PROFESSIONEEL SOFTWARE ONTWIKKELEN MET AI

Hoe je AI bij de les houdt.

Rob Vugts — AI-chitect

April 2026

EVEN VOORSTELLEN

30+ Jaar

Technical Architect & Software Engineer

Banking

ING, Credit Suisse, UBS

AI-chitect

Freelance consultancy — AI-assisted development

14 Maanden

Fulltime onderzoek: kan AI enterprise-grade code leveren?

Missie

Ontwikkelaars en organisaties helpen AI professioneel in te zetten

GEBOUWD MET AI — IN 1-2 WEKEN

SchouwMaat.nl

Offerte-tool voor schildersbedrijven

React + Vite + Supabase

VervroegdPensioenPlanner.nl

Financiële planning tool voor vervroegd pensioen

Javascript, Node.js, Azure

HomeTicz iPhone App

Native mobiele applicatie voor home automation met Domoticz

Swift, iOS

Evolutie van dezelfde technieken die ik vandaag laat zien

AGENDA

0:10	**Opening & Context**	Introductie, de belofte & het gevaar, AI als accelerator
0:20	**De Volwassenheidsladder**	Agent Rules, Gespecialiseerde Agents, Meta-Prompting
0:20	**Golden Trinity & AaC**	SDD, TDD, Architecture-as-Code
0:45	**Live Demo**	Van spec naar werkende code in VSCode + Copilot
0:15	**Enterprise & Breder**	Borging, governance, AI voorbij code
0:20	**Afsluiting**	Skills, next steps, Q&A

Onderbreek me gerust — dit is een kennissessie, geen college!

DE BELOFTE EN DE KATER

De Magie

- Je typt een prompt, code stroomt op het scherm
- Functies, tests, features — het werkt gewoon
- Je voelt je als een 10x developer
- Vibe Coding: "geef je over aan de vibes"

De 1-Uur Kater

- AI hallucineert niet-bestaande libraries of gebruikt oude versies
- Vergeet code van 6 prompts geleden
- Subtiele bugs in productie
- Semantic drift
- 2 uur debuggen voor 20 minuten "magie"

Nog erger in een grote organisatie:
Meerdere developers – verouderde systemen – compliance – security – verouderde documentatie.

AI is een Accelerator.

Pas op wat je versnelt!

“AI lost slechte datakwaliteit niet op. Het industrialiseert het.”

“Een zelfverzekerd fout antwoord is gevaarlijker dan een onzeker juist antwoord.”

“Validatie is geen redactie. Het is onderzoek.”

“De eerste demo is makkelijk. De 100e betrouwbare output is het werk.”

VIJF PRINCIPES VOOR HET VERSNELLEN VAN DE JUISTE DINGEN

1. **Begin met de data, niet het model**
   Als je je data niet vertrouwt, ben je niet klaar voor AI

2. **Ontwerp voor onzekerheid**
   Systemen die erkennen wat ze niet weten zijn betrouwbaarder

3. **Technologie volgt het probleem**
   "We willen AI deployen" is geen use case

4. **Eigen de output**
   AI is een tool. Iemand tekent onder het resultaat

5. **Investeer in de 100e output**
   De eerste demo is makkelijk. Betrouwbaarheid is het werk

DE VOLWASSENHEIDSLADDER: VAN PRUTSEN NAAR SUCCES!

De Paradox: Toenemende Beperking = Toenemende Betrouwbaarheid

Je kunt op elke trede stoppen en direct waarde krijgen.
Maar als je doorbouwt, versterkt het systeem zichzelf.

LEVEL 1

AGENT RULES — FORCEER JOUW GRONDWET

copilot-instructions.md

De "grondwet" van je project — Copilot leest dit bij elke interactie mee

.github/instructions/*.md

Taalspecifieke regels: Python, SQL, Terraform — activeren automatisch

.github/skills/*.md

Herbruikbare skills: /create-spec,/create-tasks, /generate-prompt, /refactor-python, /audit-security

Starter template beschikbaar!

github.com/rvugts/copilot-dev-starter

Voorbeeld:

- Altijd PEP 8 volgen
- Parameterized SQL queries — altijd
- Nooit eval() of exec() gebruiken
- dbt modellen: naamconventie stg_, int_, fct_
- PySpark: gebruik .select() niet .withColumn()
- Type hints verplicht op alle functie"

Investering: 1 uur | Impact: onmiddellijk

LEVEL 2

SPECIALIZED AGENTS — EEN TEAM VAN EXPERTS

/refactor-python	DRY principes, functielengte, security patterns
/audit-security	OWASP Top 10, SQL injection, code injection
/create-spec	Interactief specificatie genereren (of verbeteren) vanuit interview
/create-tasks	Spec opsplitsen in atomaire, uitvoerbare taken
/generate-prompt	Meta-prompting: AI interview -> de perfecte prompt genereren

"Een LLM is een huisarts. Vraag een huisarts niet om hartchirurgie."

LEVEL 3

META-PROMPTING & CONTEXT ENGINEERING

Meta-Prompting

AI schrijft de perfecte prompt voor AI

/generate-prompt → AI interviewt jou

Edge cases, constraints, error states

Resultaat: zeer gestructureerde prompt

/run-prompt → perfecte executie

Context Engineering

Elke token kost geld en aandacht

Surgical Context: Beperk tokens, verhoog precisie

- Alleen relevante bestanden
- Need-to-know basis voor de AI
- Juiste model voor de juiste taak
- Fresh Start bij elk breekpunt
- Meta-prompting helpt!

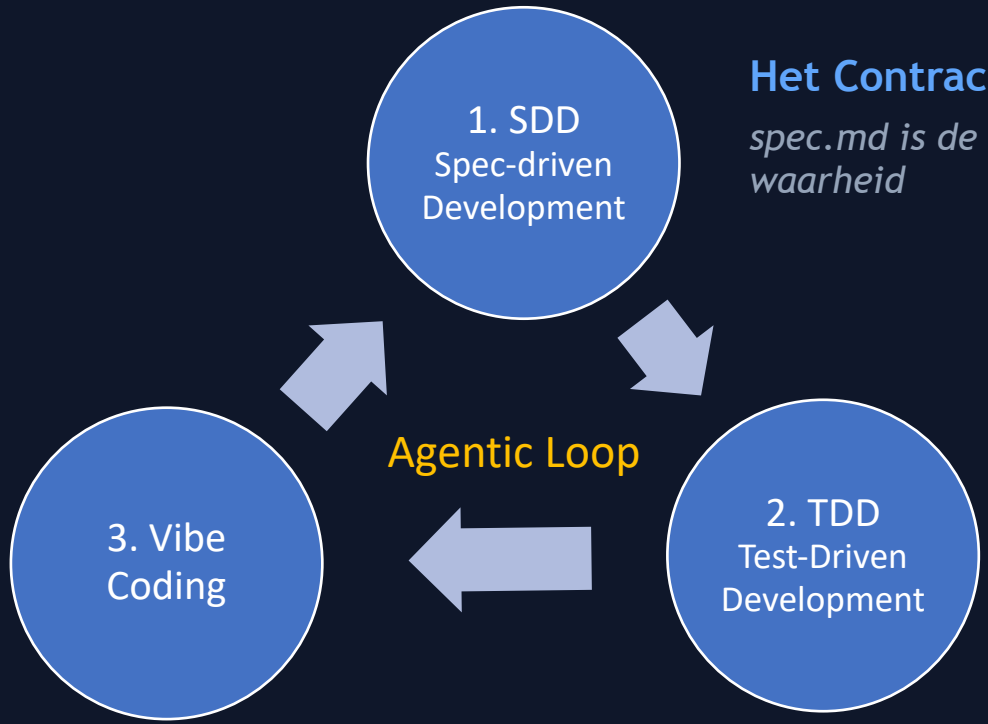
Kernvaardigheid: goede prompts schrijven — AI wil je pleasen!

PAUZE

10 minuten — vragen en discussie welkom!

LEVEL 4

DE GOLDEN TRINITY - DE KWALITEITS-WORKFLOW



Het Contract

spec.md is de enige bron van waarheid

De Executie

AI schrijft code -> Tests "Groen"

De Vangrails

Tests vóór de code -> Tests "Rood"

"AI die eerst code en dan tests schrijft, kijkt zijn eigen huiswerk na"

DE GOLDEN TRINITY IN DE PRAKTIJK

1 Plan buiten je IDE

Gebruik Gemini/ChatGPT/ Claude voor architectuur & spec

2 Maak spec.md

Contract: doel, constraints, interfaces, edge cases, ADRs, NFRs

3 Genereer tests

Tests vanuit de spec, niet de implementatie

4 Tests moeten FALEN

Rode terminal = groenlicht om door te gaan

5 Vibe Code!

AI implementeert, tests slagen, commit on green

6 Refactor & Review

DRY, type hints, security audit, commit weer

De Nucleaire Optie

Als de AI in een loop zit:
STOP → Revert naar laatste groene commit → Fresh start → Nieuwe aanpak

LEVEL 5

ARCHITECTURE-AS-CODE HIËRARCHIE

Blueprint The “Why”	Visie, filosofie, design principes
Architecture Master Plan The “What”	Tech stack, C4 modellen, ADRs, NFRs
Execution Playbook The “How”	Readiness gates, fase plan, AI protocol (SDD + TDD), documentatie
Non-Goals The “What Not”	Wat AI expliciet NIET mag doen
Domain Specs The “Contract”	Glossary, invariants, schema, API contracts
Fitness Functions The “No Cheating!”	Geautomatiseerde architectuurtests
Epics & Stories & Tasks The “Work”	Atomaire werkeenheden voor AI

Hogere lagen dienen als input voor de volgende lagen

▼ input

▼ input

▼ input

▼ input

▼ input

▼ input

Elk level beperkt de AI — schendingen worden automatisch geblokkeerd

WAT ZIT ER IN? BLUEPRINT & ARCHITECTURE MASTER PLAN

Blueprint — Het WAAROM

Core Problem Statement
Welk fundamenteel probleem lossen we op?

Governing Laws
Ononderhandelbare principes die ALLES sturen

Design Principles
Hoe moet het systeem aanvoelen?

Zorgverzekeraar voorbeeld:
"Wij verwerken gevoelige gezondheidsdata.
EU-soevereiniteit is niet-onderhandelbaar.
Elke architectuurbeslissing respecteert dit."

Master Plan — Het WAT

Technology Stack
Elke keuze linkt naar een ADR met rationale

C4 Architecture Modellen
Context → Container → Component diagrammen

Non-Functional Requirements
Meetbare targets: latency < 200ms, uptime 99.9%

Security Architecture
Threat model, layered controls, RLS policies

ADR Summary
Alle beslissingen geïndexeerd met status

De Blueprint verandert zelden. Het is jullie grondwet.
Het Architecture Master Plan maakt die grondwet technisch uitvoerbaar.

NON-GOALS & DOMAIN SPECS — DE GRENZEN

Non-Goals — Wat AI NIET mag doen

AI is behulpzaam van nature — het stelt altijd extra features voor. Non-Goals zijn permanente uitsluitingen.

NG-001	Geen directe toegang tot patiëntdossiers vanuit AI	FORBIDDEN
NG-015	Geen auto-goedkeuring van claims zonder menselijke review	FORBIDDEN
NG-035	Geen predictieve modellen op individu zonder ethische review	FORBIDDEN

Domain Specs — De Bron van Waarheid

glossary.md	Elk domeinbegrip: wat het IS, wat het NIET is, of AI het mag beïnvloeden
invariants.md	Onschendbare bedrijfsregels (volgende slide!)
schema.sql	Bevroren database schema — geen placeholders, definitief
states.md	State machines als tabellen, niet als proza

Non-Goals zijn geen suggesties — het zijn muren.
Domain Specs zijn de bron waar fitness functions tegen valideren.

INVARIANTEN — DE ONSCHENDBARE REGELS

Een bedrijfsregel die ALTIJD moet gelden — ongeacht wie de code schrijft (mens of AI).

#	Invariant	Wat het afdwingt
INV-1	**Tenant Isolatie**	Elke query filtert op verzekeraar_id — geen data-lekkage
INV-2	**Audit Trail**	Elke mutatie op claimdata wordt gelogd: wie, wanneer, wat
INV-3	**Geen Floats voor Geld**	Financiële berekeningen: Decimal, nooit float
INV-4	**Claim Status Machine**	Afgewezen claim kan NIET terug naar "in behandeling"
INV-5	**BSN Encryptie**	BSN-nummers nooit plaintext opgeslagen
INV-6	**Vier-Ogen Principe**	Claims boven €10.000 vereisen twee goedkeurders

Elke invariant wordt een fitness function. Schendt code een invariant? → Pipeline blokkeert.
Mens of AI maakt niet uit.

FITNESS FUNCTIONS — DE GUARDRAILS DIE NOOIT SLAPEN

Unit tests vragen:
"Werkt deze feature?"

Fitness functions vragen:
"Hoort deze code hier?"

Ze draaien automatisch op elke commit.
Ze vergeten nooit.
Ze worden nooit moe.

Concrete voorbeelden (pytest):

```
# INV-1: Tenant Isolatie
def test_tenant_isolation():
    for model in get_all_models():
        assert "verzekeraar_id" in model.columns

# INV-3: Geen floats voor geld
def test_no_floats_in_finance():
    for f in Path("src/claims").rglob("*.py"):
        assert "float" not in f.read_text()

# Layer Boundaries (PyTestArch)
def test_api_no_worker_imports():
    Rule().modules_that()
      .are_sub_modules_of("src.api")
      .should_not().import_modules_that()
      .are_sub_modules_of("src.workers")
```

Het immuunsysteem van je architectuur. Het detecteert vreemde patronen en blokkeert ze vóór ze schade aanrichten.

DE CI/CD PIPELINE — CONTINUE HANDHAVING

Developer commit

▼

Pre-commit Hooks

< 5 seconden

Linting (ruff, black) • Type checking (mypy) • Basis architectuurcheck • Security linting (bandit)

▼

CI Pipeline

~ 2 minuten

ALLE fitness functions • Spec compliance • Schema drift • ADR compliance • Arch. report als PR comment

▼

Deploy Gate

Blokkeert of doorlaat

BLOCKING fitness function faalt? → Deploy geblokkeerd. Geen uitzonderingen. Mens of AI — maakt niet uit.

Fitness functions zijn gewoon pytest tests in jullie bestaande pipeline.

"DIT IS TOCH SUPERVEEL WERK?"

Nee!

Sessie 1	Jij + AI → Blueprint
Sessie 2	Jij + AI → Master Plan
Sessie 3	Jij + AI → Non-Goals + Domain Specs
Sessie 4	Jij + AI → Fitness Functions
Sessie 5	Jij + AI → Agent Rules + CI/CD

Voorbeeld prompt:

```
"Hier is mijn Blueprint.

Genereer het Master Plan:
• Technology stack tabel
• C4 container diagram
• Initiële ADRs
• NFR targets
• Security architecture"
```

Weken werk wordt dagen.

Jij bent nog steeds de architect.
AI versnelt je denken — het vervangt het niet.

AI bouwt de guardrails die AI in toom houden. Meta-level vibe coding.

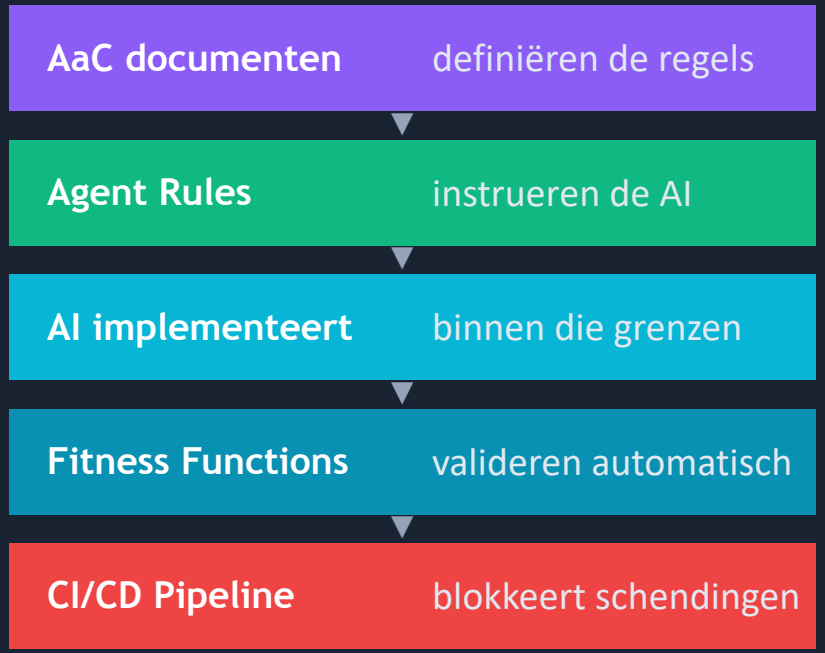
DE CIRKEL IS GESLOTEN -> EXECUTABLE ARCHITECTURE!

copilot-instructions.md:

```
# Agent Protocol
- Je MAG GEEN specs wijzigen
- Je MAG GEEN velden of states
  verzinnen
- Je MAG GEEN invarianten
  verzwakken
- Je MAG GEEN non-goals
  implementeren
- Bij onduidelijkheid: STOP.
  Gok niet.

# Bronnen van waarheid
- Schema: @docs/schema.sql
- Invarianten: @docs/invariants.md
- Non-Goals: @docs/non_goals.md
```

De complete cirkel:



↺terug naar boven

Niemand ontsnapt. Architectuur definieert. Rules instrueren. Fitness functions handhaven. CI/CD blokkeert.

ALLES BIJELKAAR - DE GEVANGENIS — DRIE MUREN

We geven de AI supermenselijke snelheid — binnen muren waar het niet uit kan

Buitenste muur: Automated Enforcement

CI/CD pipelines en Fitness Functions blokkeren alles wat door de eerste twee muren glipt.

Middelste muur: Prompt Discipline

De menselijke architect stuurt elke sessie. "Surgical context" wordt bewaakt.

Binnenste muur: Agent Rules

Specs zijn onveranderlijk • Invariants zijn absoluut • TDD is verplicht • Non-goals zijn verboden

AI bouwt zélf de guardrails (rules, CI/CD, fitness functions) die de AI in toom houden.

LIVE DEMO

Van Spec naar Werkende Code

VSCode + GitHub Copilot | Python + SQL

1. SDD: Spec genereren/verfijnen met meta-prompting
2. TDD: Tests genereren (Red)
3. Vibe Coding: Razendsnelle implementatie (Green)
4. Refactor: Consistentie, best practices, docstrings
5. Security Audit: Check op kwetsbaarheden
6. Paranoïde review: Check completeness

PAUZE

10 minuten

AI IN DE ENTERPRISE: BORGING & GOVERNANCE

Onderhoudbare code

Begrijp wat je commit. IMPLEMENTATION_STATUS.md als levend logboek. CI/CD pipeline om documentatie up-to-date te houden.

Peer reviews

Nieuwe checklist: controleer hallucinaties, verifieer tegen spec, test onafhankelijk. Paranoid Review Loop.

Audits & traceerbaarheid

Git history als audit trail. ADRs voor alle beslissingen. Privacy Mode voor gevoelige data.

Compliance (Zorgverzekeraar)

Patiëntdata vereist extra zorgvuldigheid. Teams tier voor data governance. Geen code naar AI-training.

PULL REQUESTS: PEER REVIEWS OP AI-CODE

Het probleem — en de oplossing

Het oude model werkt niet meer

- AI genereert duizenden regels in minuten
- Reviewer heeft de code niet zien ontstaan
- AI-code ziet er correct uit maar kan subtiele fouten bevatten
- Hallucinaties zijn het gevaarlijkst als ze er professioneel uitzien

95% accuraat klinkt goed — maar die 5% maakt je productie kapot

De oplossing: gelaagde defensie

1. Kleine commits via /create-tasks
2. CI/CD vangt alles wat objectief te checken is
3. Paranoïde self-review in de IDE
4. AI review op de PR
5. **Mens reviewt alleen wat er toe doet**

Elke laag filtert. De menselijke reviewer krijgt een schone, pre-geanalyseerde PR.

STAP 1-2: KLEINE COMMITS & PARANOIDE SELF-REVIEW

Kleine, atomaire commits

`/create-tasks`
splitst spec in atomaire taken

1 taak = 1 commit = 100-250 regels

AI genereert gedetailleerde commit messages:
WAT is veranderd + WAAROM

Reviewer loopt commit-voor-commit door PR

Paranoïde Self-Review in IDE

Vóór de laatste commit:
`/generate-prompt → Doe een paranoïde review`

✓ Spec volledig gehonoreerd?
✓ Test coverage >80%?
✓ Alle taken uit tasks.md afgevinkt?
✓ Strikt TDD toegepast?
✓ Geen features buiten de spec?

Output = onderdeel van de commit

Zonder structuur:
1 megacommit “Implement claim validator” — 1500 regels, niemand wil dit reviewen

Met /create-tasks:
6 commits van 100-250 regels, elk met context — review in 5-10 min per commit

STAP 3: AI REVIEW OP DE PR

Copilot als automatische senior reviewer

Review prompt (domein-specifiek):

```
Review this PR as a senior
engineer in a healthcare
insurance company.

Be critical. Focus on:
- data correctness
- edge cases
- financial impact
- hidden risks
- compliance with spec.md
```

Volledig geautomatiseerd

GitHub Actions trigger bij elke PR

Copilot / LLM analyseert de diff

Output als PR commentaar

Optioneel: AI vs AI uitdaging
"Wat miste de reviewer?"

Reviewer ziet AI-analyse voordat die zelf begint

De AI vindt 80% van de mechanische issues.
De mens focust op de 20% die oordeelsvermogen vereist.

STAP 4: MENSELIJKE REVIEW — ALLEEN WAT ER TOE DOET

AI afgehandeld:

✓ ~~Style & formatting~~	← *Linting (ruff, black)*
✓ ~~Type errors~~	← *mypy*
✓ ~~Ssecurity patterns~~	← *bandit*
✓ ~~Architectuur-schendingen~~	← *Fitness functions*
✓ ~~Spec compliance~~	← *Paranoide review + Copilot review*
✓ ~~Test coverage~~	← *CI coverage check*

De mens reviewt:

Business logica
Klopt de interpretatie van de business rules?

Data correctheid
Juiste velden, joins, race conditions?

Risico's & impact
Wat als dit faalt in productie?

"Ruikt dit goed?"
Ervaring die niet te automatiseren is

Resultaat: in plaats van 200 comments over styling heb je 3-5 comments over business logica die er écht toe doen.

DE REVIEW PIPELINE — COMPLEET OVERZICHT

Wat elke laag vangt:

Kleine commits	Overzichtelijke, reviewbare eenheden met context
Paranoid Review	Spec compliance, coverage, task completeness, TDD discipline
Automated Checks	Linting, types, security, fitness functions, coverage ≥80%
AI Review	Data correctheid, edge cases, financiële impact, verborgen risico's
Human Review	Business logica, domeinkennis, risico-inschatting, "ruikt dit goed?"

Alles is traceerbaar: paranoid review in commit, AI review als PR comment, CI logs, human approval. Auditors kunnen de hele keten volgen.

AI VOORBIJ CODE — BREDER INZETTEN

Deep Research

NotebookLM: upload documenten, genereer inzichten, verbanden, slides, videos, podcasts

Meeting Intelligence

Transcriptie → blindspot-analyse, SWOT, verborgen inzichten, actiepunten

Strategie & Voorstellen

Structureren, valideren, research, kritiek van andere AI-agents, meerdere verfijningsrondes

Persoonlijke Ontwikkeling

Experimenteer thuis! AI als tutor, bouw je eigen projecten, blijf voorop

VAN METSELAAR NAAR ARCHITECT

Vroeger

- Syntax kennen
- Libraries memoriseren
- Code typen in een bestand
- Boilerplate schrijven
- Tests "later" schrijven
- Documentatie "later" schrijven

Nu

- Systemen ontwerpen
- Constraints definiëren
- AI-agents orkestreren
- Kwaliteit waarborgen
- Strategisch denken
- Werkvreugde!

Developers met AI vervangen developers zonder AI!

BEGIN MAANDAG

Skills om te ontwikkelen

- Prompt + context engineering (Meta-prompting!)
- Architectureel denken (Architecture first!)
- Spec writing: requirements helder formuleren
- Kritisch reviewen: nooit blindelings accepteren
- Gedeelde governance in het team

Tools om mee te starten

- GitHub Copilot in VSCode (hebben jullie!)
- copilot-dev-starter template (klaar om te gebruiken)
- NotebookLM (gratis) voor deep research
- Claude / Gemini / ChatGPT / NotebookLM voor brainstormen en uitwerken van specs

Gebruik de copilot-dev-starter template — alle rules, skills en templates klaar voor gebruik!

BEDANKT!

Resources

Copilot Dev Starter	github.com/rvugts/copilot-dev-starter
Vibe Coding Playbook	Binnenkort beschikbaar op aanvraag
Cursor Resources	github.com/rvugts/cursor-resources
Mijn Website	aichitect.eu
LinkedIn Artikel	aichitect.eu/blog/ai-is-an-accelerator.html

Rob Vugts | info@aichitect.eu | +31 6 46432755

"Humans define meaning. Specs define truth. Tests define correctness. Fitness functions define enforcement. AI fills in the mechanical gaps."