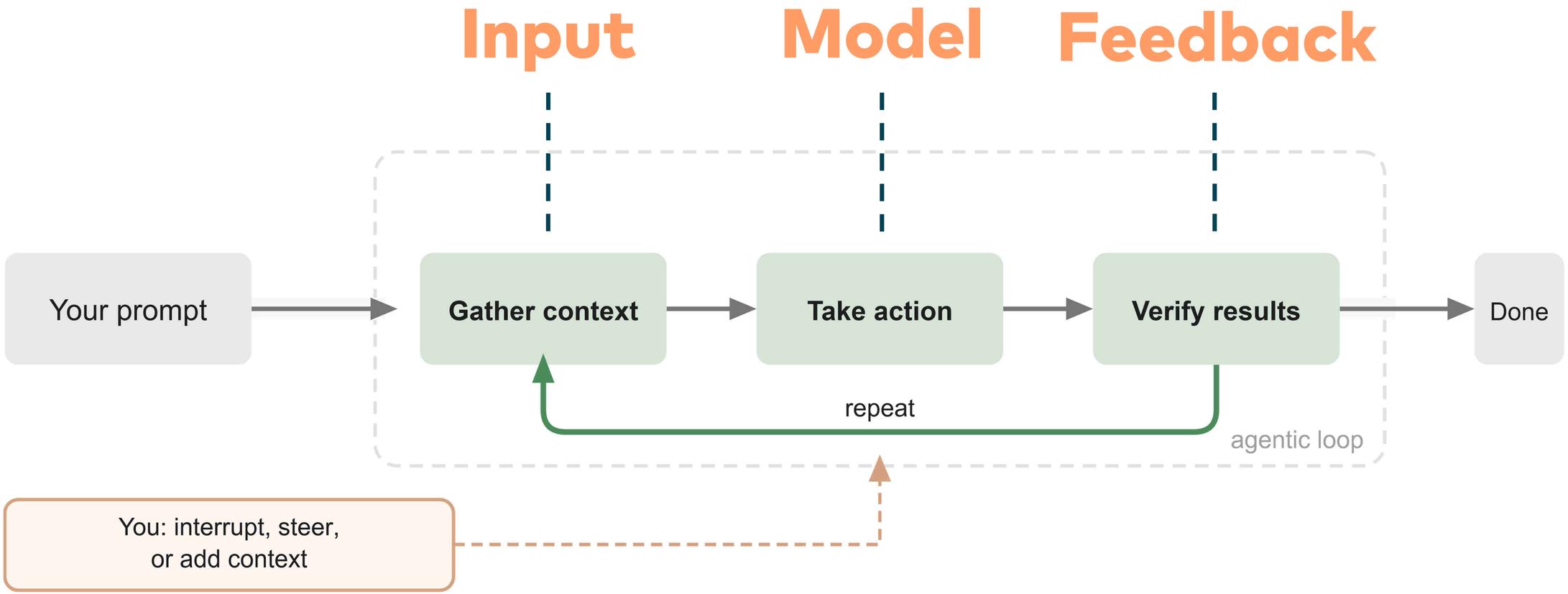
Wir wollen die Codequalität verbessern, damit wir unseren Code verstehen können

Wir wollen die Codequalität verbessern, damit AI unseren Code versteht

Manager

Agentischen Loop verbessern

Der „Agentic Loop"



Grafik von https://code.claude.com/docs/en/how-claude-code-works

Wichtig

> "The model is only as good as its context.

Programmierer

Anthropic: Transform Legacy Systems into Strategic Assets - Code Modernization with AI
https://www.youtube.com/watch?v=8qtSeQuNv0o

Agenten und ihr Kontext

Clean engineering environments allow AI to autonomously drive a larger share of the sprint

Clean Code Amplifies AI Gains

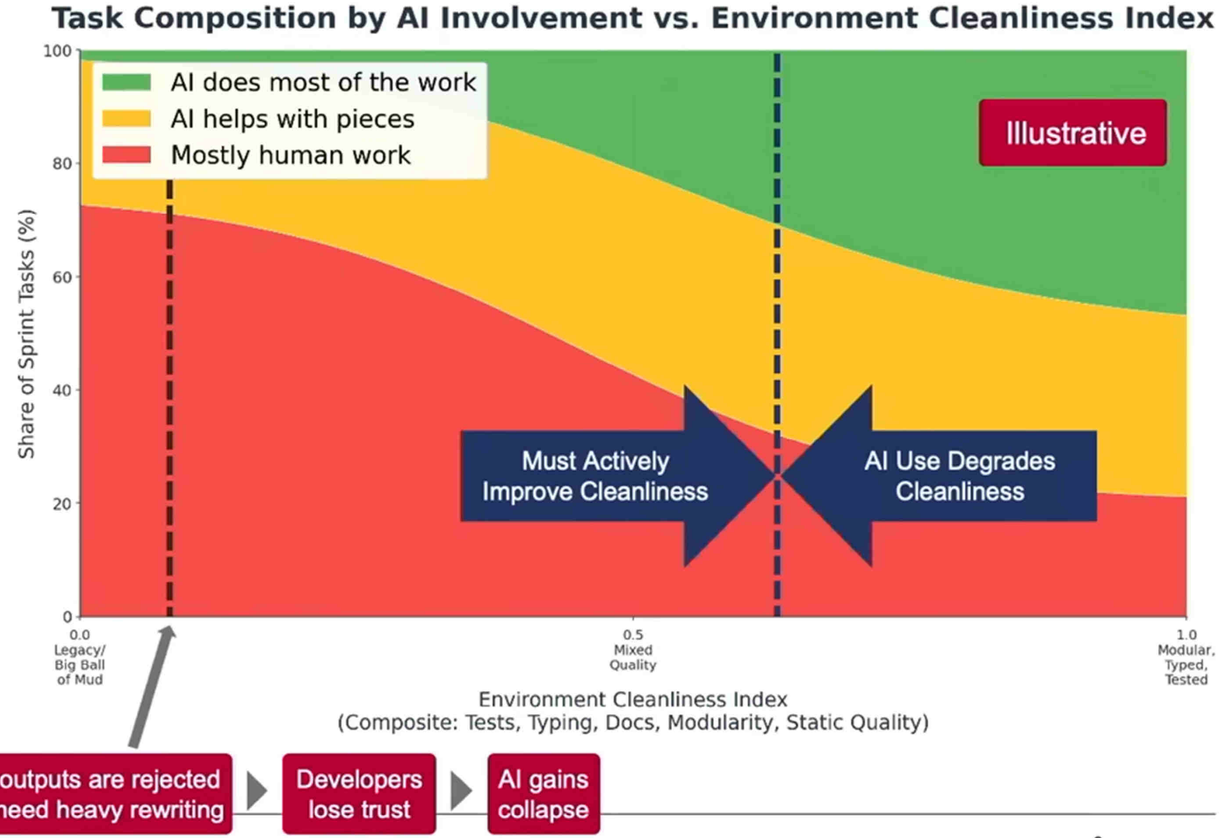
- A clean codebase allows AI to complete a larger share of sprint tasks

Manage Codebase Entropy

- Unchecked AI accelerates entropy (tech debt)
- High entropy degrades future AI performance

Engineers Must Master Task Selection

- Knowing when to use AI
- …and when to write code manually



Agenten und ihr Kontext

Clean Code Amplifies AI Gains

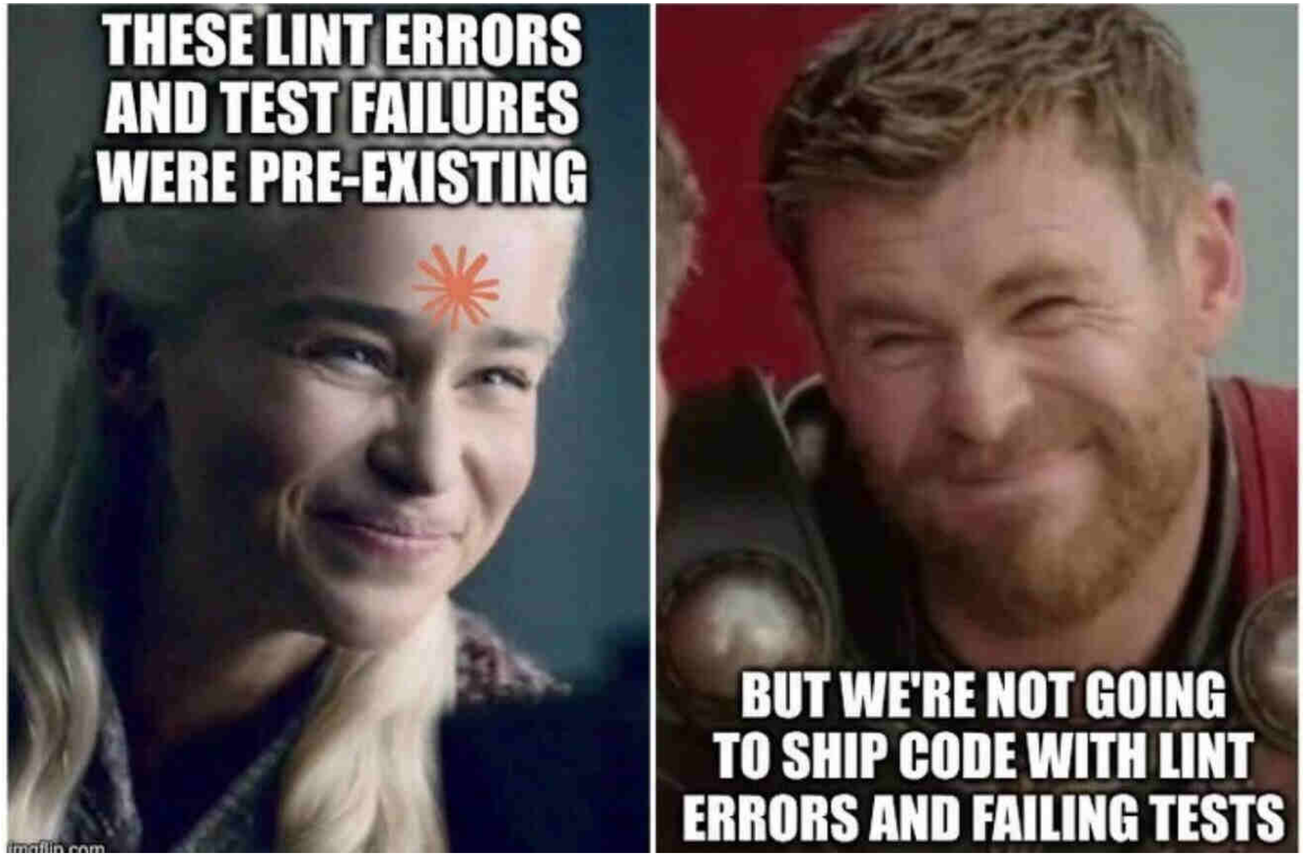
- A clean codebase allows AI to complete a larger share of sprint tasks

Manage Codebase Entropy

- Unchecked AI accelerates entropy (tech debt)
- High entropy degrades future AI performance

Engineers Must Master Task Selection

- Knowing when to use AI
- …and when to write code manually



https://www.reddit.com/r/ClaudeCode/comments/1rdfcyp/dont_worry_ive_got_all_day/

Über Harness Input & Feedback einbringen

Context Engineering

Kontinuierliches Anreichern der Wissensbasis in der Codebasis

Architectural Constraints

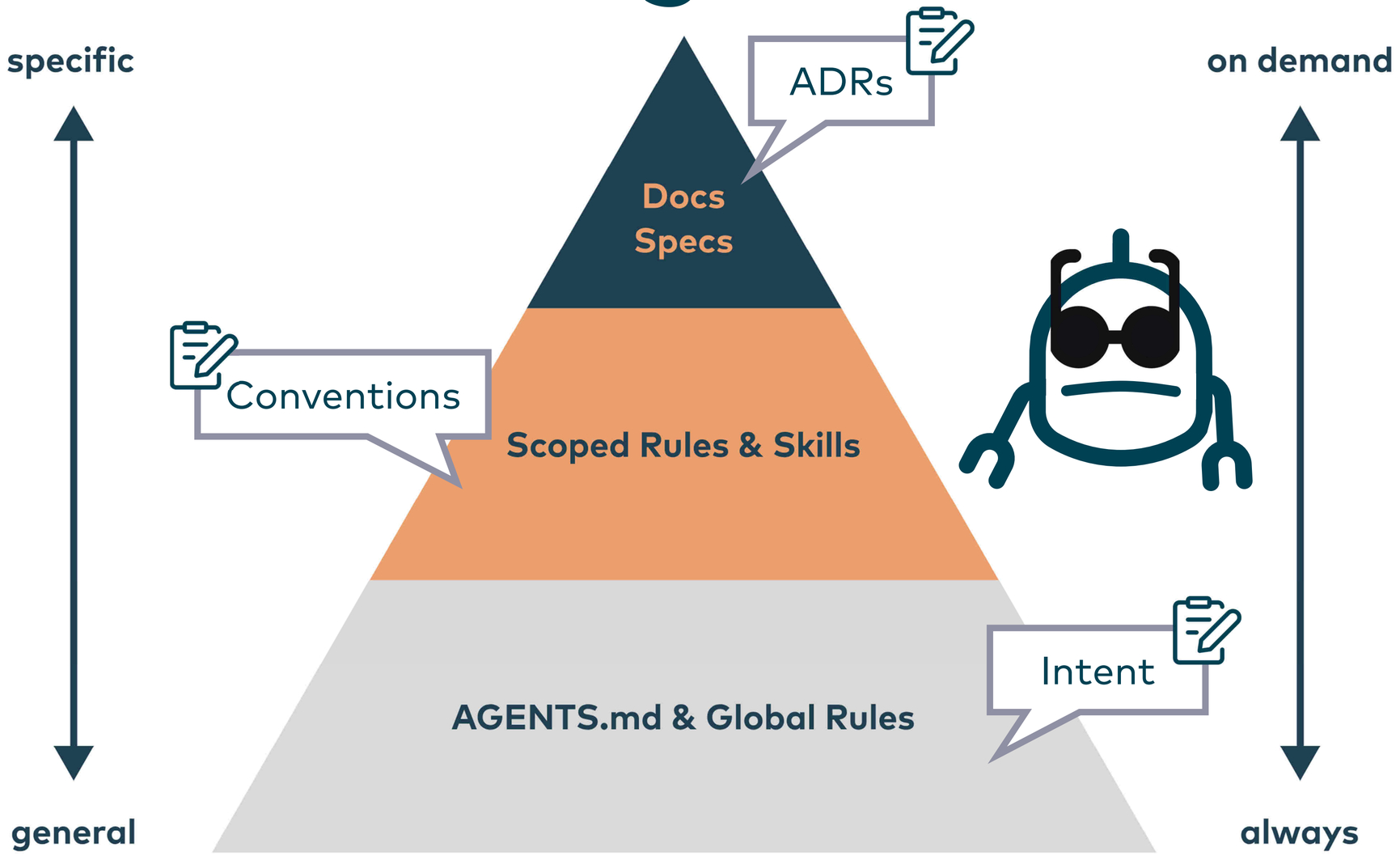
Überwachen durch LLM-basierte Agenten und deterministische Tools

Garbage Collection

Aufspüren von globalen Inkonsistenzen gegen lokale Entropie und schleichendem Verfall

inspiriert durch Birgitta Böckeler
https://martinfowler.com/articles/exploring-gen-ai/harness-engineering.html

Input vorher an Agenten liefern



https://www.innoq.com/de/blog/2026/04/vom-vibe-coder-zum-agentic-engineer/

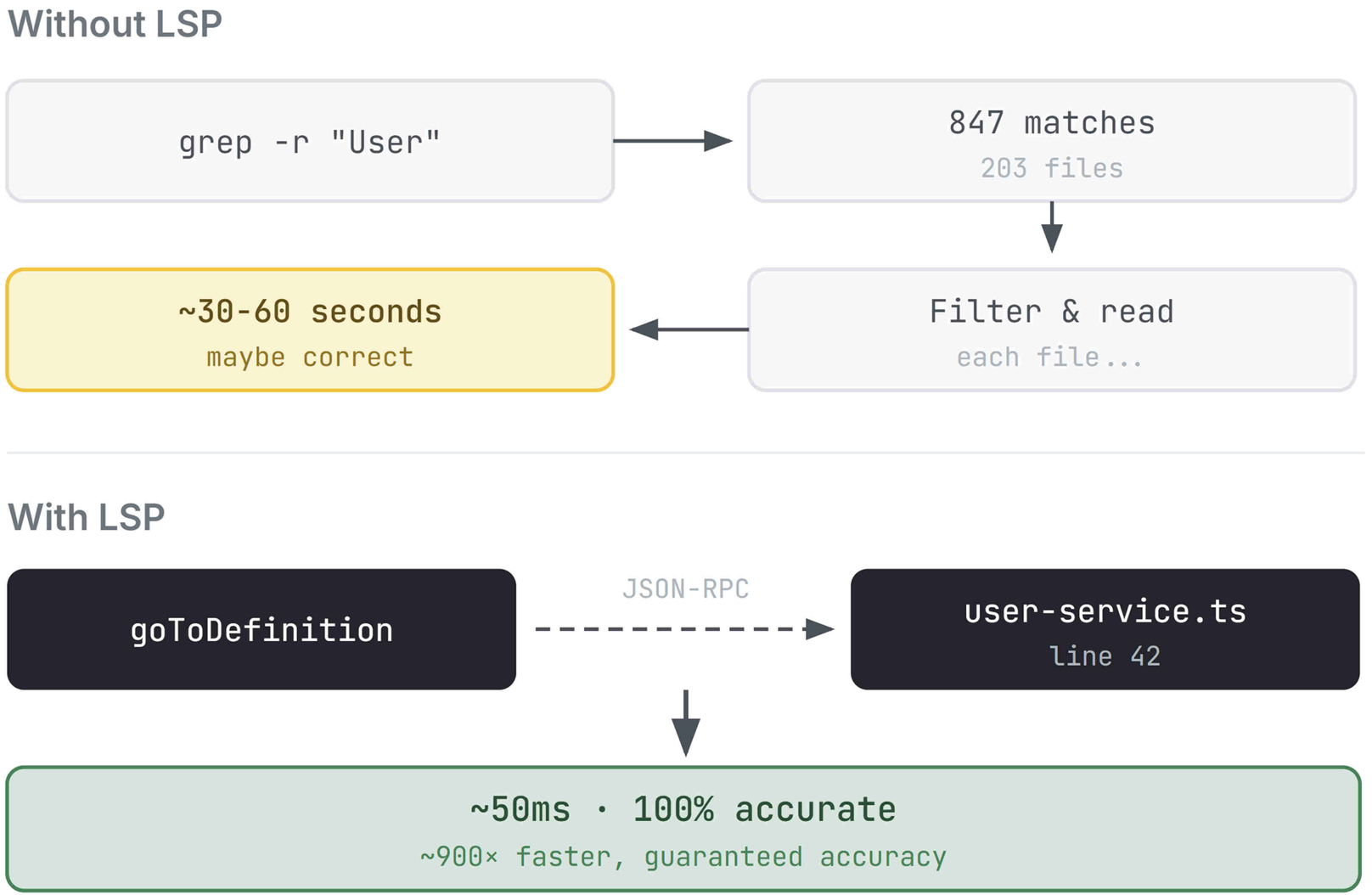
Language Server Protocol

> "real-time code intelligence while working on your codebase.

https://code.claude.com/docs/en/plugins-reference

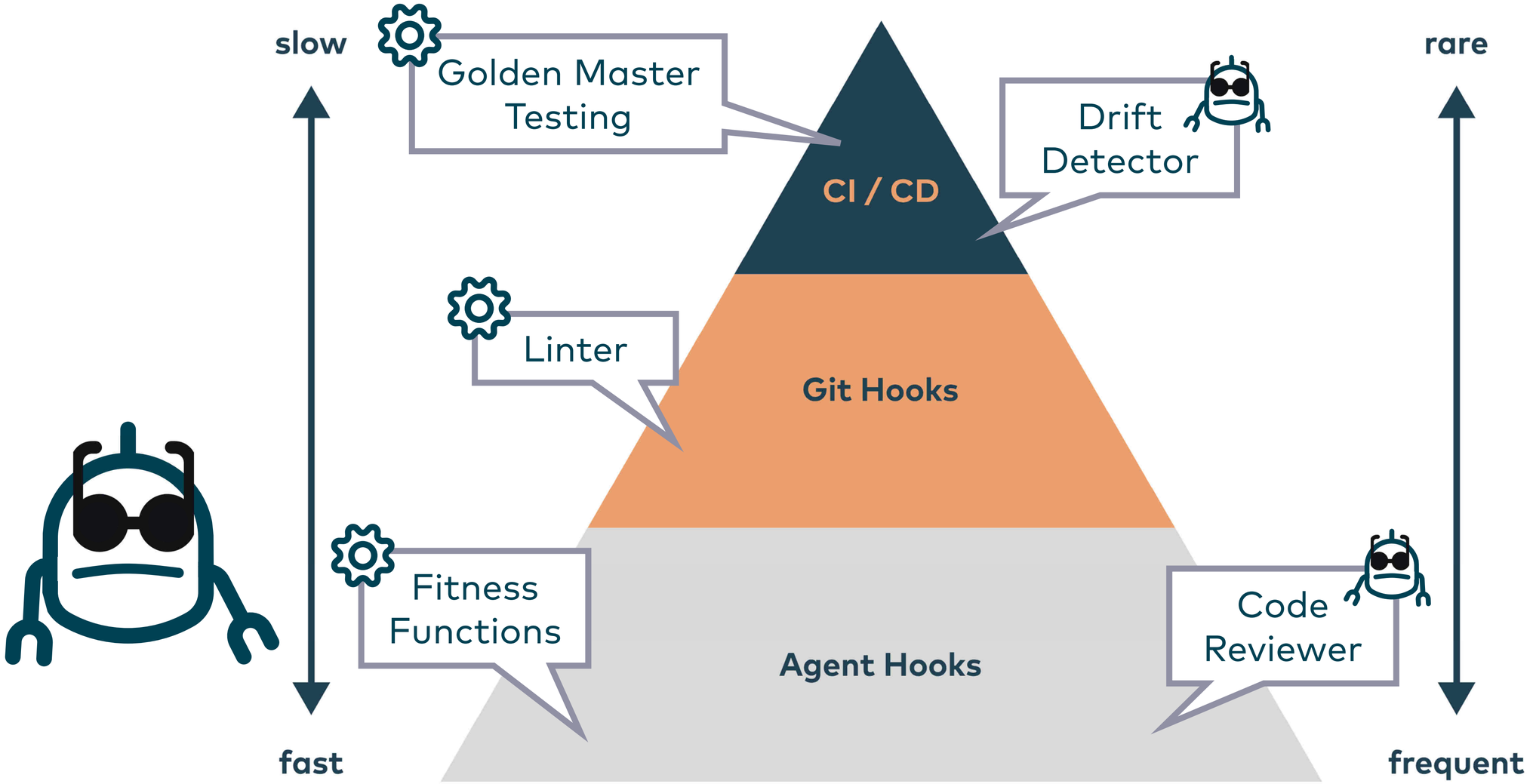
Language Server Protocol

als Input für Agenten



https://karanbansal.in/blog/claude-code-lsp/

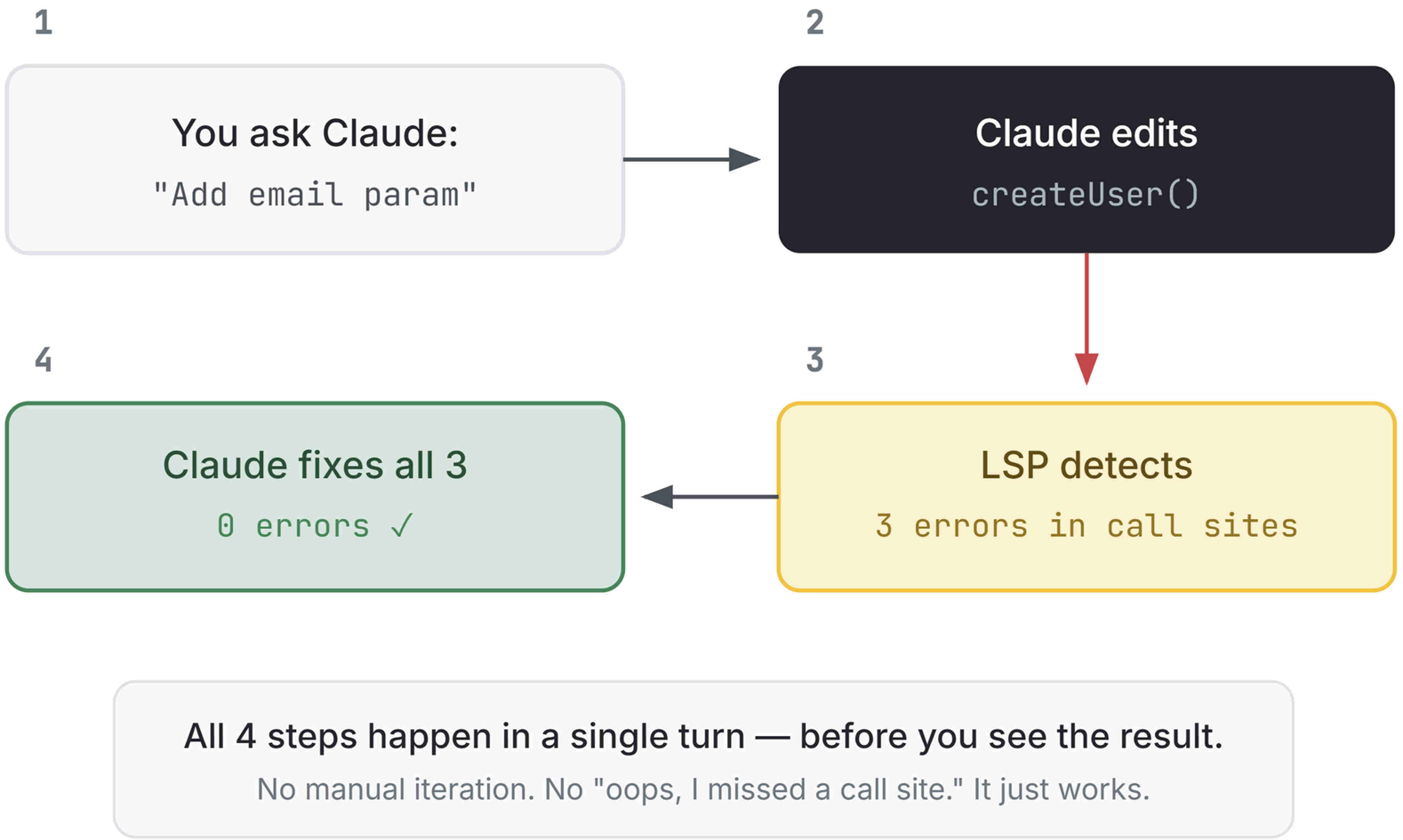
Feedback zurück an Agenten liefern



https://www.innoq.com/de/blog/2026/04/vom-vibe-coder-zum-agentic-engineer/

Language Server Protocol

als Feedback für Agenten



https://karanbansal.in/blog/claude-code-lsp/

Fitness Functions

Feedback für den Agenten über ArchUnit

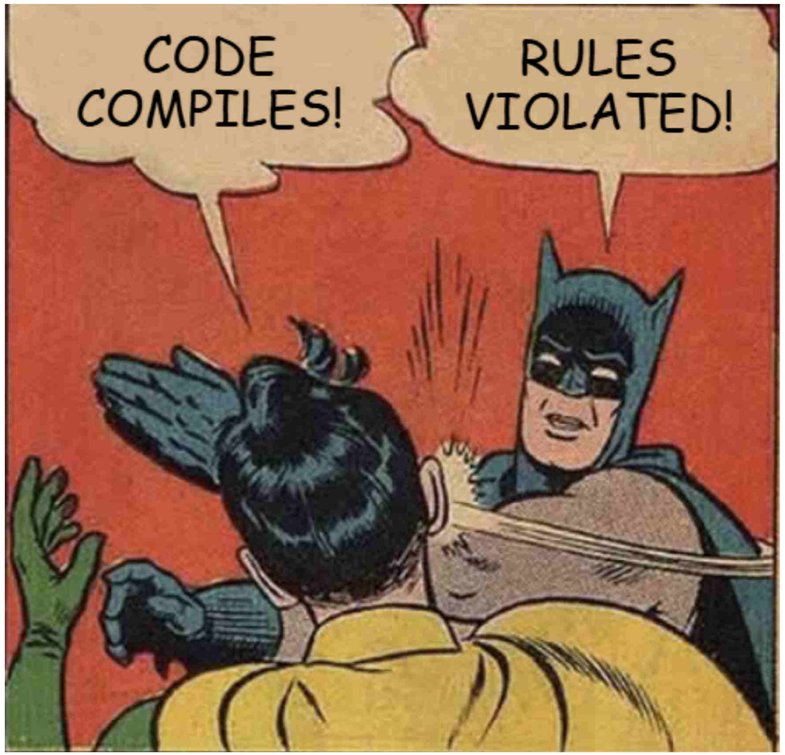
1. Erstellung von Architekturregeln mittels einer domänen-spezifischen Sprache (DSL) für strukturelle Checks

```
classes().that().areAnnotatedWith(Service.class)
         .or().haveNameMatching(".*Service")
         .should().resideInAPackage("..service..")
         .orShould().beAnnotatedWith(LegacyBridge.class);
```

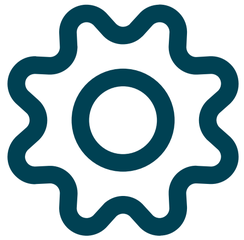
2. Prüfung von Architekturverletzungen zur Testzeit über einen Hook für den Agenten, wo Build darüber angestoßen wird.

3. Interpretation und Rückmeldung von Architekturverletzungen über Subagent an den Haupt-Agenten

```
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running com.example.ServiceLocationRuleFailingTest
[main] INFO com.tngtech.archunit.core.PluginLoader - Detected Java
version 25.0.2
[ERROR] Tests run: 1, Failures: 1, Errors: 0, Skipped: 0, Time elap
sed: 5.209 s <<< FAILURE! -- in com.example.ServiceLocationRuleFail
ingTest
[ERROR] com.example.ServiceLocationRuleFailingTest.allClassesMustFo
llowServiceLocationRule -- Time elapsed: 4.297 s <<< FAILURE!
java.lang.AssertionError:
Architecture Violation [Priority: MEDIUM] - Rule 'classes that are
annotated with @Service or have name matching '.*Service' should re
side in a package '..service..' or should be annotated with @Legacy
Bridge' was violated (2 times):
Class <com.example.controller.PaymentService> does not reside in a
package '..service..' in (PaymentService.java:0) and Class <com.exa
mple.controller.PaymentService> is not annotated with @LegacyBridge
 in (PaymentService.java:0)
Class <com.example.util.CacheManager> does not reside in a package
'..service..' in (CacheManager.java:0) and Class <com.example.util.
CacheManager> is not annotated with @LegacyBridge in (CacheManager.
java:0)
```



Goldene Regel für Harnesses



If you can tool it, tool it.
If you can’t, prompt it.

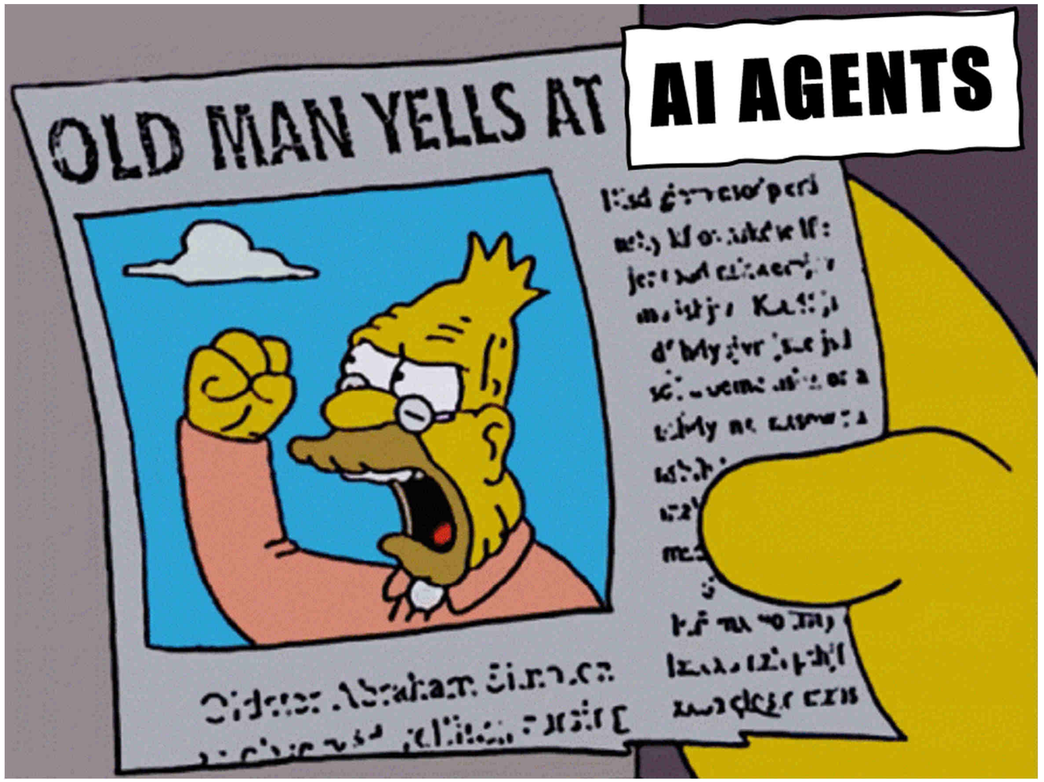
Torben Keller

https://www.innoq.com/de/blog/2026/04/vom-vibe-coder-zum-agentic-engineer/

Zusammenfassung

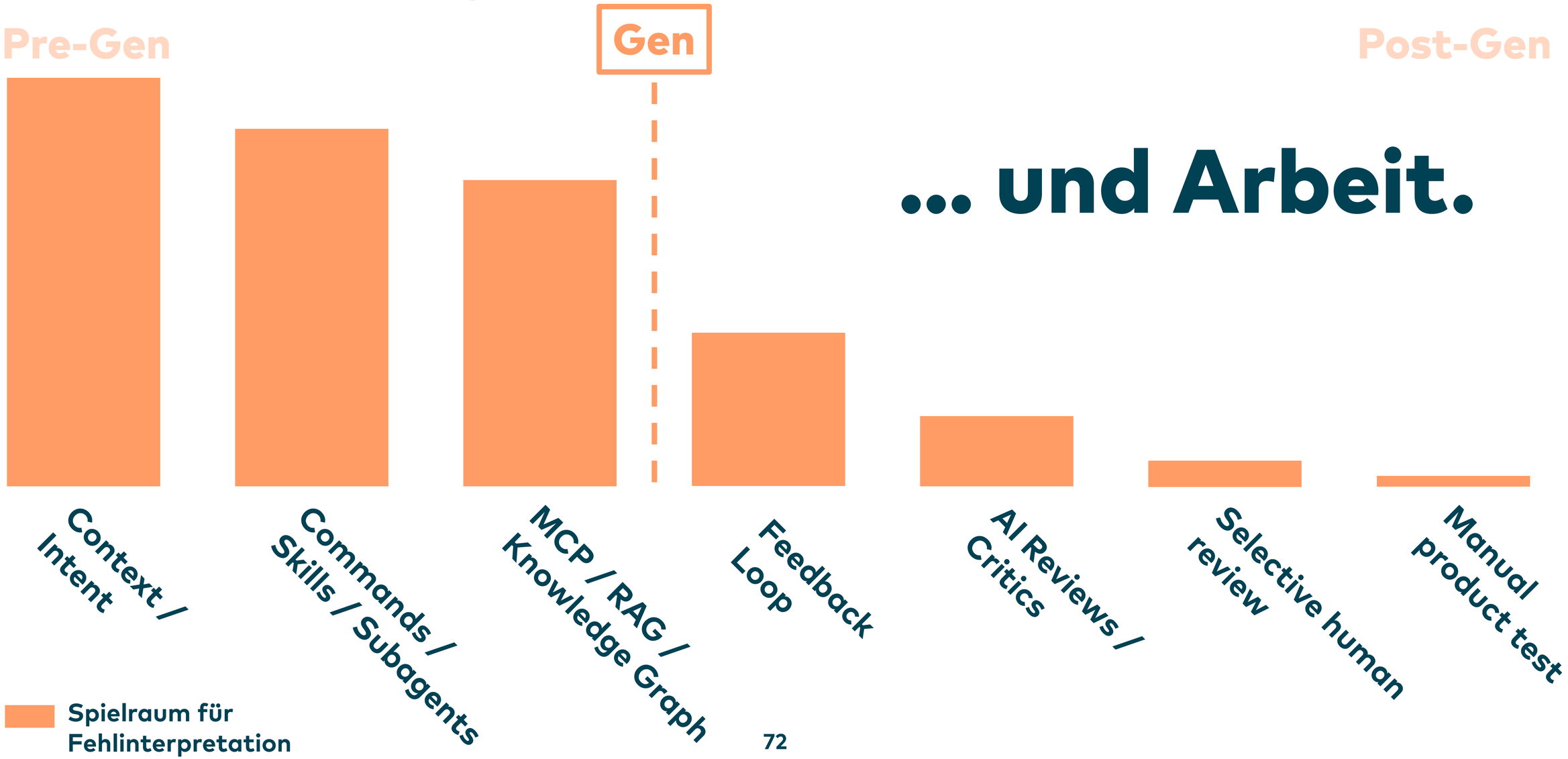
Agentic Software Modernization

Ich hoffe, der Talk hatte sich nicht so angehört …



agentisch

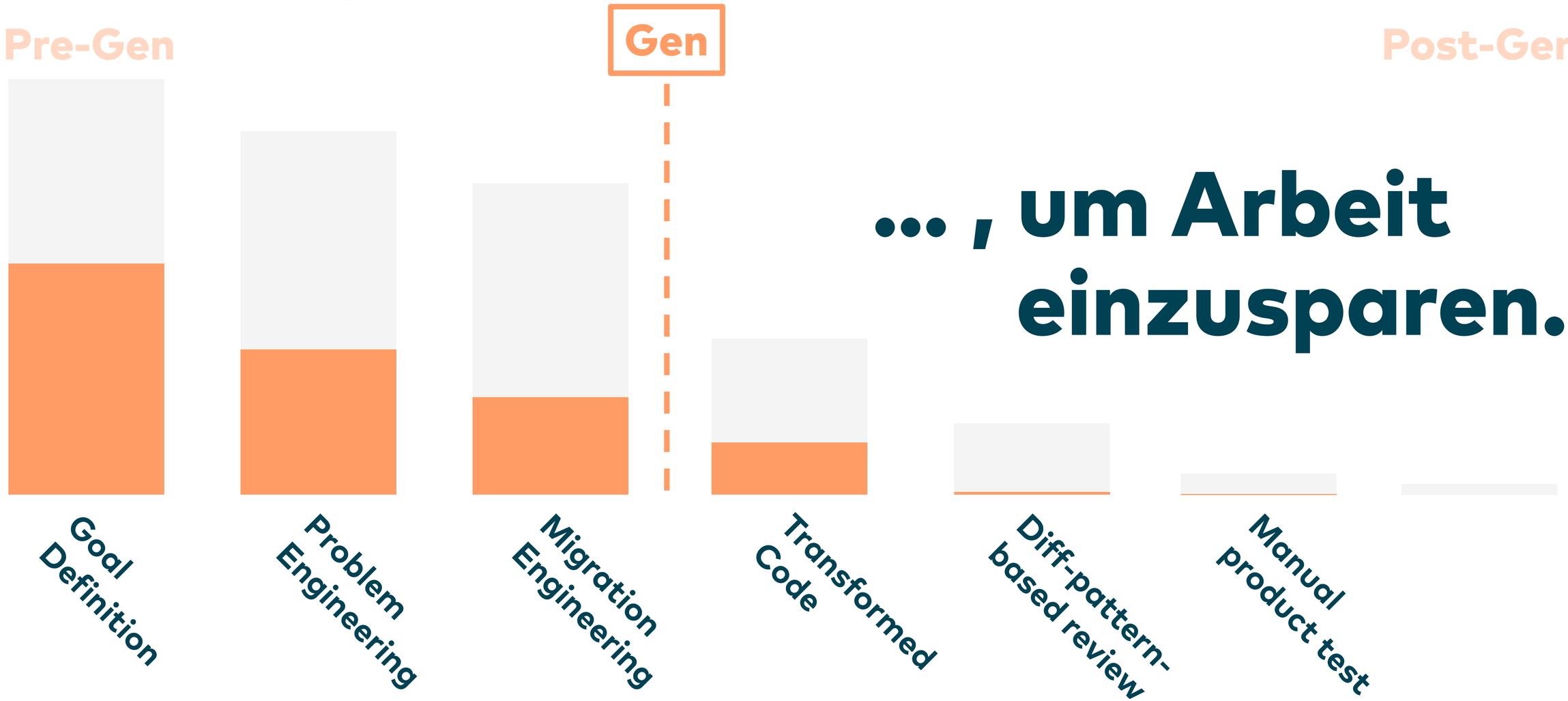
… denn es gibt viel Potenzial



… und Arbeit.

hybridisch

Auch: Spannende Alternativen



… , um Arbeit einzusparen.

Vielen Dank!

Fragen, Diskussionen, Anmerkungen ?

Networking

Markus Harrer

Anhang

Agents Grundlagen

Agentic – technisch gesehen

Goal-Driven bekommt Ziel / Auffor-
derung und legt los bis Zielerreichung

Structured Output
definierte Strukturen und
Formate als Ausgaben

Memory kurzfristig (Kontext)
und langfristig (externer Speicher)

Tool Use kann externe
Werkzeuge verwenden

Multi-Step Planning
zerlegt komplexe Aufgaben
selbstständig in Teilschritte

Self-Correction
erkennt Fehler und
korrigiert den eigenen Kurs

Reasoning Loop
Reason → Act → Observe

Grundmechanik – AGENTS.md*

Instruktionen und Kontext für den Agenten

```
# CLAUDE.md

## 1. Think Before Coding

**Don't assume. Don't hide confusion. Surface tradeoffs.**

Before implementing:
- State your assumptions explicitly. If uncertain, ask.
- If multiple interpretations exist, present them - don't pick silently.
- If a simpler approach exists, say so. Push back when warranted.
- If something is unclear, stop. Name what's confusing. Ask.
...
```

Auszug von https://github.com/multica-ai/andrej-karpathy-skills/blob/main/CLAUDE.md

* oder CLAUDE.md bei Anthropic

Grundmechanik – Commands

Gespeicherte Anweisungen, die Entwickler als Befehl aufrufen

```
---
allowed-tools: Bash(./scripts/gh.sh:*), Bash(./scripts/comment-on-duplicates.sh:*)
description: Find duplicate GitHub issues
---

Find up to 3 likely duplicate issues for a given GitHub issue.

To do this, follow these steps precisely:

1. Use an agent to check if the Github issue (a) is closed, (b) does not need to
be deduped (eg. because it is broad product feedback without a specific solution,
or positive feedback), or (c) already has a duplicates comment that you made
earlier. If so, do not proceed.
...
```

Auszug von https://github.com/anthropics/claude-code/blob/main/.claude/commands/dedupe.md

Grundmechanik – Skills

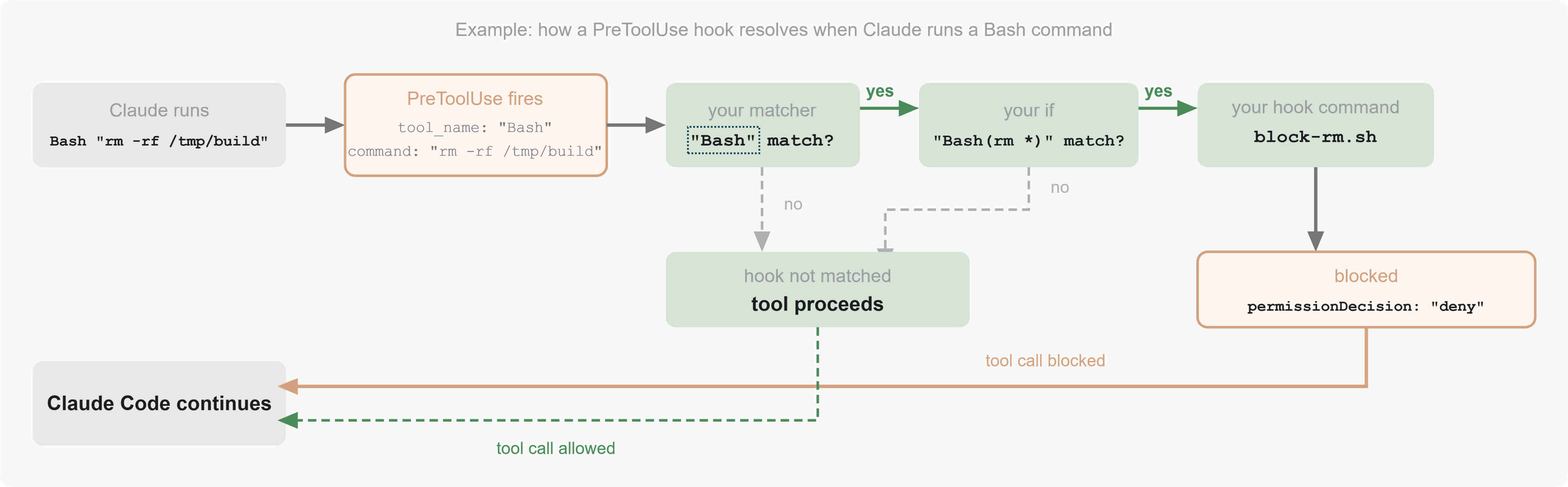
Umfangreichere Aktivitäten, die ein Agent bei bestimmten Trigger-Wörter ausführt / nutzt

```
---
name: grill-me
description: Interview the user relentlessly about a plan or design until reaching
shared understanding, resolving each branch of the decision tree. Use when user
wants to stress-test a plan, get grilled on their design, or mentions "grill me".
---

Interview me relentlessly about every aspect of this plan until we reach a shared
understanding. Walk down each branch of the design tree, resolving dependencies
between decisions one-by-one. For each question, provide your recommended answer.

Ask the questions one at a time. ...
```

Auszug von https://github.com/mattpocock/skills/blob/main/skills/productivity/grill-me/SKILL.md

Grundmechanik – Hooks

Aktivitäten, welche ein Agent an bestimmten Lifecycle-Punkten automatisch auslöst



Grafik von https://code.claude.com/docs/en/hooks

Weitere Grundlagen

Überblick

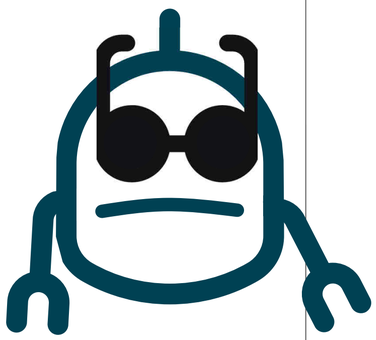
- **Plugins**
  Verteilungsmöglichkeit für die Erweiterung von paketierten Agenten-Fähigkeiten
- **RAG** Retrieval-Augmented Generation
  Anbindung weiterer Infos, um Agenten Zugriff auf spezifische Informationen zu geben
- **MCP** Model Context Protocol
  externe Tools einbinden, um die Fähigkeiten und das Wissen von Agenten zu erweitern
- **Sub Agents / Agent Teams**
  Agenten, die miteinander zusammenarbeiten
- **Sandboxing**
  Geschützte Ausführungsumgebung (Container, Firewall, Workspace) für Agents

Anhang

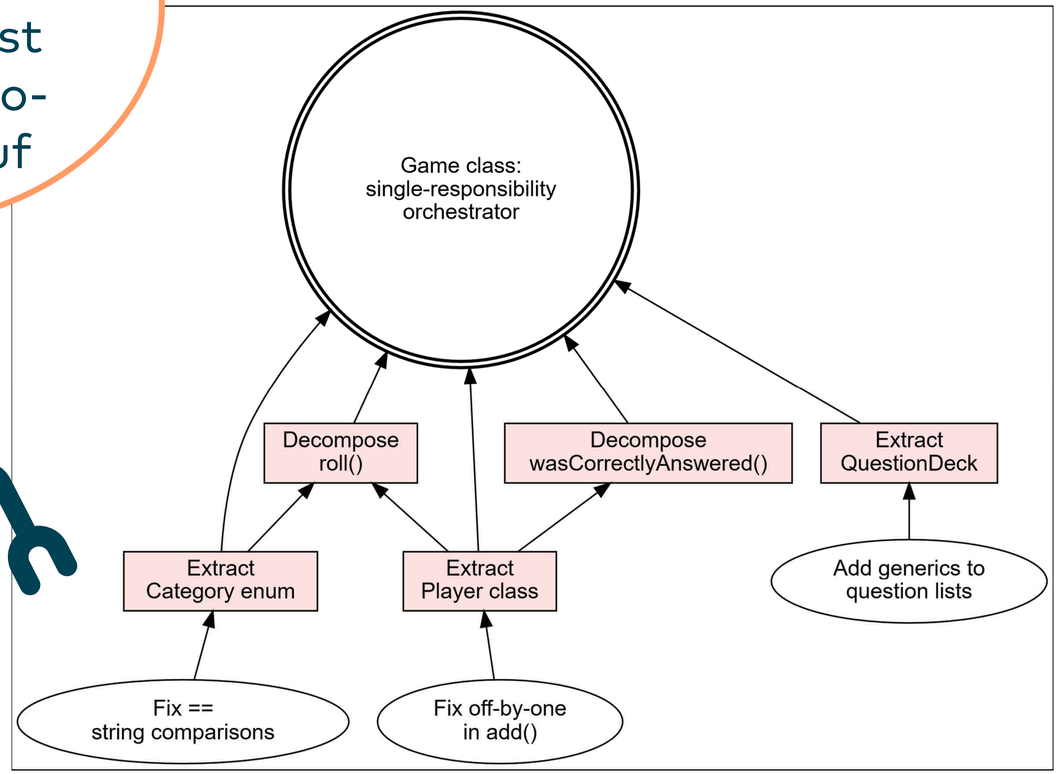
Mikado-Refactoring Skill
mit Characterization Testing Hook
und Fehleranalyse-Subagent
sowie gevibecodetem Dashboard

Kleines Beispiel

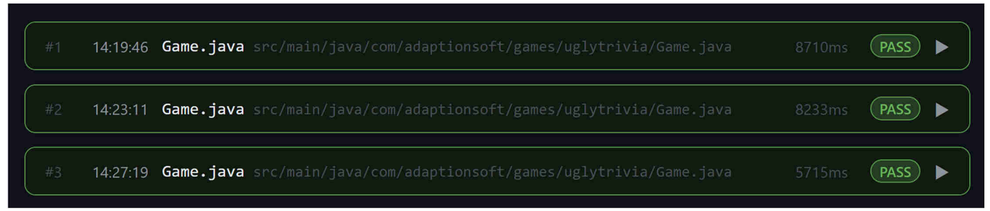
Zerteile diese Gottklasse. Baue dir hierfür zuerst einen Mikado-Graphen auf



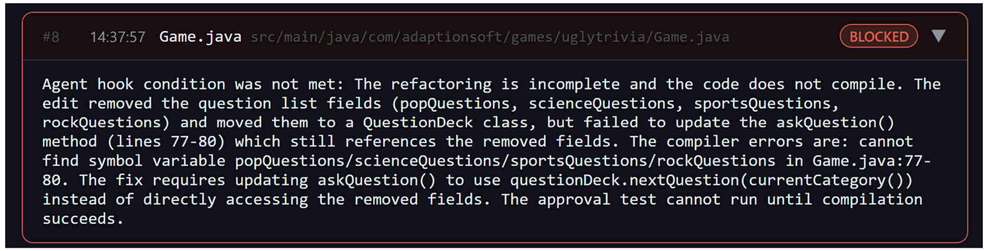
(1) skill: mikado-graph



(2) hook: PostToolUse
-> run-approval-test.sh



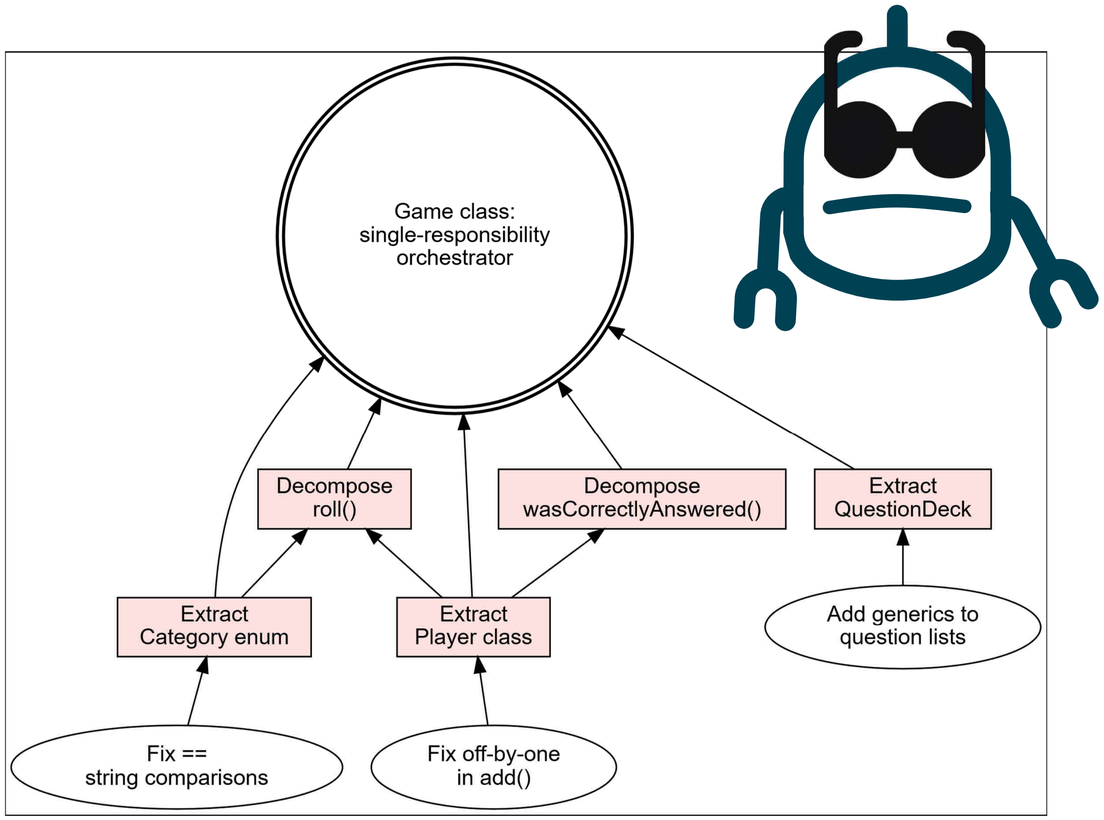
(3) subagent:
-> test analysis agent



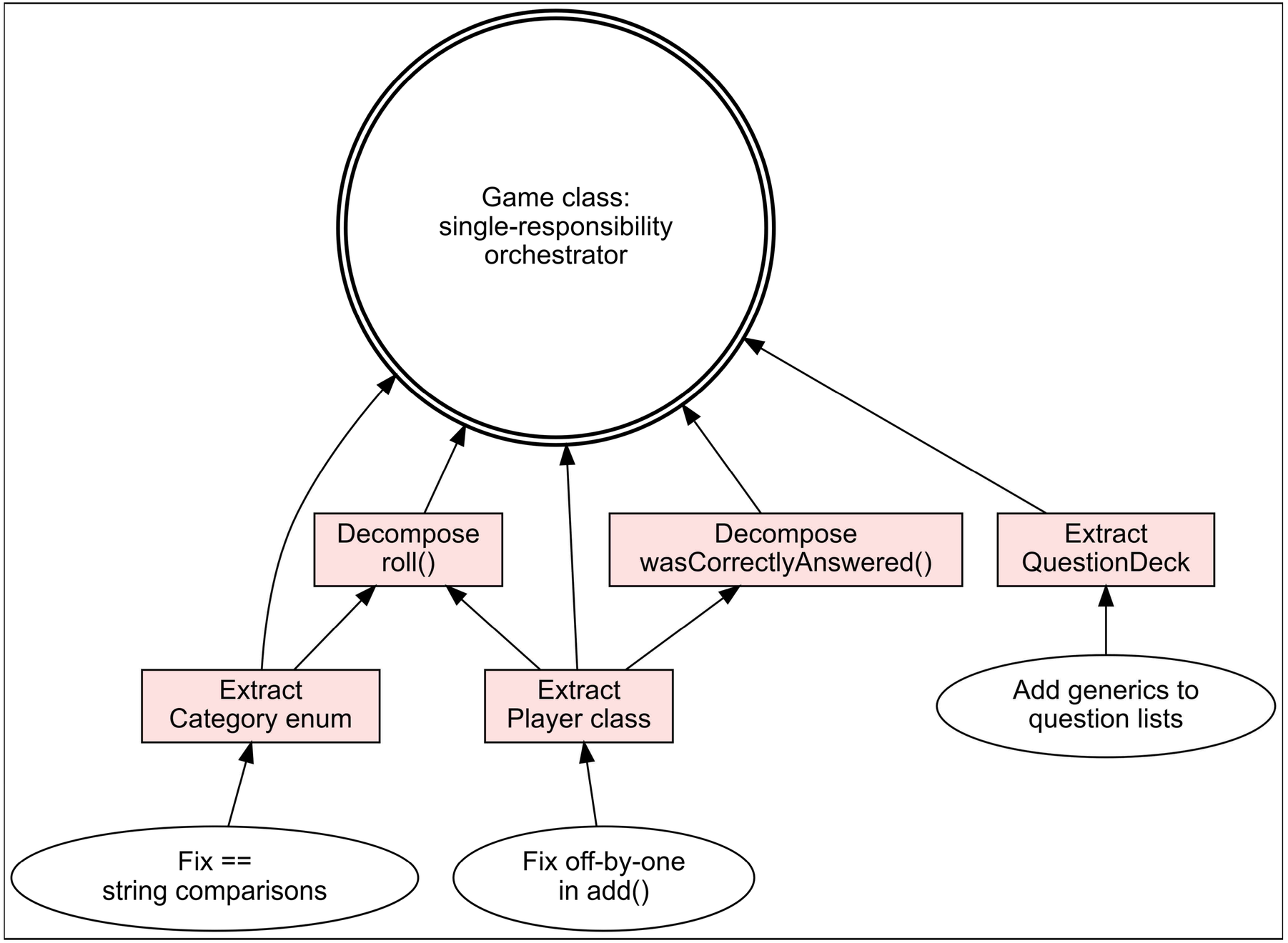
https://github.com/feststelltaste/trivia/tree/agentic-golden-master/java/.claude

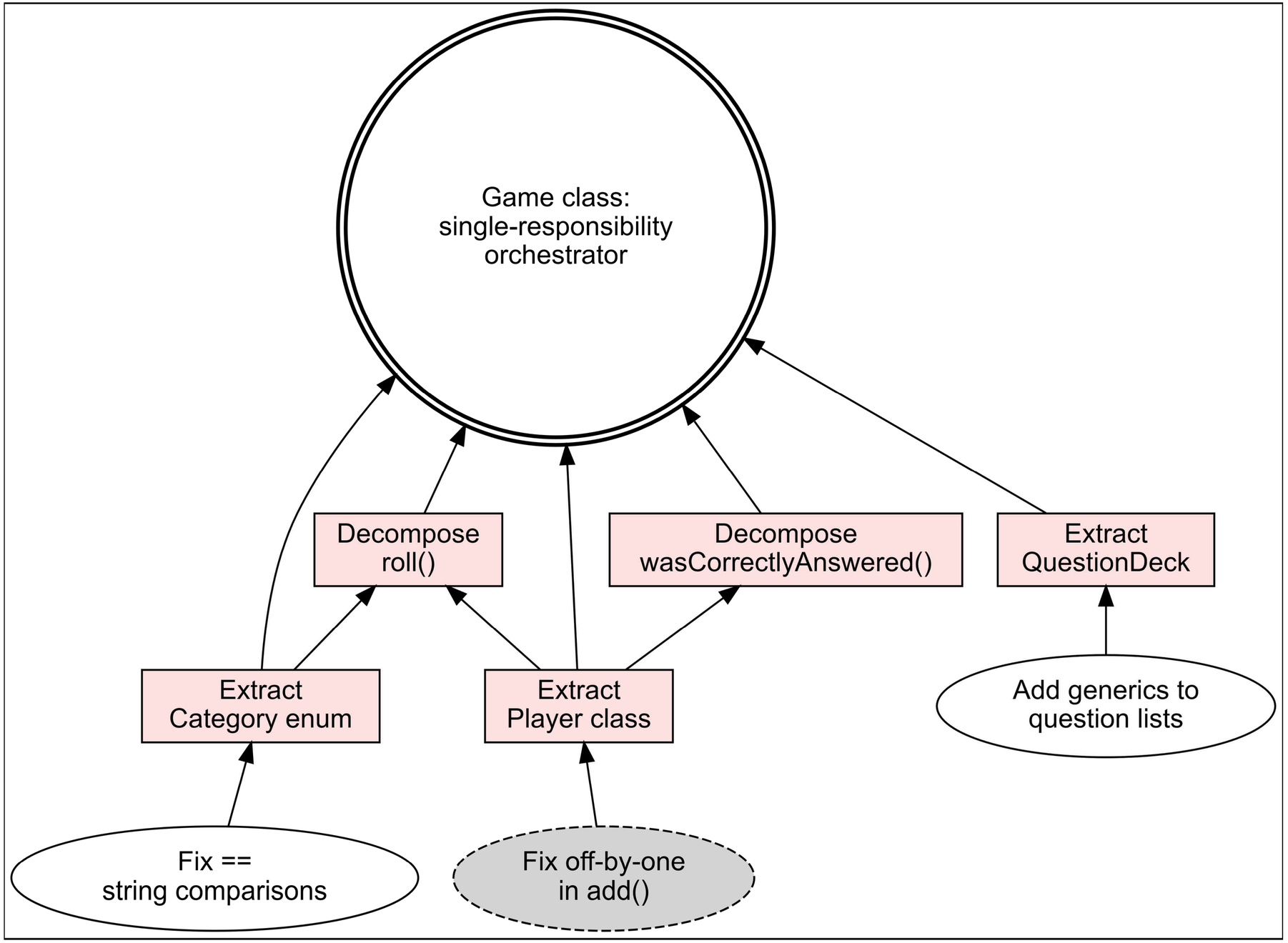
Kleines Beispiel: Mikado-Refactoring

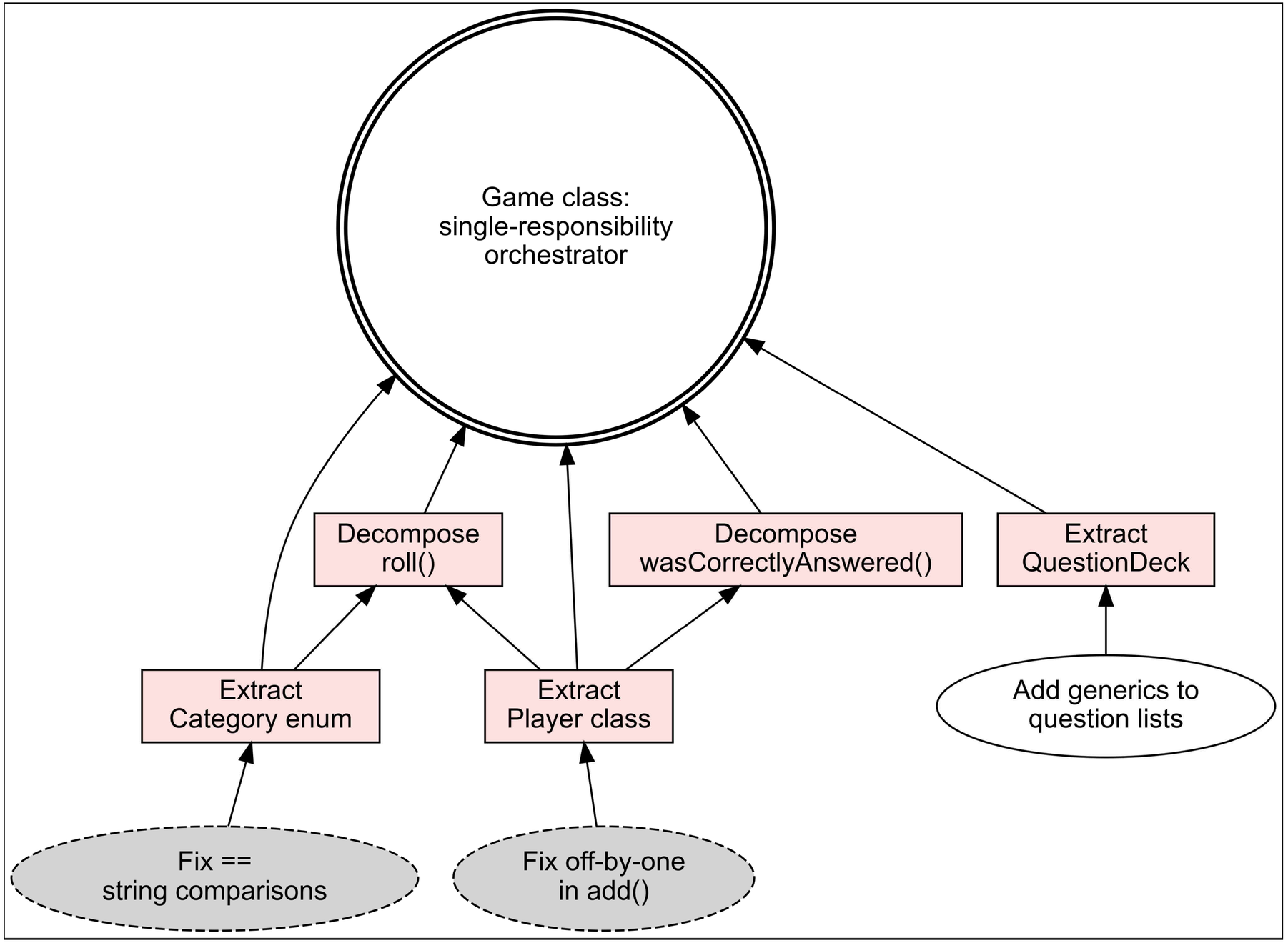
(1) skill: mikado-graph

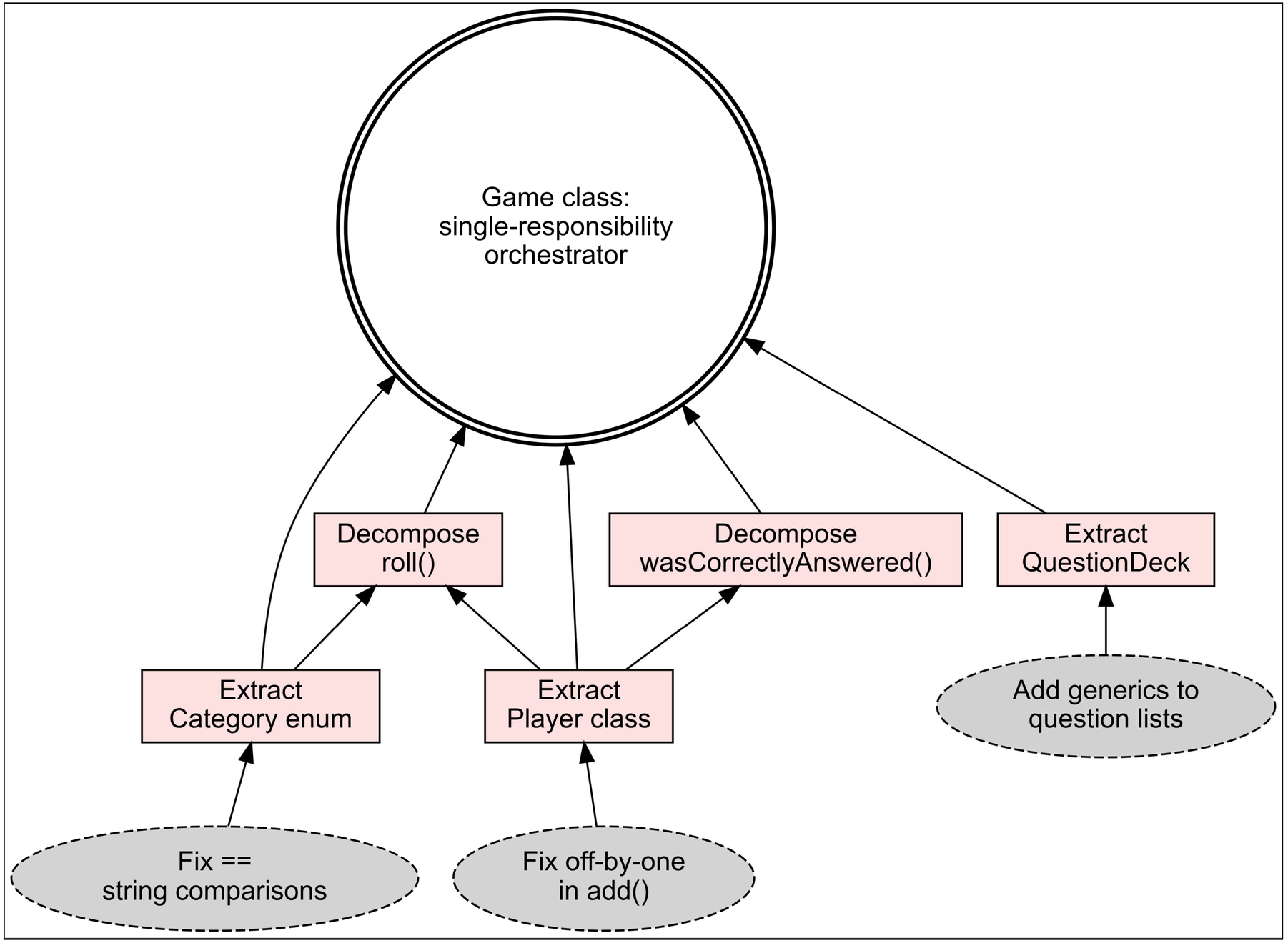


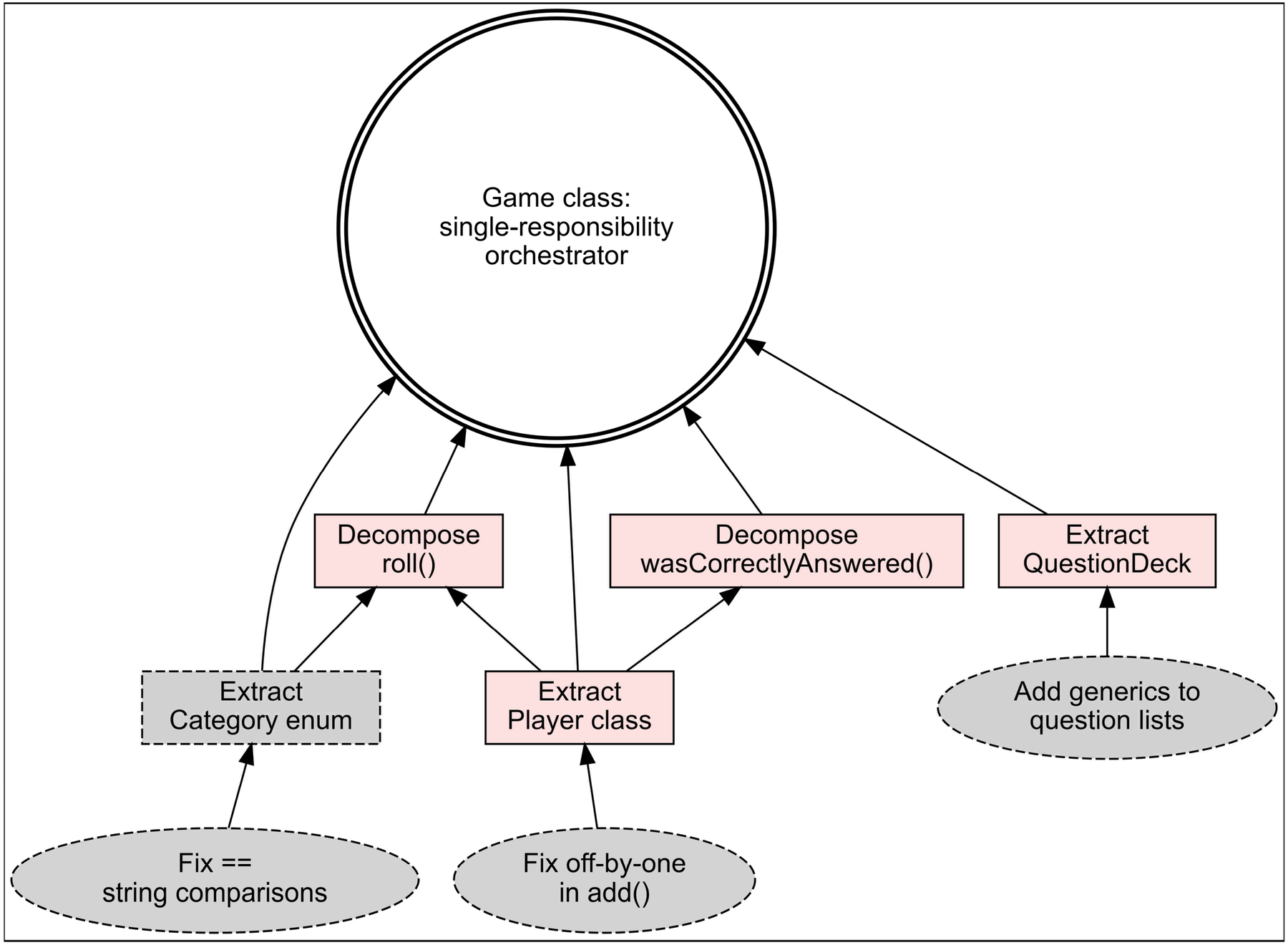

```
---
name: mikado-graph
description: Plan and track a complex refactoring using
the Mikado Method. Trigger when the user mentions
"mikado", wants to plan a refactoring or upgrade
incrementally, needs to untangle blockers before
changing code, or asks to track dependencies visually.
---

# Mikado Graph

The Mikado Method is an incremental refactoring
technique: try a change, note what breaks, revert, then
fix the blockers first. This skill tracks that process
as a dependency graph — each node is a goal, a blocker,
or a leaf task you can act on right now.

## Node types
...
```

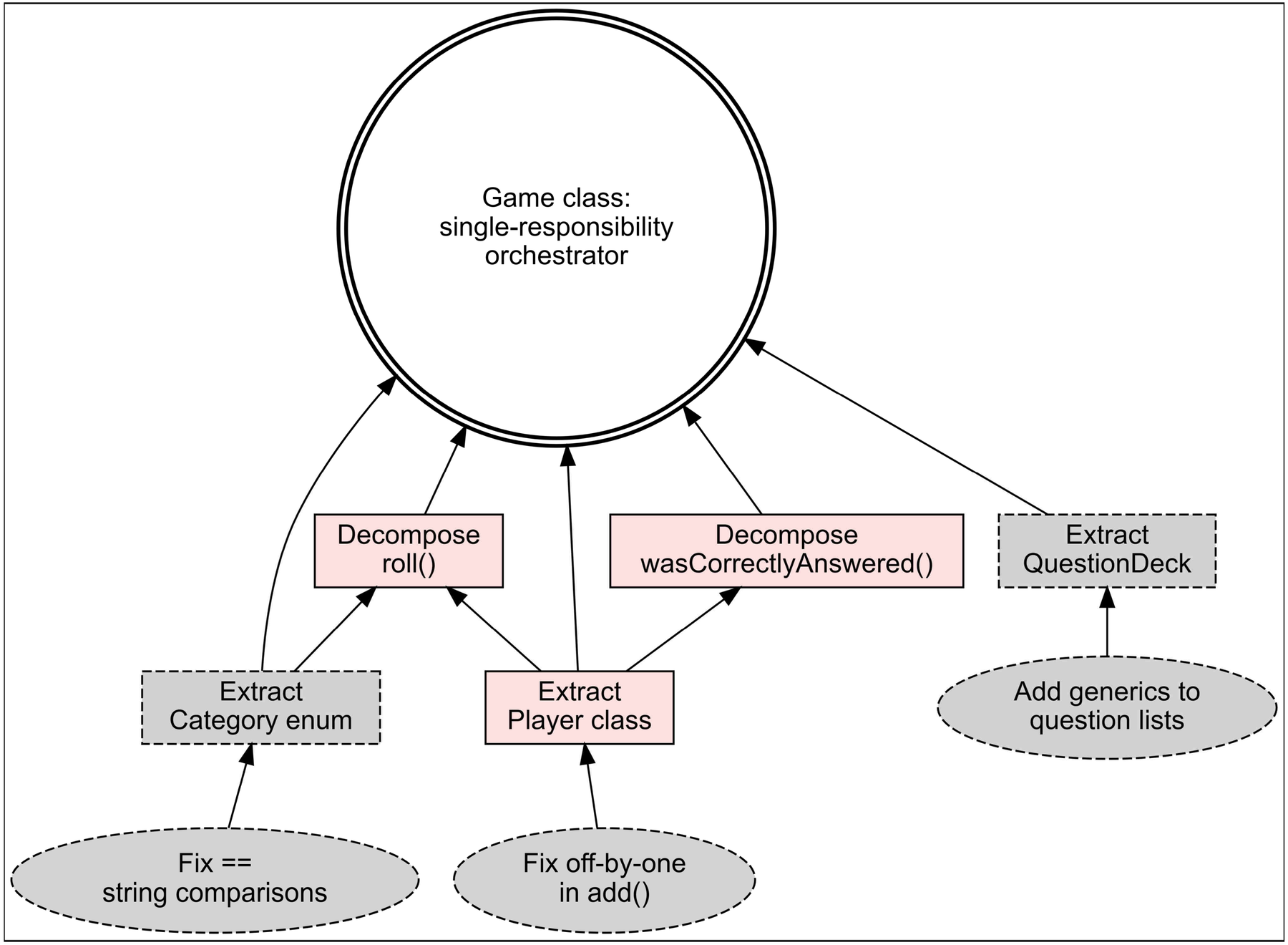

Game class:
single-responsibility
orchestrator
Decompose
roll()
Decompose
wasCorrectlyAnswered()
Extract
QuestionDeck
Extract
Category enum
Extract
Player class
Add generics to
question lists
Fix ==
string comparisons
Fix off-by-one
in add()

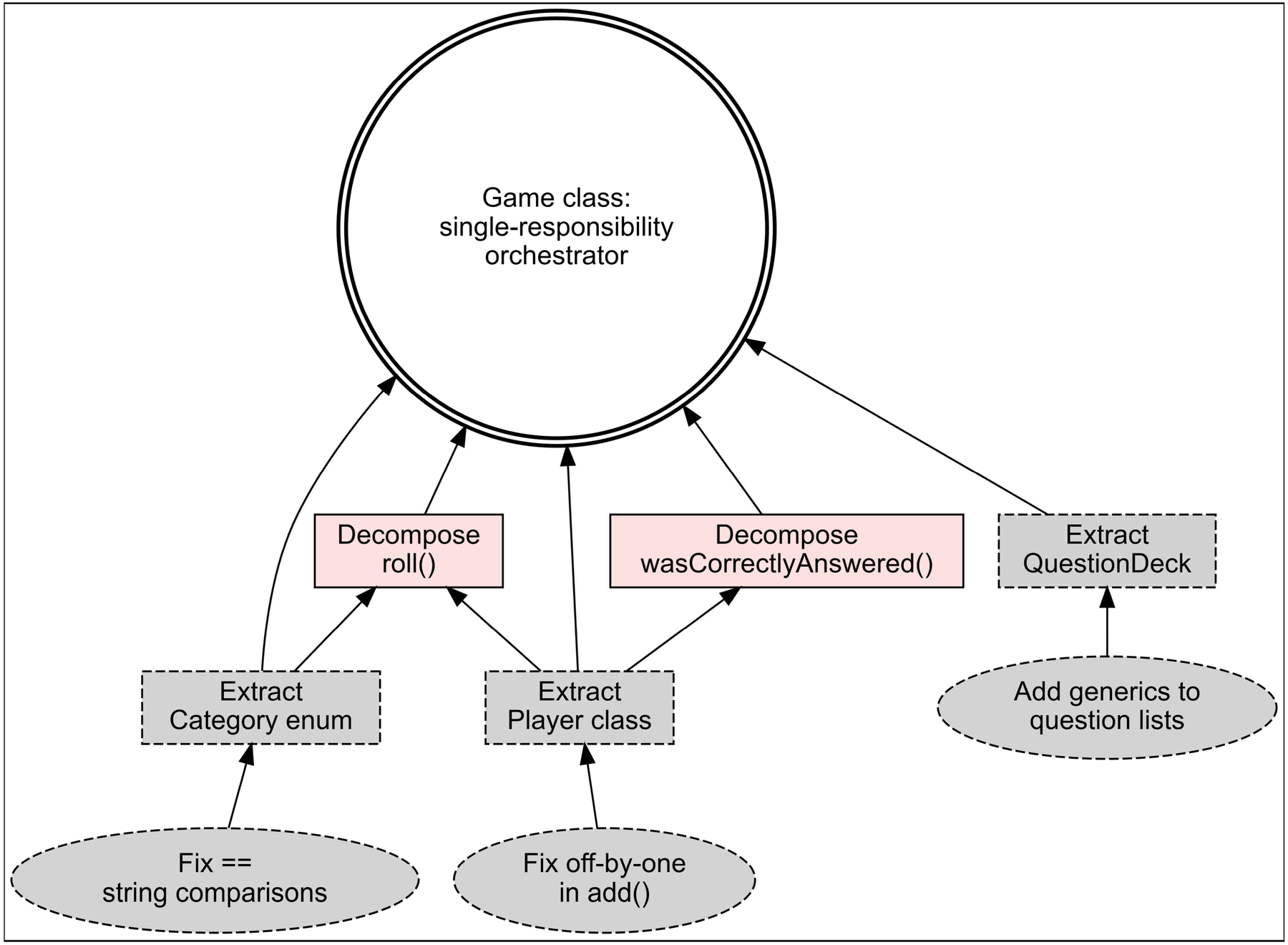

Game class:
single-responsibility
orchestrator
Decompose
roll()
Decompose
wasCorrectlyAnswered()
Extract
QuestionDeck
Extract
Category enum
Extract
Player class
Add generics to
question lists
Fix ==
string comparisons
Fix off-by-one
in add()

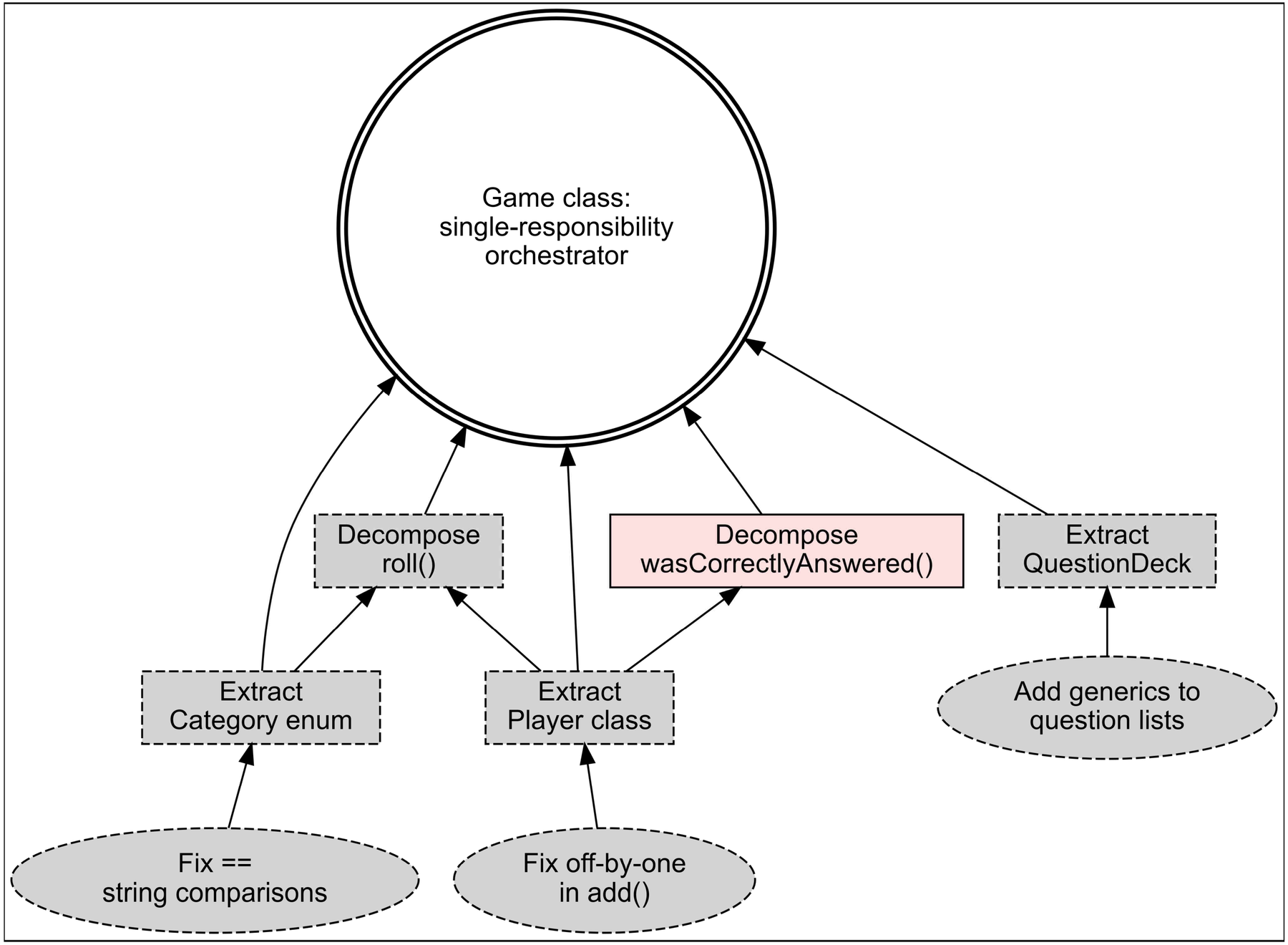

Game class:
single-responsibility
orchestrator
Decompose
roll()
Decompose
wasCorrectlyAnswered()
Extract
QuestionDeck
Extract
Category enum
Extract
Player class
Add generics to
question lists
Fix ==
string comparisons
Fix off-by-one
in add()

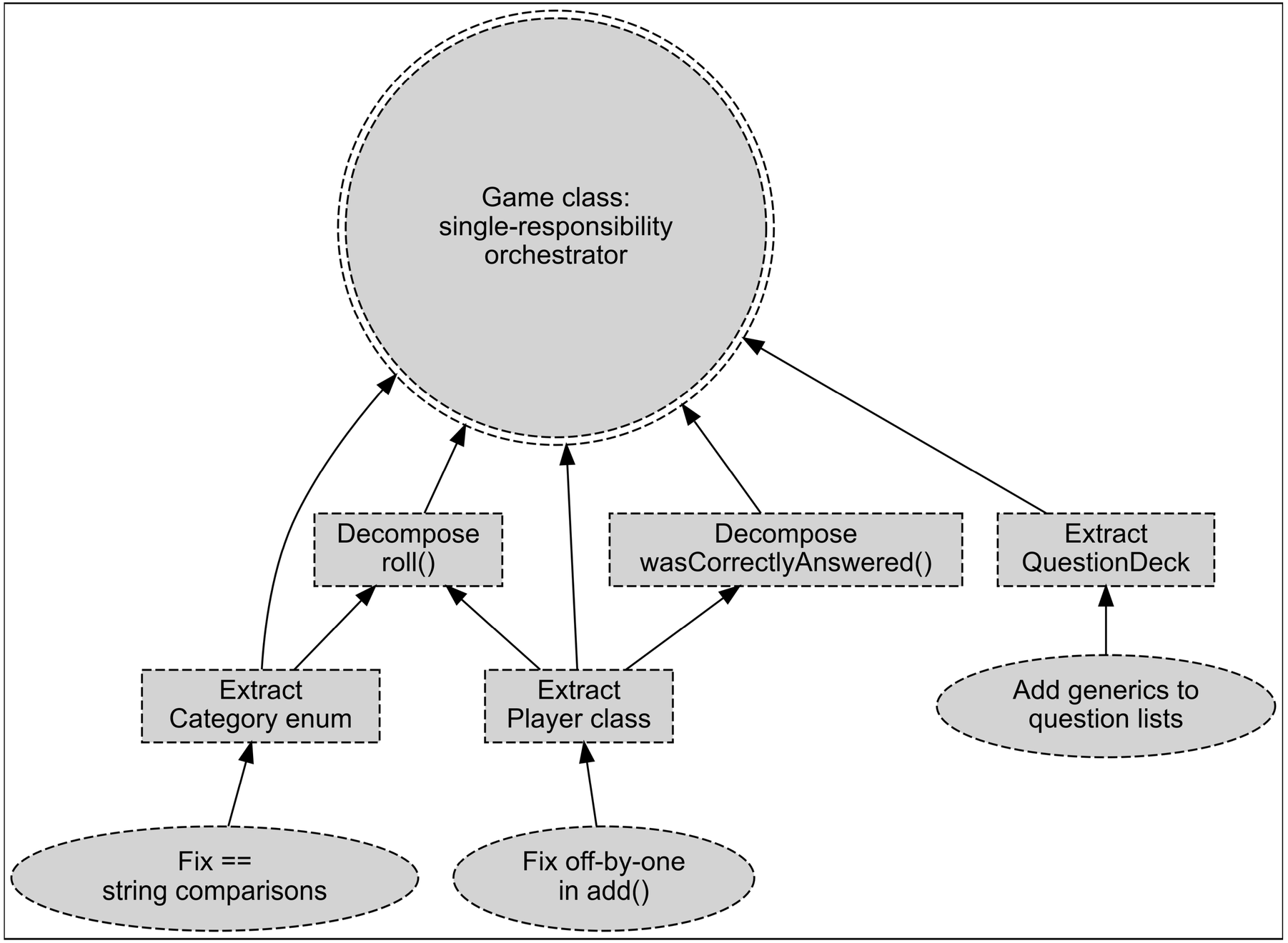

Game class:
single-responsibility
orchestrator
Decompose
roll()
Decompose
wasCorrectlyAnswered()
Extract
QuestionDeck
Extract
Category enum
Extract
Player class
Add generics to
question lists
Fix ==
string comparisons
Fix off-by-one
in add()


Game class:
single-responsibility
orchestrator
Decompose
roll()
Decompose
wasCorrectlyAnswered()
Extract
QuestionDeck
Extract
Category enum
Extract
Player class
Add generics to
question lists
Fix ==
string comparisons
Fix off-by-one
in add()


Game class:
single-responsibility
orchestrator
Decompose
roll()
Decompose
wasCorrectlyAnswered()
Extract
QuestionDeck
Extract
Category enum
Extract
Player class
Add generics to
question lists
Fix ==
string comparisons
Fix off-by-one
in add()


Game class:
single-responsibility
orchestrator
Decompose
roll()
Decompose
wasCorrectlyAnswered()
Extract
QuestionDeck
Extract
Category enum
Extract
Player class
Add generics to
question lists
Fix ==
string comparisons
Fix off-by-one
in add()

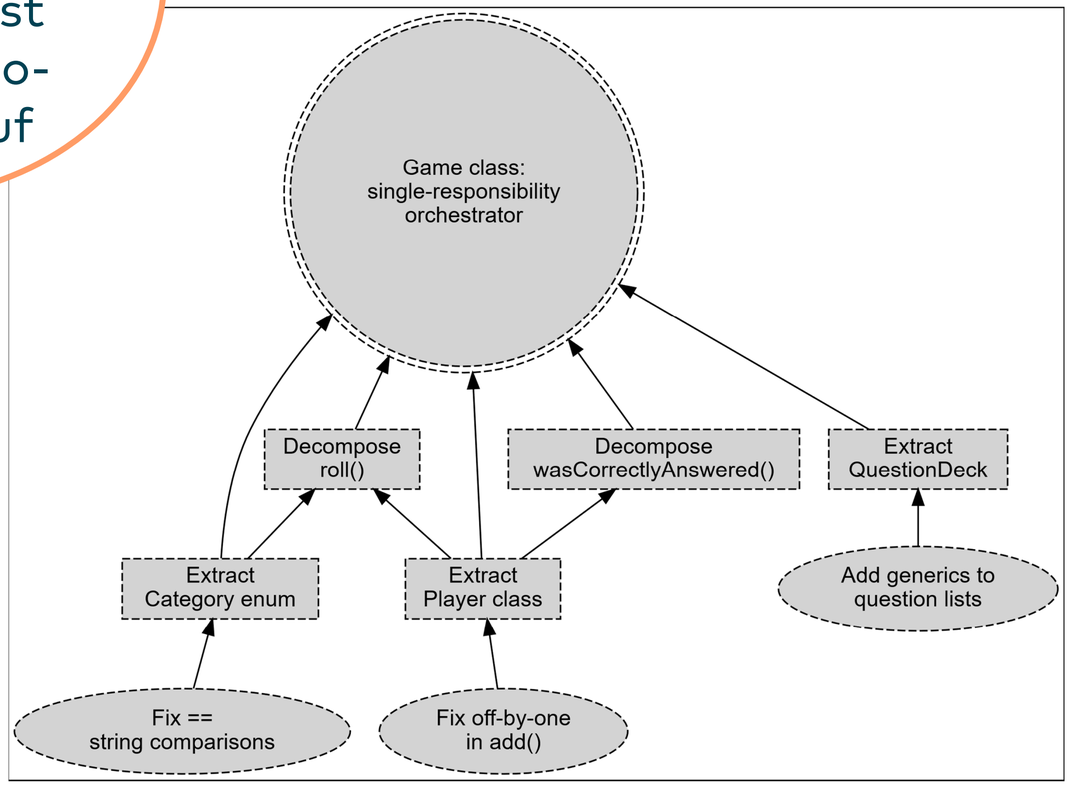

Game class:
single-responsibility
orchestrator
Decompose
roll()
Decompose
wasCorrectlyAnswered()
Extract
QuestionDeck
Extract
Category enum
Extract
Player class
Add generics to
question lists
Fix ==
string comparisons
Fix off-by-one
in add()


Game class:
single-responsibility
orchestrator
Decompose
roll()
Decompose
wasCorrectlyAnswered()
Extract
QuestionDeck
Extract
Category enum
Extract
Player class
Add generics to
question lists
Fix ==
string comparisons
Fix off-by-one
in add()

Kleines Beispiel

Zerteile diese Gottklasse. Baue dir hierfür zuerst einen Mikado-Graphen auf

(1) skill: mikado-graph



(2) hook: PostToolUse
-> run-approval-test.sh

```
#1  14:19:46  Game.java  src/main/java/com/adaptionsoft/games/uglytrivia/Game.java  8710ms  PASS
#2  14:23:11  Game.java  src/main/java/com/adaptionsoft/games/uglytrivia/Game.java  8233ms  PASS
#3  14:27:19  Game.java  src/main/java/com/adaptionsoft/games/uglytrivia/Game.java  5715ms  PASS
```

(3) subagent:
-> test analysis agent

```
#8  14:37:57  Game.java  src/main/java/com/adaptionsoft/games/uglytrivia/Game.java  BLOCKED

Agent hook condition was not met: The refactoring is incomplete and the code does not compile. The
edit removed the question list fields (popQuestions, scienceQuestions, sportsQuestions,
rockQuestions) and moved them to a QuestionDeck class, but failed to update the askQuestion()
method (lines 77-80) which still references the removed fields. The compiler errors are: cannot
find symbol variable popQuestions/scienceQuestions/sportsQuestions/rockQuestions in Game.java:77-
80. The fix requires updating askQuestion() to use questionDeck.nextQuestion(currentCategory())
instead of directly accessing the removed fields. The approval test cannot run until compilation
succeeds.
```

13 hook executions · 11 passed · 2 failed

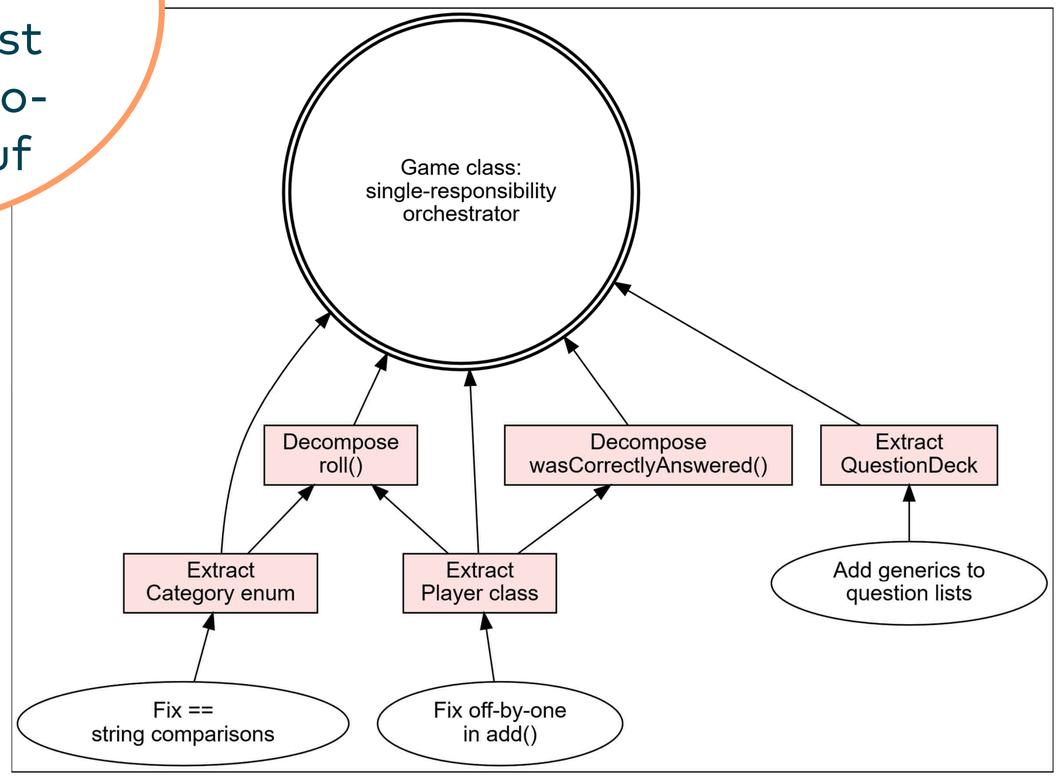
#1	14:19:46	**Game.java**	src/main/java/com/adaptionsoft/games/uglytrivia/Game.java	8710ms	PASS ▶
#2	14:23:11	**Game.java**	src/main/java/com/adaptionsoft/games/uglytrivia/Game.java	8233ms	PASS ▶
#3	14:27:19	**Game.java**	src/main/java/com/adaptionsoft/games/uglytrivia/Game.java	5715ms	PASS ▶
#4	14:30:22	**Category.java**	src/main/java/com/adaptionsoft/games/uglytrivia/Category.…	4987ms	PASS ▶
#5	14:31:37	**Game.java**	src/main/java/com/adaptionsoft/games/uglytrivia/Game.java	7847ms	PASS ▶
#6	14:36:17	**QuestionDeck.java**	src/main/java/com/adaptionsoft/games/uglytrivia/Quest…	5017ms	PASS ▶
#7	14:37:13	**Game.java**	src/main/java/com/adaptionsoft/games/uglytrivia/Game.java	5116ms	FAIL ▼

```
APPROVAL TEST: FAILED
```

Kleines Beispiel

Zerteile diese Gottklasse. Baue dir hierfür zuerst einen Mikado-Graphen auf

(1) skill: mikado-graph



(2) hook: PostToolUse
-> run-approval-test.sh

```
#1  14:19:46  Game.java  src/main/java/com/adaptionsoft/games/uglytrivia/Game.java  8710ms  PASS
#2  14:23:11  Game.java  src/main/java/com/adaptionsoft/games/uglytrivia/Game.java  8233ms  PASS
#3  14:27:19  Game.java  src/main/java/com/adaptionsoft/games/uglytrivia/Game.java  5715ms  PASS
```

(3) subagent:
-> test analysis agent

```
#8  14:37:57  Game.java  src/main/java/com/adaptionsoft/games/uglytrivia/Game.java  BLOCKED

Agent hook condition was not met: The refactoring is incomplete and the code does not compile. The
edit removed the question list fields (popQuestions, scienceQuestions, sportsQuestions,
rockQuestions) and moved them to a QuestionDeck class, but failed to update the askQuestion()
method (lines 77-80) which still references the removed fields. The compiler errors are: cannot
find symbol variable popQuestions/scienceQuestions/sportsQuestions/rockQuestions in Game.java:77-
80. The fix requires updating askQuestion() to use questionDeck.nextQuestion(currentCategory())
instead of directly accessing the removed fields. The approval test cannot run until compilation
succeeds.
```

#5 14:31:37 **Game.java** src/main/java/com/adaptionsoft/games/uglytrivia/Game.java 7847ms PASS ▶

#6 14:36:17 **QuestionDeck.java** src/main/java/com/adaptionsoft/games/uglytrivia/Quest… 5017ms PASS ▶

#7 14:37:13 **Game.java** src/main/java/com/adaptionsoft/games/uglytrivia/Game.java 5116ms FAIL ▼

APPROVAL TEST: FAILED

#8 14:37:57 **Game.java** src/main/java/com/adaptionsoft/games/uglytrivia/Game.java BLOCKED ▼

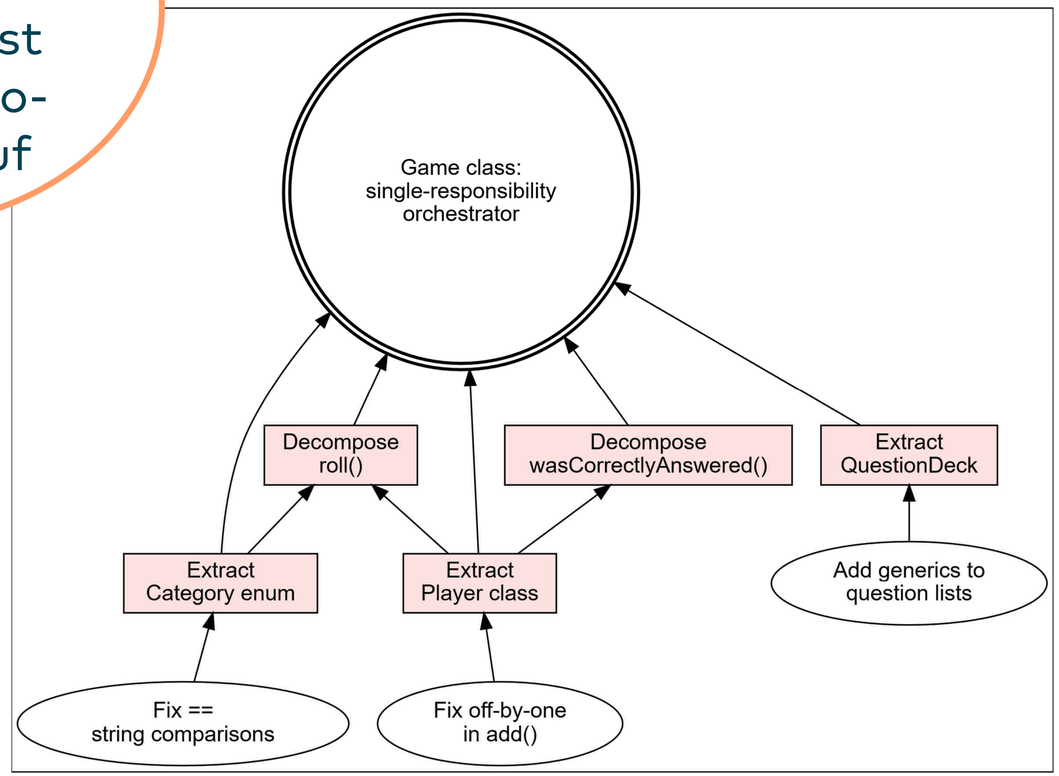
Agent hook condition was not met: The refactoring is incomplete and the code does not compile. The edit removed the question list fields (popQuestions, scienceQuestions, sportsQuestions, rockQuestions) and moved them to a QuestionDeck class, but failed to update the askQuestion() method (lines 77-80) which still references the removed fields. The compiler errors are: cannot find symbol variable popQuestions/scienceQuestions/sportsQuestions/rockQuestions in Game.java:77-80. The fix requires updating askQuestion() to use questionDeck.nextQuestion(currentCategory()) instead of directly accessing the removed fields. The approval test cannot run until compilation succeeds.

#9 14:38:13 **Game.java** src/main/java/com/adaptionsoft/games/uglytrivia/Game.java 6645ms PASS ▶

Kleines Beispiel: Mikado-Refactoring

Zerteile diese Gottklasse. Baue dir hierfür zuerst einen Mikado-Graphen auf

(1) skill: mikado-graph



(2) hook: PostToolUse
-> run-approval-test.sh

```
#1  14:19:46  Game.java  src/main/java/com/adaptionsoft/games/uglytrivia/Game.java  8710ms  PASS
#2  14:23:11  Game.java  src/main/java/com/adaptionsoft/games/uglytrivia/Game.java  8233ms  PASS
#3  14:27:19  Game.java  src/main/java/com/adaptionsoft/games/uglytrivia/Game.java  5715ms  PASS
```

(3) subagent:
-> test analysis agent

```
#8  14:37:57  Game.java  src/main/java/com/adaptionsoft/games/uglytrivia/Game.java  BLOCKED

Agent hook condition was not met: The refactoring is incomplete and the code does not compile. The
edit removed the question list fields (popQuestions, scienceQuestions, sportsQuestions,
rockQuestions) and moved them to a QuestionDeck class, but failed to update the askQuestion()
method (lines 77-80) which still references the removed fields. The compiler errors are: cannot
find symbol variable popQuestions/scienceQuestions/sportsQuestions/rockQuestions in Game.java:77-
80. The fix requires updating askQuestion() to use questionDeck.nextQuestion(currentCategory())
instead of directly accessing the removed fields. The approval test cannot run until compilation
succeeds.
```

Anhang

Agentic Software Modernization Cycle

Agentic Software Modernization Cycle

I. Kick-Off Workshop

II. Kontinuierliche Begleitung

1. Verstehen

Modernisierungsbedarf erheben und gemeinsam die Richtung festlegen.

2. Entscheiden

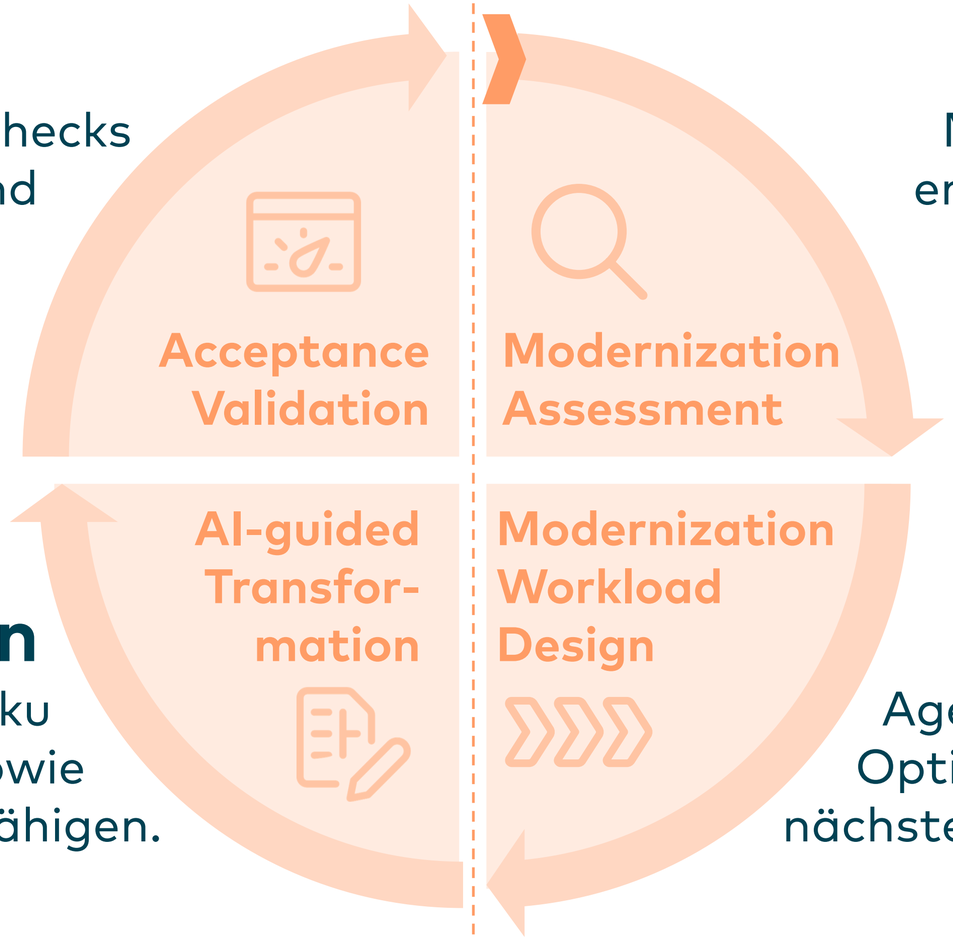
Agentische und klassische Optionen abwägen und die nächsten Schritte priorisieren.

3. Transformieren

Code, Architektur und Doku KI-gestützt verbessern sowie das Team mit Tooling befähigen.

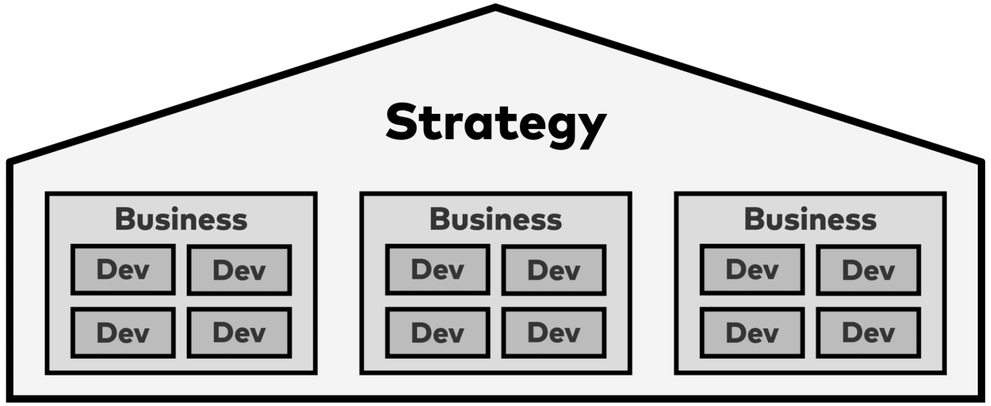
4. Validieren

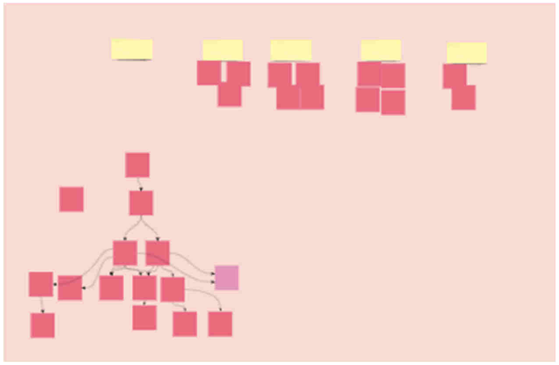
Fortschritt durch Tests, Checks sowie Berichte messen und Learnings für die nächste Iteration aufbereiten.

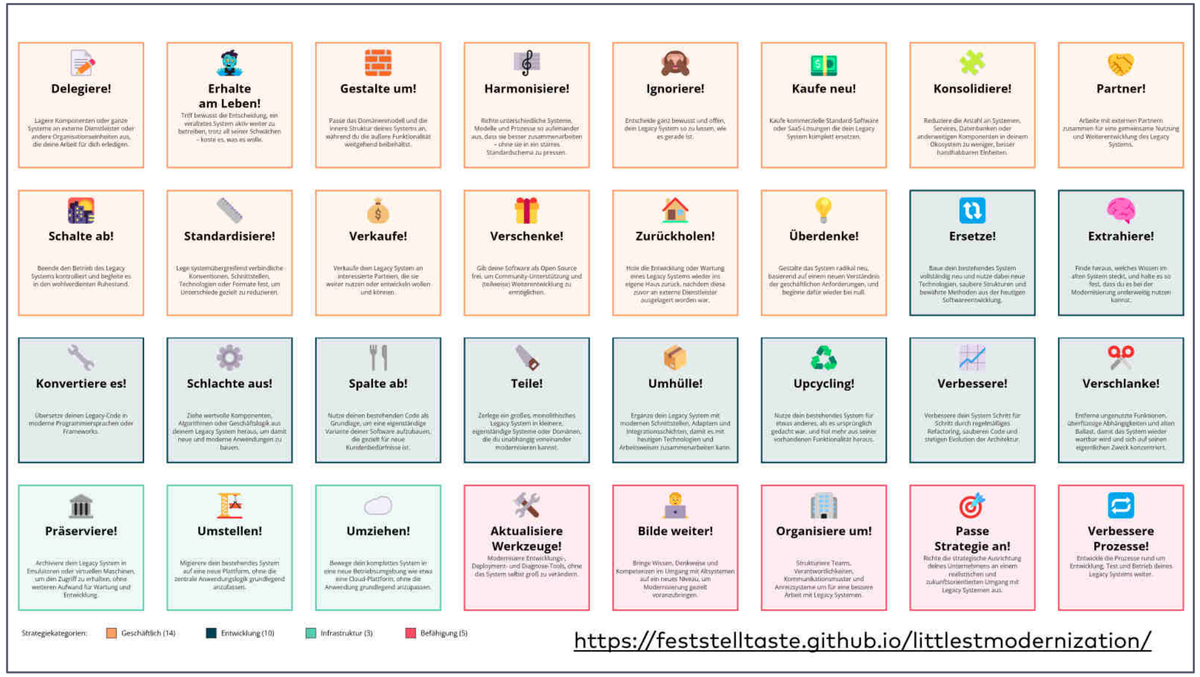


Agentic Software Modernization Cycle

Verstehen: Problemdefinition und Modernisierungsansätze







Agentic Software Modernization Cycle

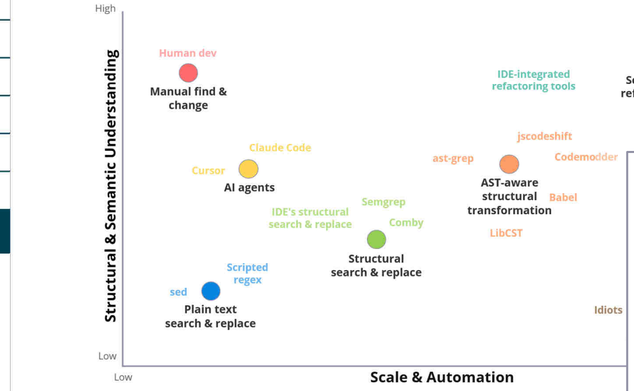
Entscheiden: Workload-Design und Modernization Cases

AI for Legacy Modernization: When and How to Use (or not)

	AI agents	AI assistants	Guided AI	Transformation Tools	Manual Work
General Idea	Autonomous systems orchestrate analysis, transformation and validation of modernization workflows	Human-guided AI-based task execution for fixing code in smaller areas / clearly scoped contexts	Human-led detection of issues or anti-patterns, followed by localized AI-generated fixes within defined areas	Human-based creation of formal rules and recipes to perform consistent, automated code transformations	Developers manually analyze, reason about, and fix issues (based on deep domain and system knowledge)
Possible Use Cases	Cleanup ideation, multistep refactoring planning, smaller bug fixing across code bases	Summarizing code, generating tests & comments, renaming identifiers, writing code snippets	Identifying systemic issues and using AI to propose or apply localized solutions	Framework migrations, API upgrades, bulk renames,	Special issues like redesign of critical parts of business logic or
Control How much humans can be in the loop	--	-	o		
Risk How likely changes go wrong	++	+	-		
Breadth How wide the method can operate	++	o	-		
Accuracy How well problematic spots are addressed	o	o	+		
Traceability How well actions can be tracked	-	o	++		
Efforts How much work setup and use need	o	o	o		
Volume How much can be processed	+	-	o		

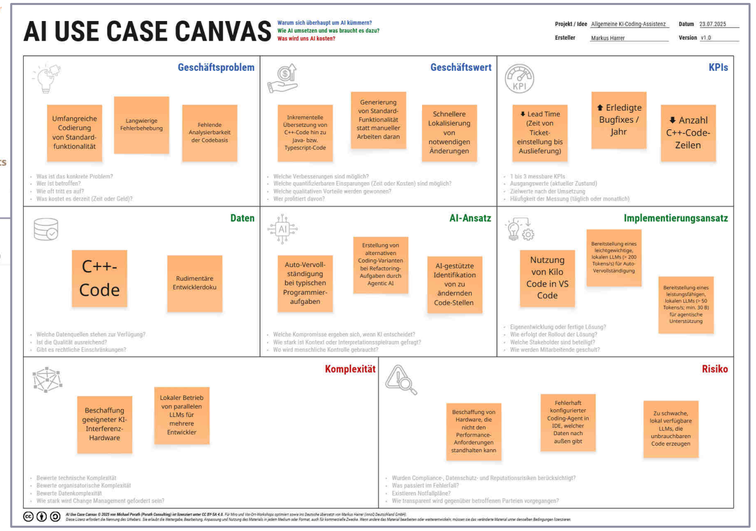
INNOQ

The Landscape of Code Transformation Tools & Methods



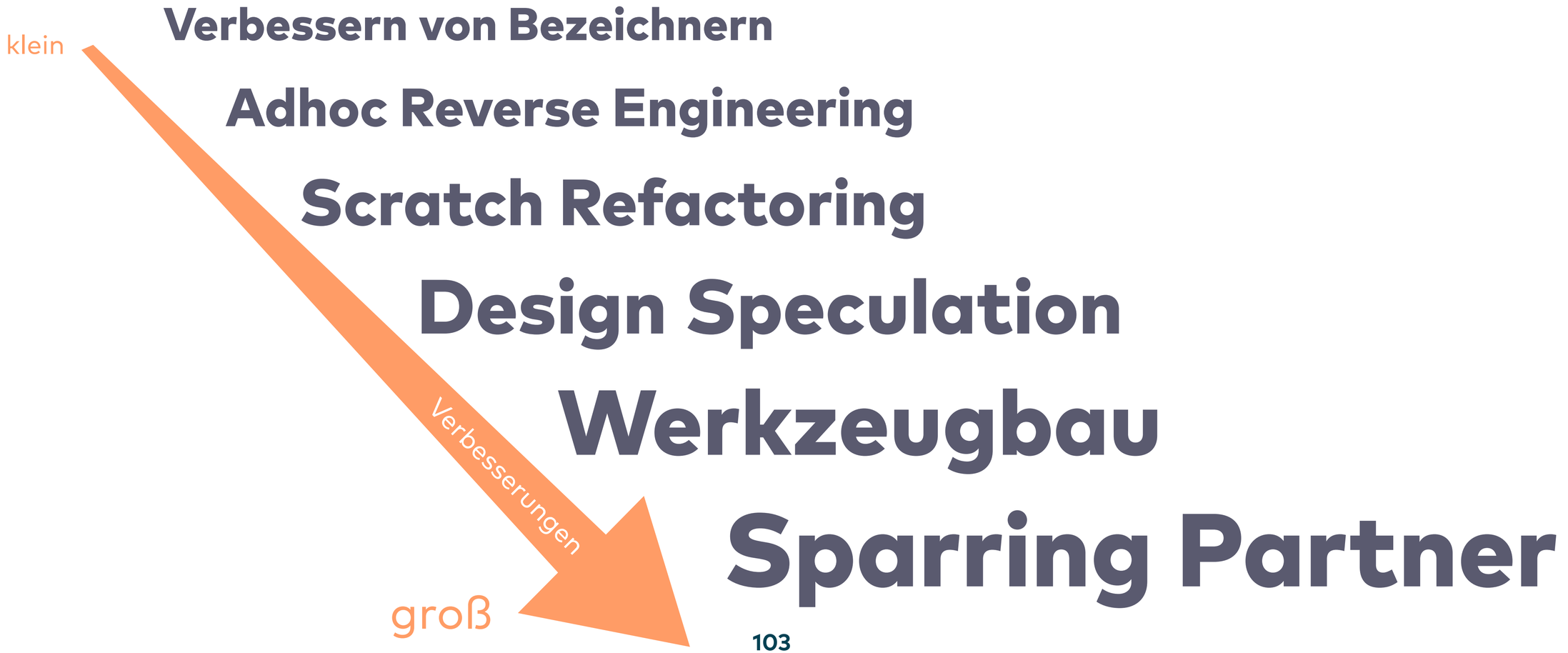
Made by Markus Harrer (markusharrer.de), licensed under Creative Commons - Attribution-ShareAlike 4.0 International (CC BY-SA 4.0)

AI USE CASE CANVAS



Agentic Software Modernization Cycle

Transformieren: je nach Use Case umsetzen, z. B.

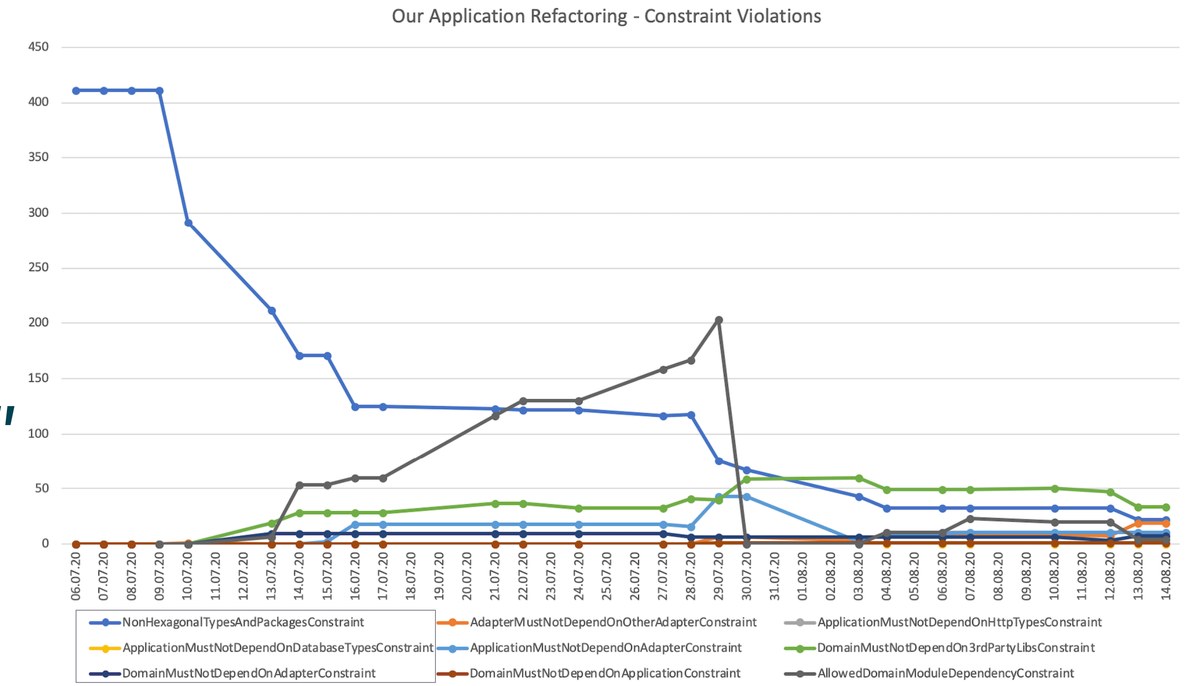


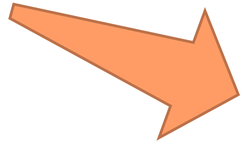
Agentic Software Modernization Cycle

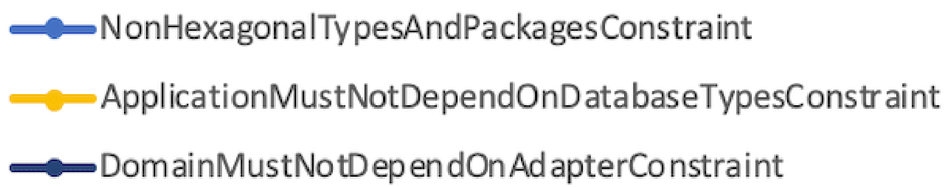
Validieren: Umbauarbeiten über die Zeit hinweg verfolgen

Beispiel:

Architektur-Refactoring von “No Architecture” zu “Hexagonal Architecture”







https://www.innoq.com/en/blog/visualizing-progress-of-refactoring-into-hexagonal-architecture-using-jqassistant