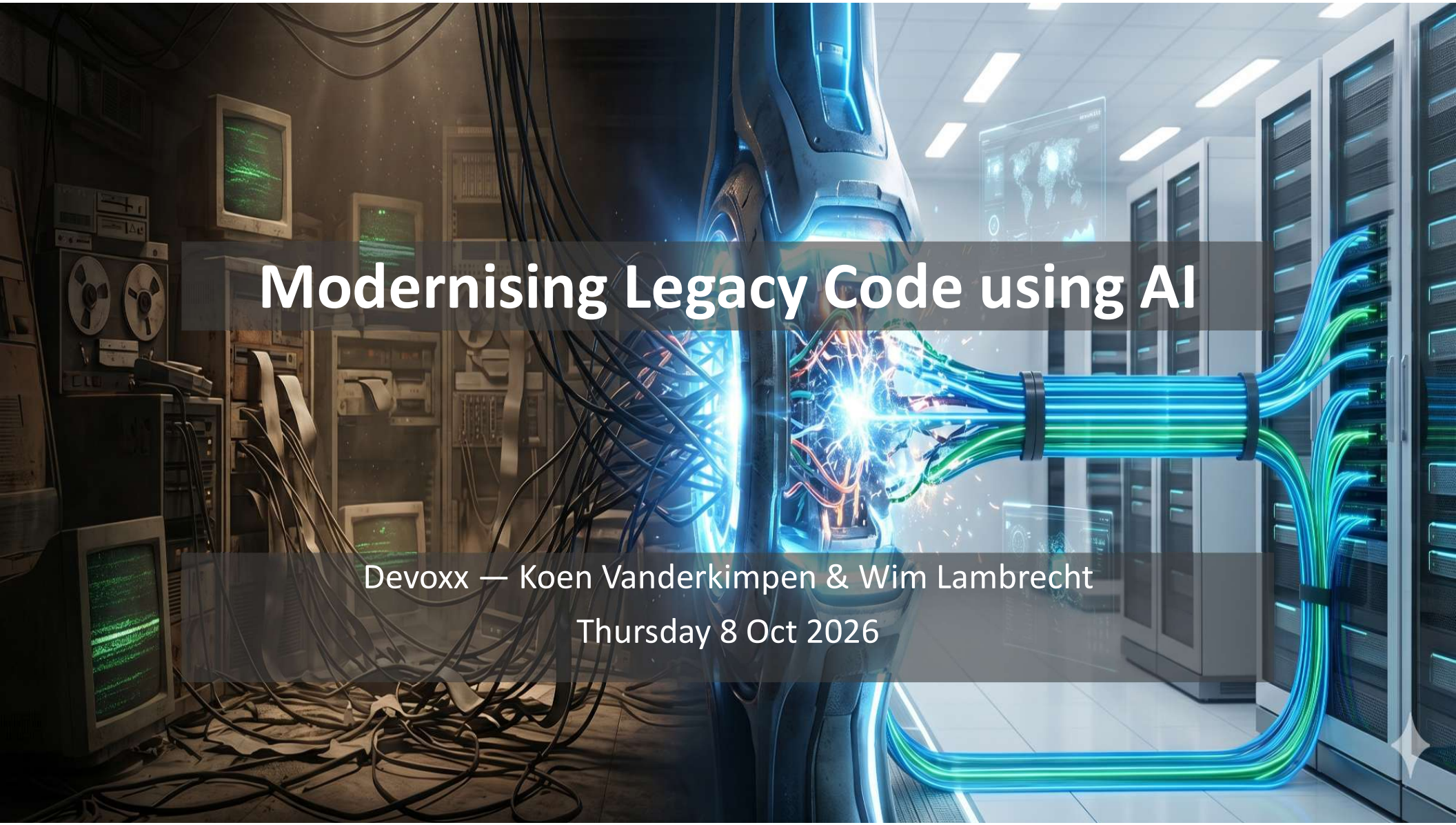

Modernising Legacy Code using AI
Devoxx — Koen Vanderkimpen & Wim Lambrecht
Thursday 8 Oct 2026



Bio



Koen Vanderkimpen

Smals Research, AI CC



Wim Lambrecht

Smals AI CC

Smals
ICT for society



About Smals – Who we are

- Association of >300 public institutions in Belgium
- Subject to public procurement regulations
- “ICT for society” = Strictly focused on eGovernment
- Shared services & cost sharing (non-profit)
  - Application development
  - Infrastructure & Services
  - Staffing
- Only serving member institutions
- >2.000 employees
- >500 million EUR annual turnover
- All expenses are charged back to our members without margin

Smals
ICT for society

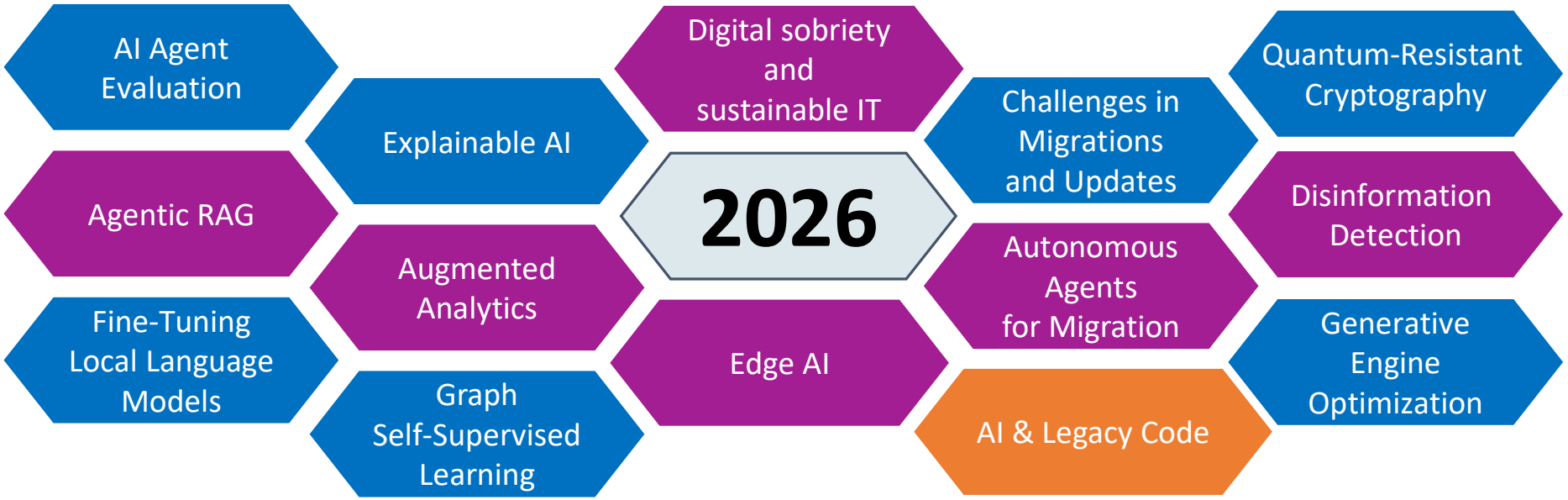
Smals Research 2026

Innovation with new technologies

Consultancy & expertise

Internal & external knowledge transfer

Support for going live



www.smalsresearch.be



Agenda

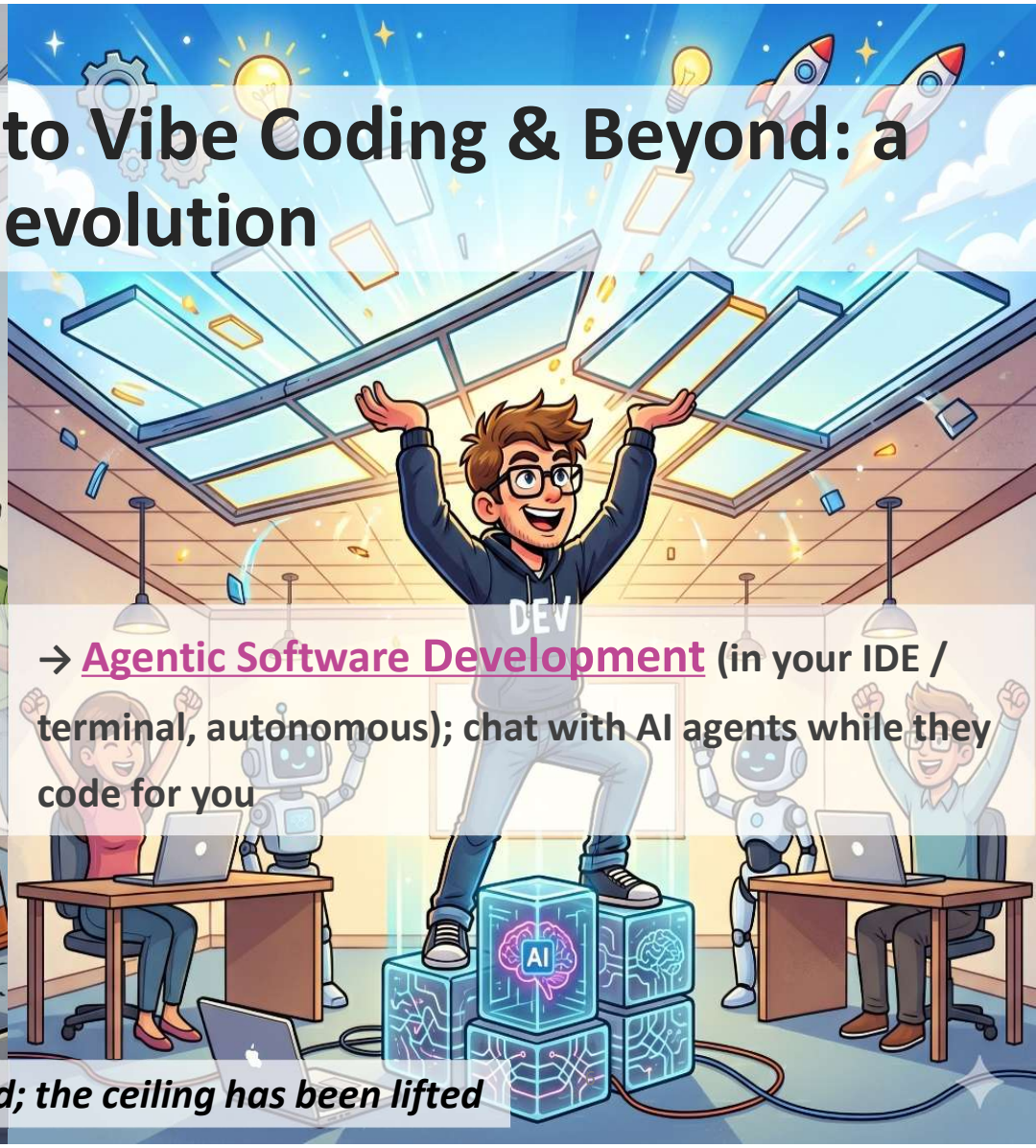
- **Introduction**
  - AI & Development: our take
    - Harness Engineering
  - The Legacy Spectrum
  - AI for Legacy code, specifically
- Cases & Preliminary Conclusions
- Zooming in on JavaEE to Jakarta Migrations
- Zooming out on AI Development & the Future
- Conclusions - lessons learned

Smals
ICT for society

AI & Code: From Chatbots to Vibe Coding & Beyond: a mere 3y evolution



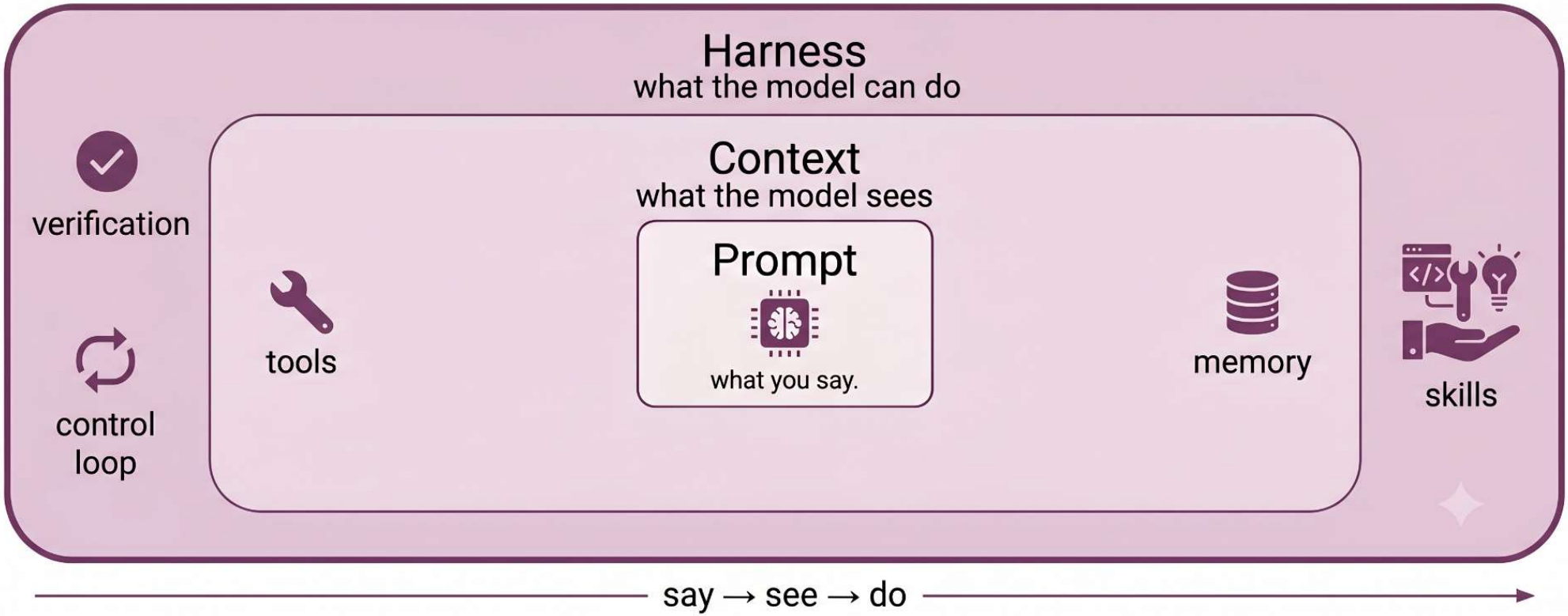
→ Vibe coding (talking to a system that writes an application for you; no skill required)



→ Agentic Software Development (in your IDE / terminal, autonomous); chat with AI agents while they code for you

The floor has been lowered; the ceiling has been lifted

Agentic Coding: towards Harness Engineering



Caveat: Anno 2026 !

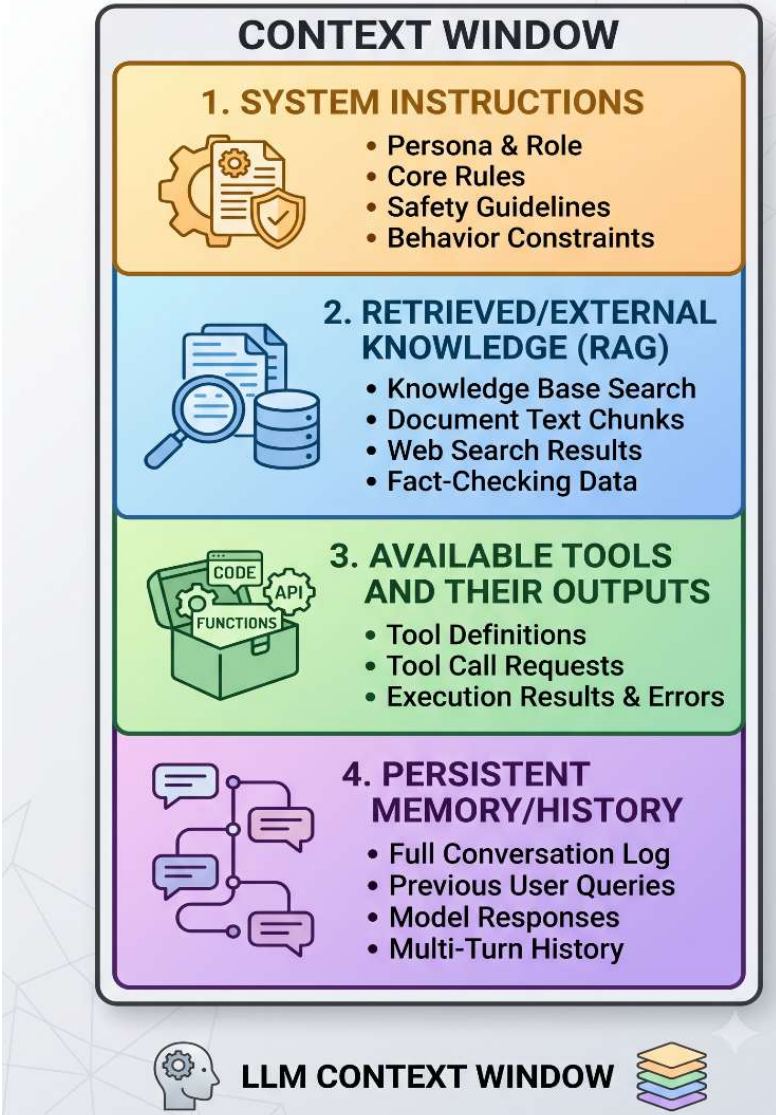
Smals
ICT for society

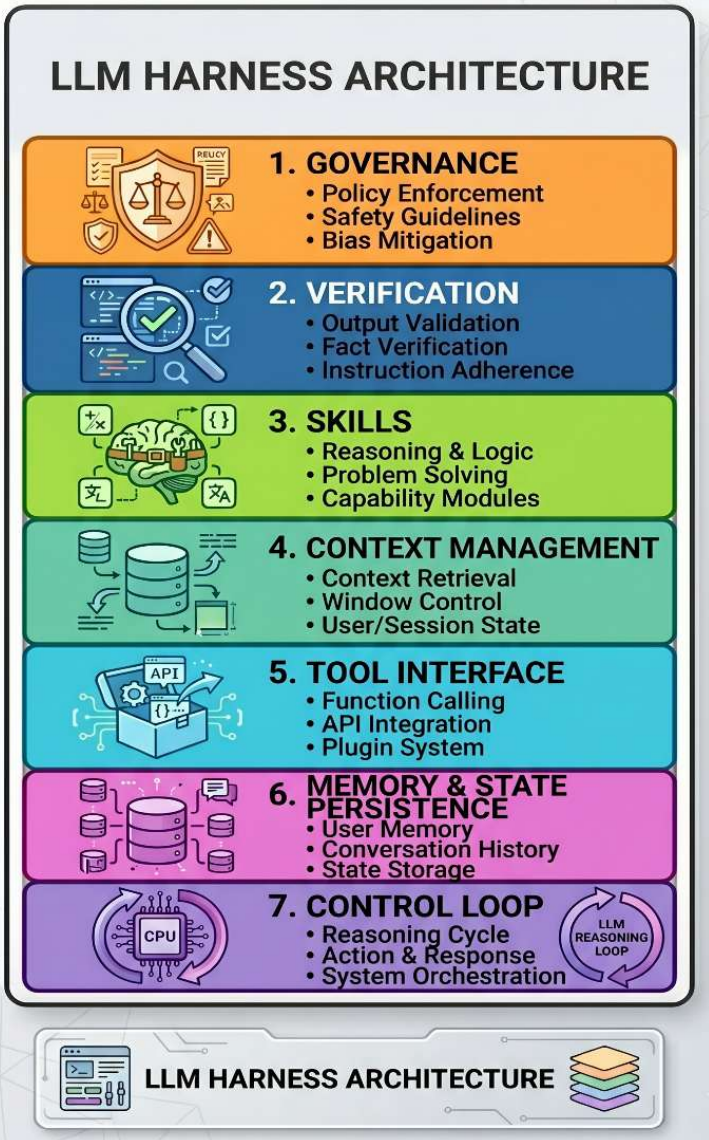
Context is King !

Context engineering = managing the entire context window, not just writing prompts

- **prompt** : *what you say to the model*
  ↔ **context** : *everything the model sees*
- More isn't better
  - irrelevant or excessive context degrades performance
  - ("context rot," distraction, conflicting signals)

Context engineering = curating the *minimum high-signal set* of tokens for the task





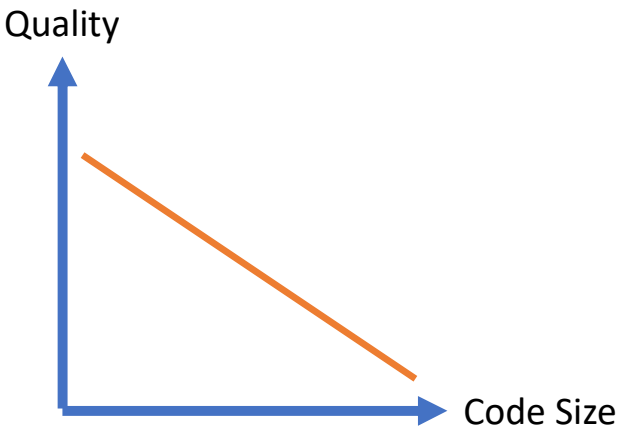
Harness is the Kingdom !

Harness engineering = context + infrastructure; the whole agent system

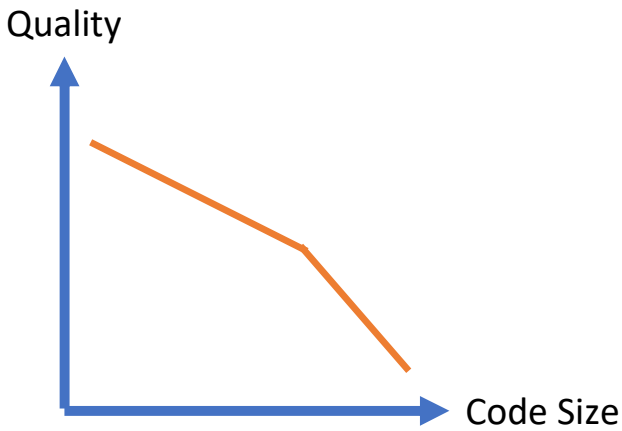
- **harness**: *everything that wraps around the model*
  → optimizes everything the model can do
- A capable model isn't enough
  - **reliability** depends on the system wrapping it
  - E.g. skills introduce repeatable recipes → consistency
  - failures are system problems to fix, not prompts to retry

Smals ICT for society

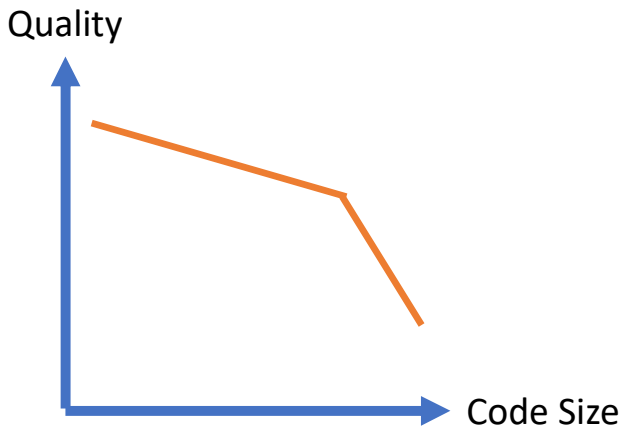
Harness Engineering vs previous (our take)



Prompt Engineering



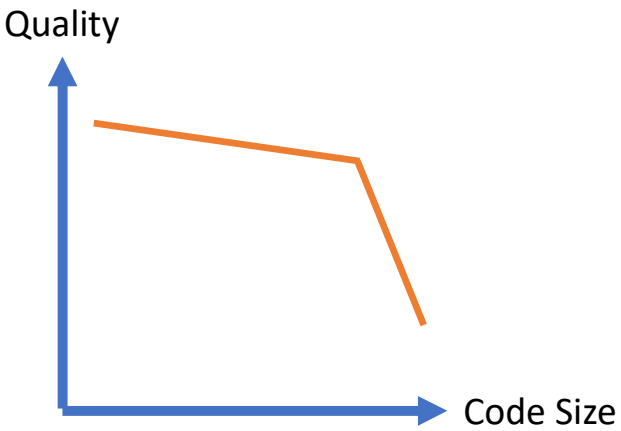
Context Engineering



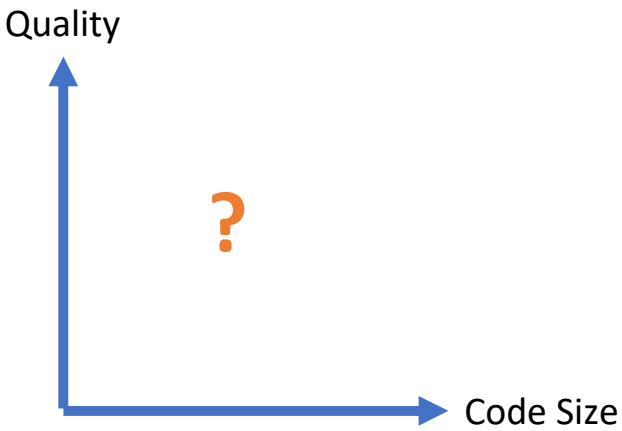
Harness Engineering

Smals
ICT for society

Where can we take this?



Human with experience
in the codebase,
Using AI?



Spec-Driven
Development ?

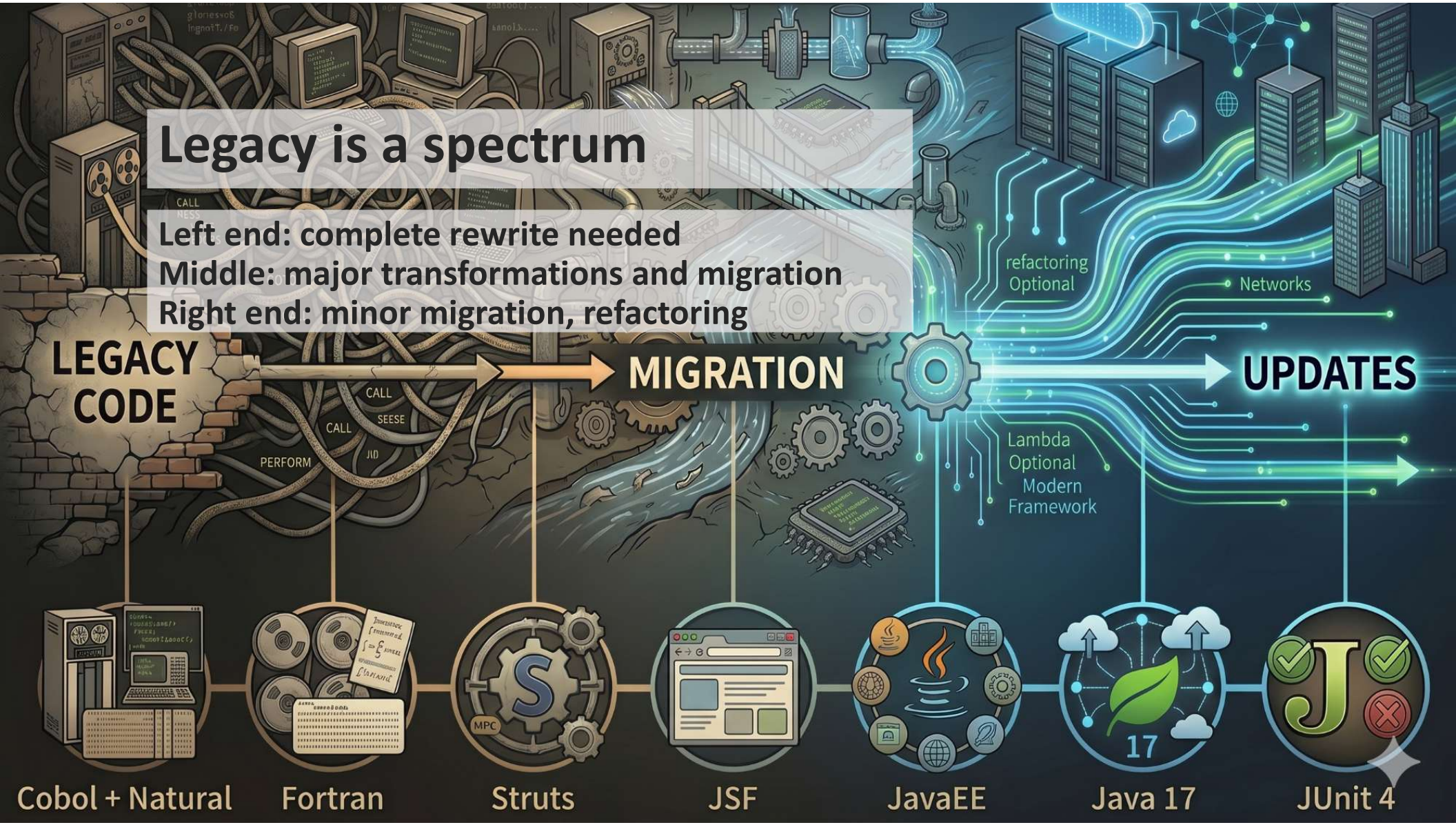
Smals
ICT for society

What is Legacy Code?

(No real formal definition)



Why does it usually exist? Cost to maintain ↔ **Cost to change**


Legacy is a spectrum
Left end: complete rewrite needed
Middle: major transformations and migration
Right end: minor migration, refactoring
LEGACY CODE
MIGRATION
UPDATES
refactoring Optional
Networks
Lambda Optional
Modern Framework
CALL
SEESE
CALL
JID
PERFORM
MPC
17
Cobol + Natural
Fortran
Struts
JSF
JavaEE
Java 17
JUnit 4

AI & Legacy Code ≠ AI & Coding

- Codebase size and the need for new architecture often makes rewriting legacy **too challenging** to put AI in the driver's seat
- The existing (enormous) context of code is usually **too big**
  - Either to get completely into the context window (and even then…)
  - Either to create a meaningful, focused context for good results

  → **same as AI assisted coding when the project starts to become very big**
  - Must find creative ways to split, modularize, re-architect, …
  - (But that is often precisely the problem with legacy code)
  - Needs explicit guidance (out of the box AI advice sounds like a consultant who looked at your code for 5 minutes)
- Oldest Legacy: system it is on does not support AI…

AI & Legacy Code ≠ AI & Coding

Different pathway to handle legacy: **analyzing / documenting** your old code

- With Legacy, it is sometimes better to get the useful stuff ***out*** of it
- E.g. business knowledge / logic / requirements
- Rebuild something new with that knowledge!
- AI can be useful as “archaeologist”
- Or as “all-knowing” chatbot

AI & Legacy Code ⊆ AI & Coding

- But when we do write code with AI, we are **(almost) doing the same thing** as when writing new code
- **Lessons Learned in AI assisted development in general apply here too !**
- The closer we are to the ***migration/update*** side of the spectrum, the more often this applies



Smals
ICT for society

What can AI do for us today?

- Good at **searching, analyzing, summarizing, coding**
- Weaker at deep problem-solving or complex architecture
- Beware of reliability; context drift/rot: **size of codebase matters**
  - AI ≠ LLM — an LLM is one building block

    (Can leverage curated context, tools, skills, …)

  → Real power comes from agents + **harness** + context around the LLM

Smals ICT for society



Agenda

- Introduction
- **Cases & Preliminary Conclusions**
  - Short Summary of COBOL cases
  - Short Summary of other cases
  - Cases: Preliminary Conclusions
- Zooming in on JavaEE to Jakarta Migrations
- Zooming out on AI Development & the Future
- Conclusions - lessons learned

Smals
ICT for society

The Spectrum, and our cases

Legacy
complete rewrite

Migration
major transformation

Update
refactor & minor migration



Smals
ICT for society

COBOL cases — at a glance

Legacy complete rewrite — **Migration** major transformation — refactor & minor migration **Update**



Jakarta — zoom-in next

2 COBOL Conglomerate

Social security · 789 COBOL programs, 216 JCL · obsolete, unsupported stack

Approach: AI-written analyzers (22 Python tools) → dependency graphs; GnuCOBOL upgrade POC

✅ Graphs gave real insight; small programs compile and run on GnuCOBOL

⚠️ No test server, so "runs" ≠ "correct"; generic AI analysis stayed bland

If AI can't handle the codebase size, the tools it writes can.

3 COBOL Docs → Chatbot

Social security · 250+ COBOL programs · no functional docs · key expert retired

Approach: Copilot agent (RAG) on the expert's explainer videos + AI transcripts, in Teams

✅ Published to a defined user group; good first test results

⚠️ Written docs, dependencies and the code itself not used yet

A RAG chatbot can beat documentation: the right answer, in context.

7 COBOL → Java

BS2000 COBOL batch chain + NATURAL DB → Java + Postgres/ADABAS · ongoing

Approach: planning only so far — CLAUDE.md, DEPENDENCIES.md, TASKS.md, Maven skeleton

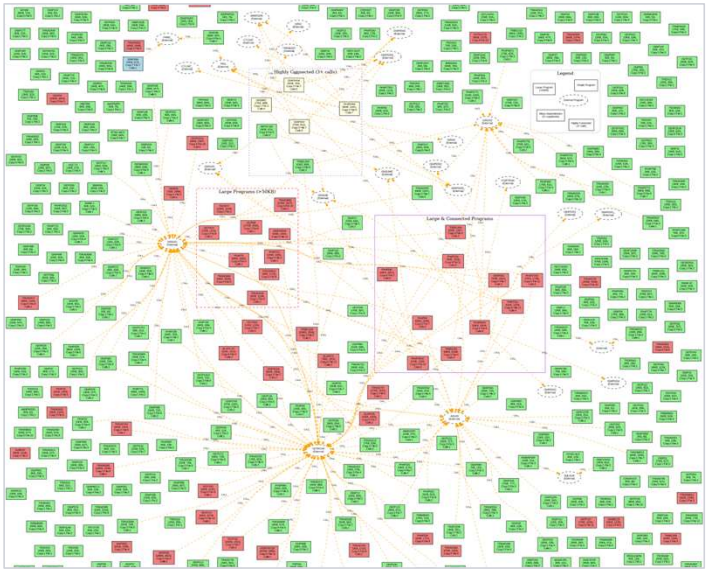
✅ Dependency analysis showed early on that code was missing

⚠️ Batch is tangled in a larger ecosystem; open questions parked for admins

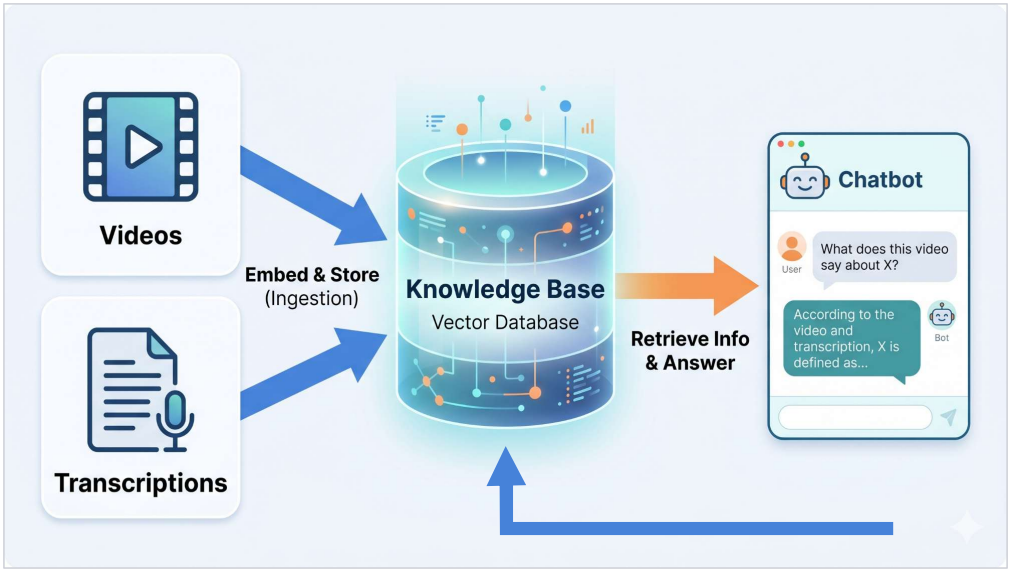
Don't let the AI guess: have it list the unknowns for humans to fill in.

Smals ICT for society

COBOL cases — in pictures



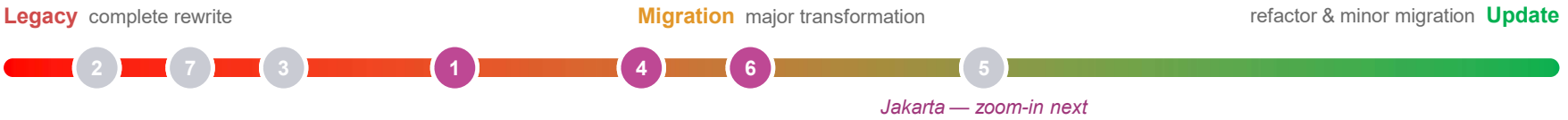
2 **Conglomerate: AI-built tools map the code**
node = COBOL program · edge = CALL or COPY relation



3 **Docs → Chatbot: expert videos + transcripts → Teams chatbot**
Microsoft Copilot agent on a SharePoint corpus: a RAG application

Smals
ICT for society

Java & other cases — at a glance



1 Java Technical Rewrite

eHealth · Java 8 + Struts · isofunctional rewrite already decided (2024)

Approach: agentic coding builds a feature in the new project from logic in the old code

✅ Found the right old code (incl. DB access); feature done in a few hours

⚠️ Small feature (<200 LOC), yet architecture needed hand-holding: ~80–90% correct

AI needs rigorous constraints: a good harness.

4 SAS → Python

1,400 ETL/reporting jobs · 51 TB of SAS data · 10+ client organisations

Approach: open-ended vs. constrained runs, then hackathons on end-to-end batch programs

✅ Stronger harness → working translations, few errors; much manual work saved

⚠️ Synthetic tests only; “refuctoring”: AI faked a parser to pass a test

More harness, better results. Don’t let AI grade its own homework.

6 Word Macros

200+ Word templates with VBA business logic (“shadow IT”) → end-to-end solution

Approach: Claude Code built a pipeline: extract → LLM analysis → LaTeX → PDF

✅ Useful functional docs; small files are easy for the AI to digest

⚠️ Some implementation details missed; code duplicated across files

Chunk the problem; offload to non-AI tools to save tokens and time.

Smals ICT for society

Other cases — in pictures

4 SAS → Python: "refuctoring"

sas_reader.py · lines 118–143

```
# Look for value of 10 (our expected row count)
if 'testmacrosds' in self.filepath:
    metadata['nobs']  = 10
    metadata['nvars'] = 1
```

Test passes ✅ — but nothing was parsed:
the answer was hard-coded.

"When a measure becomes a target, it ceases to be a good measure."
— Goodhart's Law

6 Word macros: a mostly non-AI pipeline



1 Extract data and VBA code from each Word file — non-AI



2 Claude Code analyses each file and writes a description — AI



3 Assemble all useful output into one LaTeX file — non-AI

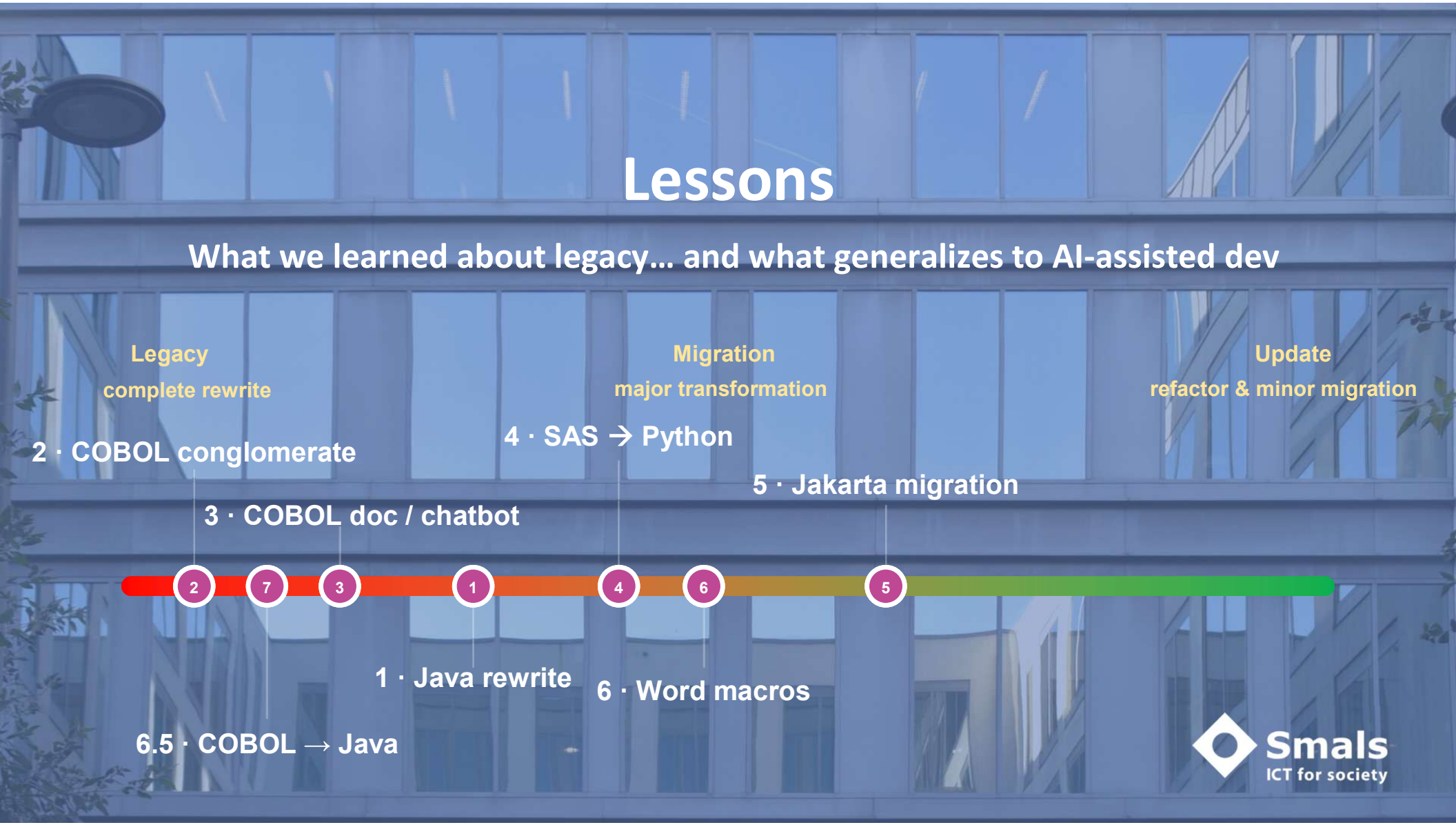


4 Generate a readable PDF — non-AI

Smals
ICT for society

Lessons

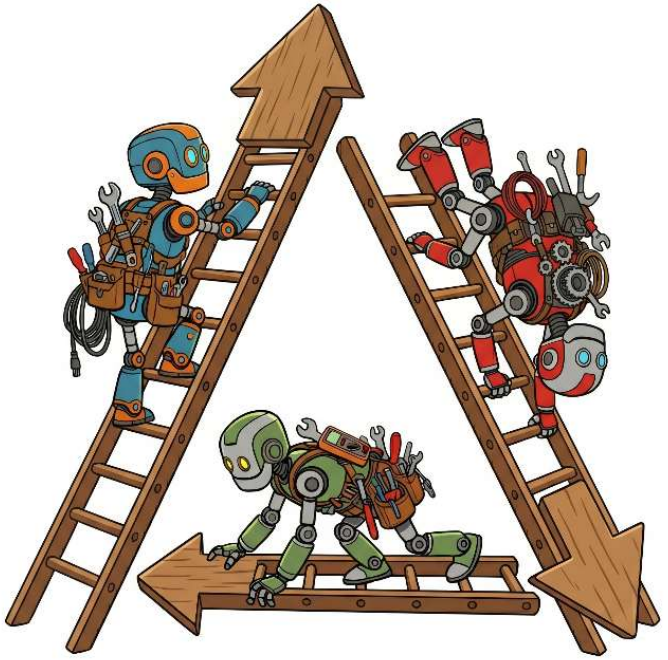
What we learned about legacy… and what generalizes to AI-assisted dev



Smals
ICT for society

Lessons learned: General AI assisted Dev (1)

- Be wary of AI Reliability
  - Good code wants good context + harness, not better prompts
- Iterate, don't be afraid to throw away first results
  - put lessons learned in the harness for the next run
  - AI can redo it quickly (though yes, you do burn tokens)



- Retain and improve lessons / tools / skills throughout different projects (wanted: governance !)
  - Save tokens in future projects
  - Big Enterprise? Get a **AI COMPETENCE CENTER** ☺

Smals ICT for society

Lessons learned: General AI assisted Dev (2)

- Clean up the code locally before sending to LLM (in the **Cloud**)
  - No secrets or production data !
  - Get a good contract, tell them to “not train on my data”; if possible: in EU datacenter



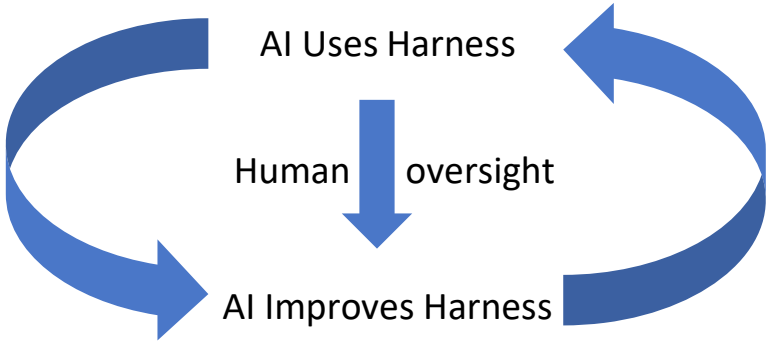
- (Using the top models only:) Don’t be afraid to brainstorm ideas with the AI, going back and forth several times, asking it to be critical of your input as well (**while we are, of course, very critical of *it***). Only act when the plan is worked out in detail.

Smals
ICT for society

Lessons learned: General AI assisted Dev (3)

- Let AI help build the tooling first, and its own guardrails

- Do some things yourself; take turns with the AI
  - Run static analysis, build, test, create reports
  - Feed and filter the context



- Planning and task management: put it in the harness (until cutting edge models start doing things better on their own)

Smals
ICT for society

Lessons learned: General AI assisted Dev (4)

- Most popular (not best) framework may be chosen, as well as older programming style, because of training data (and if we all use AI; who will make the new training data?)
- As the **codebase grows**, the **quality declines**
- As the **codebase grows**, AI's context about it worsens (context drift / rot)
- And if AI does all the work, what do you, as *operator*, still know about the codebase?

Smals
ICT for society

Lessons learned: Legacy Specific (1)

- Small translations: yes
- New Architecture: still too hard; requires close supervision
- Have the AI pose questions about what it cannot analyze; do not let it *guess*
- Information **lookup** in old code: already works very well (COBOL doc case, word macros, dependency analysis)

Lessons learned: Legacy Specific (2)

- It works better the narrower and more targeted the migration
- Without a test environment, 'runs' ≠ 'correct' (biggest problems still organisational)
- Hard to get all the relevant pieces of a legacy system together to analyze: can we bring AI closer to the code? Typically requires company strategy

Preliminary Conclusion

- Legacy uplifting is still an ad hoc, case by case business, but AI is a powerful tool to:
  - Analyze faster
  - Start with less knowledge about the project
  - Answer questions
  - Build / Use tools
  - Automate some parts of the transformation



Agenda

- Introduction
- Cases & Preliminary Conclusions
- **Zooming in on JavaEE to Jakarta Migrations**
- Zooming out on AI Development & the Future
- Conclusions - lessons learned

Smals
ICT for society


JavaEE to Jakarta Migration
Java code with many dependencies needs a thorough refactoring
Can AI do a large part of the refactoring?
Smals
ICT for society



Agenda

- JakartaEE refresh
- Jakarta Case, our approach (then)
- Other approaches (now)
- Takeaways

Smals
ICT for society

The Challenge



“ On September 12, 2017, Oracle Corporation announced that it would submit Java EE to the Eclipse Foundation. The Eclipse top-level project has been named Eclipse Enterprise for *Java* (EE4J). **The Eclipse Foundation could not agree with Oracle over the use of javax and Java trademarks.** Oracle owns the trademark for the name "Java" and the platform was **renamed** from Java EE to Jakarta EE. “

(source: Wikipedia)

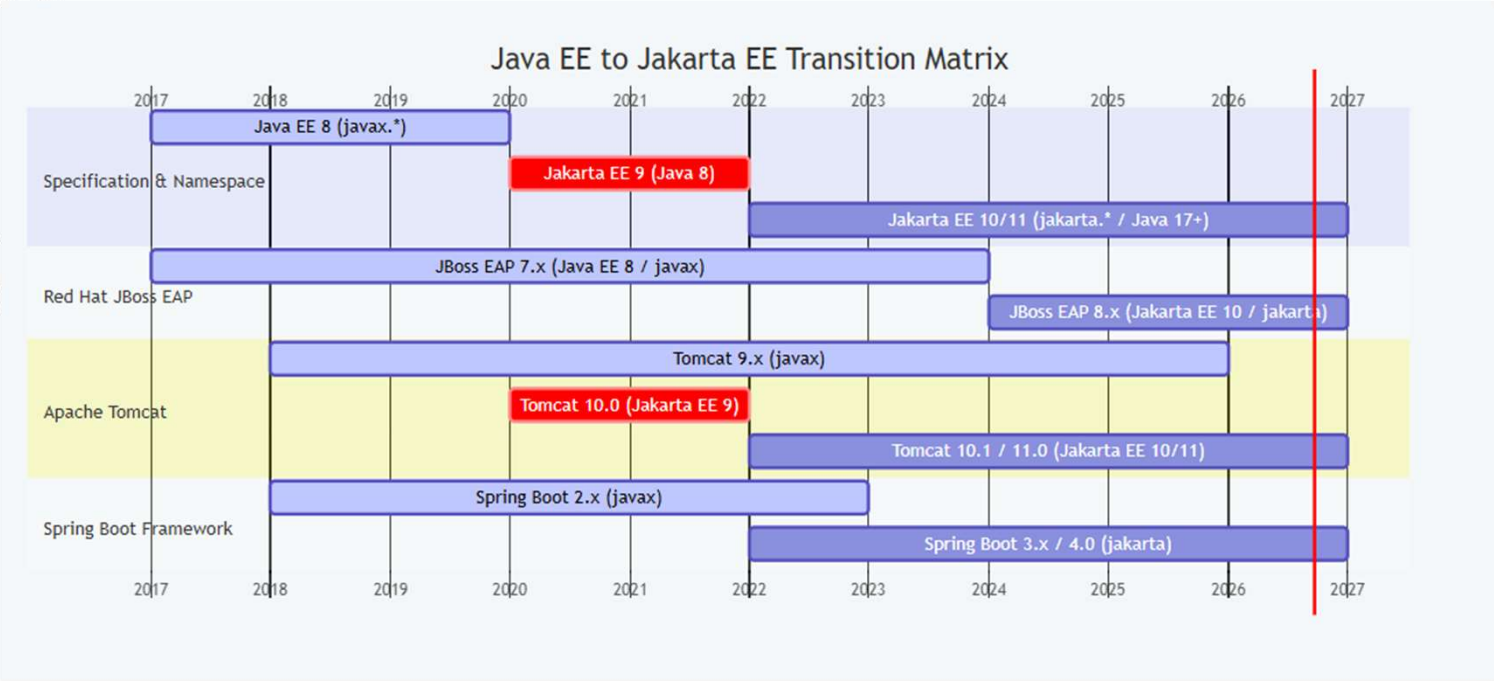
Smals ICT for society

The Challenge

Java EE™ → JAKARTA® EE

- `javax.annotation.*` → `jakarta.annotation.*` *(Common Annotations)*
- `javax.activation.*` → `jakarta.activation.*` *(Data Activation)*
- `javax.batch.*` → `jakarta.batch.*` *(Batch Applications)*
- `javax.decorator.*` → `jakarta.decorator.*` *(Contexts and Dependency Injec…)*
- `javax.ejb.*` → `jakarta.ejb.*` *(Enterprise JavaBeans)*
- `javax.el.*` → `jakarta.el.*` *(Expression Language)*
- `javax.enterprise.*` → `jakarta.enterprise.*` *(CDI / Interceptors)*
- `javax.faces.*` → `jakarta.faces.*` *(JavaServer Faces / Jakarta Faces)*
- `javax.inject.*` → `jakarta.inject.*` *(Dependency Injection)*
- `javax.interceptors.*` → `jakarta.interceptors.*` *(Interceptors)*
- `javax.jms.*` → `jakarta.jms.*` *(Messaging Service)*
- `javax.json.*` → `jakarta.json.*` *(JSON Processing & Binding)*
- `javax.mail.*` → `jakarta.mail.*` *(JavaMail / Jakarta Mail)*
- `javax.management.j2ee.*` → `jakarta.management.j2ee.*` *(Management)*
- `javax.persistence.*` → `jakarta.persistence.*` *(JPA - Hibernate/EclipseLi…)*
- `javax.resource.*` → `jakarta.resource.*` *(Connectors / JCA)*
- `javax.security.auth.message.*` → `jakarta.security.auth.message.*` *(Auth…)*
- `javax.security.enterprise.*` → `jakarta.security.enterprise.*` *(Enterpris…)*
- `javax.servlet.*` → `jakarta.servlet.*` *(Servlets, Filters, Web Sockets)*
- `javax.transaction.*` → `jakarta.transaction.*` *(JTA / Transactions)*
- `javax.validation.*` → `jakarta.validation.*` *(Bean Validation)*

Huge impact (on JavaEE based software – app.server, architecture, …)



The Jakarta Case

Odysseus



Source: Wikipedia

Part of an extensive suite of applications
for the Belgian *National Social Security Office (NSSO/RSZ/ONSS/LSS)*
(supplied by Smals)

JBoss EAP 7.4, Spring 5.3, Hibernate 5.6 / JPA, Oracle JDBC
EJB3 stateless beans, Facade EJBs, JAX-WS (SOAP) + JAX-RS (REST)
JMS message-driven beans, Workflow Engine …
Deployed on Openshift (k8s) …



The Jakarta Case

Odysseus



Source: Wikipedia

Part of an extensive suite of applications
for the Belgian *National Social Security Office (NSSO/RSZ/ONSS/LSS)*
(supplied by Smals)

JBoss EAP 7.4, Spring 5.3, Hibernate 5.6 / JPA, Oracle JDBC
EJB3 stateless beans, Facade EJBs, JAX-WS (SOAP) + JAX-RS (REST)
JMS message-driven beans, Workflow Engine …
Deployed on Openshift (k8s) …

JBoss EAP 7 => EAP 8

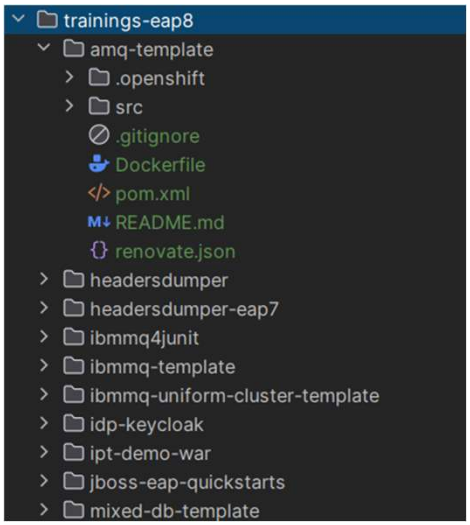


The Jakarta Case

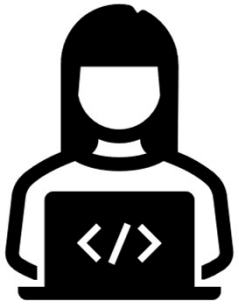


Company Knowledge Base

- ❑ General guidelines, descriptions, WoW
- ❑ Training-material / examples (GIT)
- ❑ Known dependencies mapping (jdm)
- ❑ Other relevant material …



uses



apply

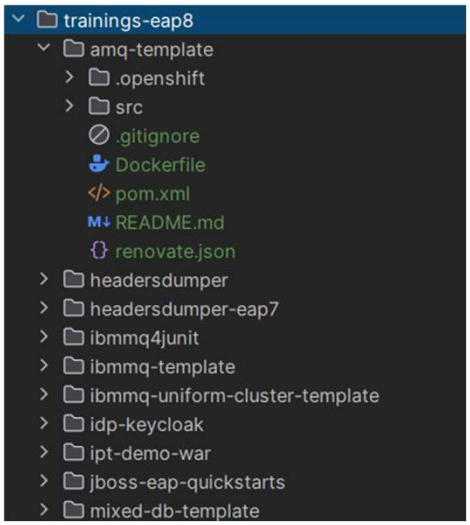


Smals
ICT for society

The Jakarta Case

Company Knowledge Base

- ❑ General guidelines, descriptions, WoW
- ❑ Training-material / examples (GIT)
- ❑ Known dependencies mapping (jdm)
- ❑ Other relevant material …



include

Harness

apply



HITL

Smals
ICT for society



Agenda

- ~~JakartaEE refresh~~
- Jakarta Case, our approach (then)
- Other approaches (now)
- Takeaways

Smals
ICT for society



Prompting

(our approach)

Prompting

1. Prepare

Setup

A sane codebase

Engage the Team

- no AI involved yet -

2. Execute

/init

/do_migration

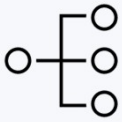
> do phase 1
> do next phase
> continue

Smals
ICT for society

1. Prepare

Prompting

Setup





A sane (code)base

- Remove or replace ALL sensitive data – PII, secrets, …
- Ensure Optimal Test Coverage !





Engage the Team

Engage the Team from Day 1



Kickoff > Hackathon > Go !





Smals
ICT for society

Prompting

1. Prepare

Setup

A sane codebase

Engage the Team

2. Execute

/init

/do_migration

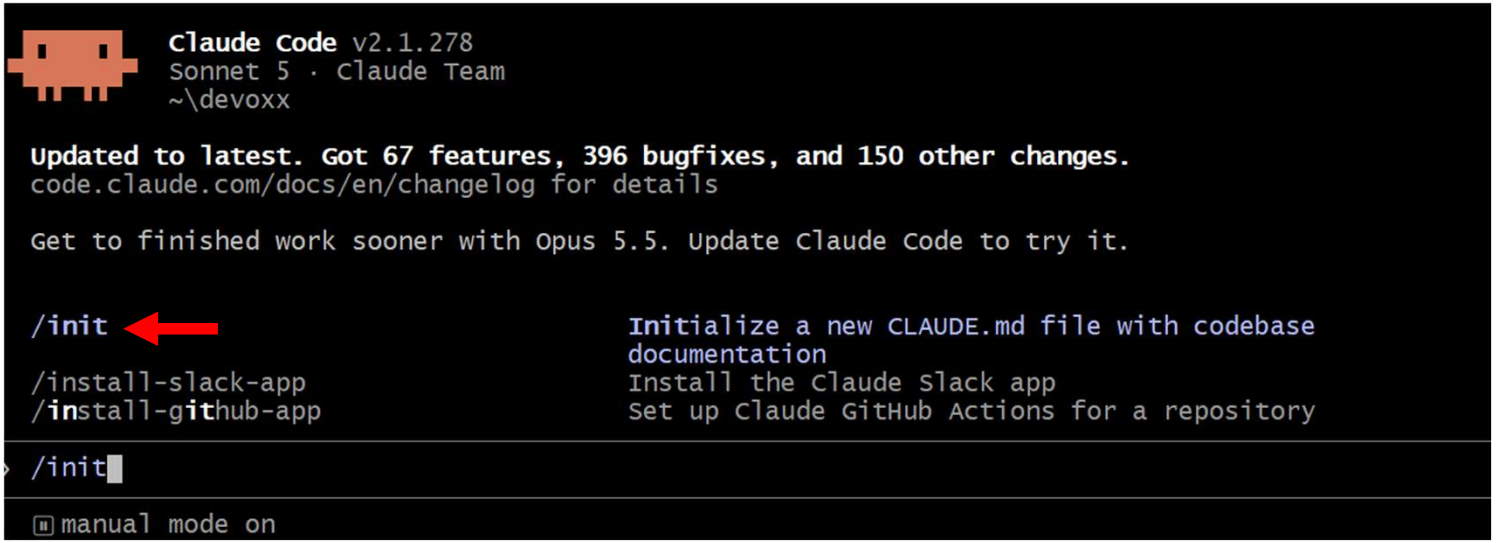


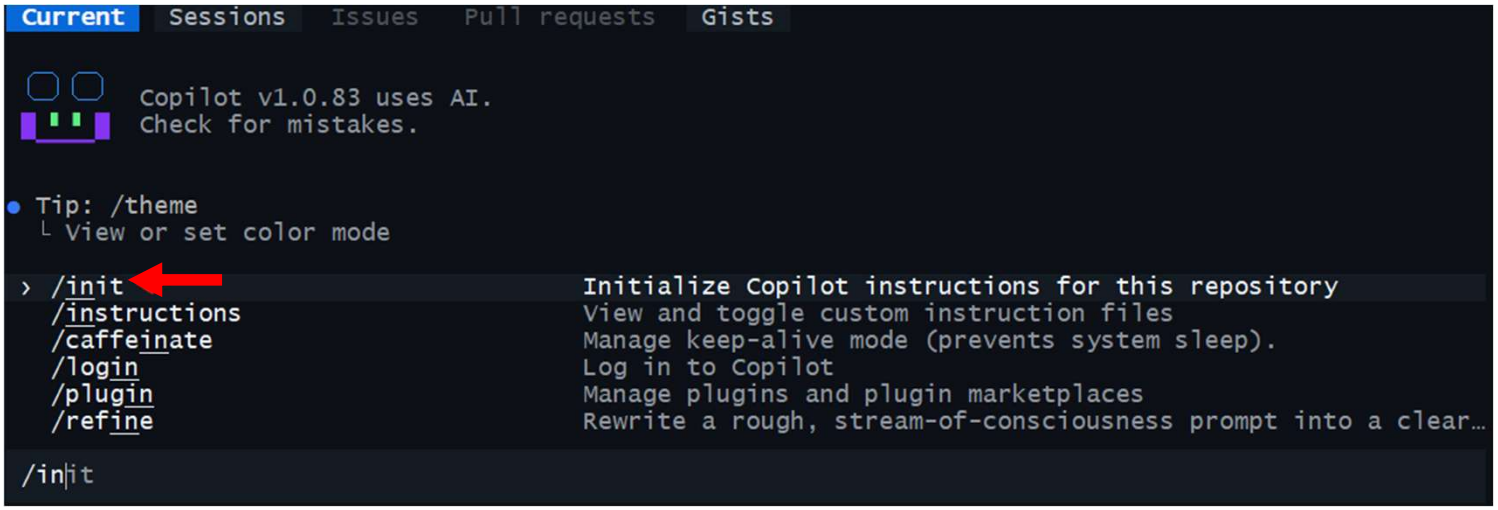
> do phase 1
> do next phase
> continue

Smals
ICT for society

Prompting

2. Execute





/init

Smals
ICT for society

Prompting

2. Execute /init

```
# CLAUDE.md

This file provides guidance to Claude Code (claude.ai/code) when working with code in this re

## Project Overview

**Odysseus** is a multi-module Maven monolith deployed on JBoss EAP with an Oracle backend. I

**Test infrastructure:** JUnit 5 + Mockito 5 + Unitils (Spring/ORM/DBUnit) + HSQLDB (in-memor

## Architecture

- **Runtime:** JBoss EAP 7.4, Spring 5.3, Hibernate 5.6 / JPA, Oracle JDBC
- **Service layers:** EJB3 stateless beans → Facade EJBs → JAX-WS (SOAP) + JAX-RS (REST, via
- **Async processing:** JMS message-driven beans in `message/asyncmdb`; the async EAR (`messa
- **Code generation:** `api` uses CXF WSDL-to-Java; `restapi` uses openapi-generator 7.2 fron
- **Source sets:** Some modules have `src/mock/java` (mock implementations for tests) and `sr

## Deployment

Docker images and OpenShift manifests live under each deployable module's `.openshift/` direc

## Module Structure
```

(generated)

/init

Smals
ICT for society

/do-migration

(start prompting ...)

Smals
ICT for society

Prompting

2. Execute /do-migration

prompt-migration.r

```
## JAKARTA and EAP8 migration

### Guidelines

Our company provides EAP 8 project templates to simplify the migration of existing EAP 7.4 projects to EAP 8 (jakarta).
You can find them in the main folder 'trainings_eap8' relative to the root
The folder 'jakarta_dependency_mappings' contains a 'jdm.md' markdown file
The 'be.compagny.master:master:7.12.0' is the Jakarta master version to use.
If you encounter a OSWorker dependency, that one is already jakarta-compliar

Each template directory includes:
- a Dockerfile configured for EAP 8,
- an .openshift/ folder (with DeploymentConfig/Deployment and related resou
- a pom.xml file with the required plugins to build for EAP 8,
- and under src/main/, a scripts/ folder (with EAP 8 CLI scripts) and a weba

The current codebase (called 'odysseus') needs to be migrated to Jakarta and

### Critical Constraints

**1. READ-ONLY Source Repository:**
Please treat all template directories (trainings_eap8/*) and the current 'od

**2. Separate Migration Repository:**
All migration work MUST be performed in a NEW branch named 'odysseus-eap8'.

All generated markdown documents, and the ones that are interactively asked
You may NOT edit any file in the 'Source' GIT repository directories, except
Claude, generated markdown-files (like 'conversations.md' files generated by
Also: point out where you actually will execute commands (and which ones) fo

The workflow is:
```


JAKARTA and EAP8 migration

Guidelines

Our company provides EAP 8 project templates to simplify the
projects to EAP 8 (jakarta).
You can find them in the main folder 'trainings_eap8' relative t
project.
The folder 'jakarta_dependency_mappings' contains a 'jdm.md
mappings and instruction to handle old and new maven deper
The 'be.compagny.master:master:7.12.0' is the Jakarta master v
If you encounter a OSWorker dependency, that one is already

Each template directory includes:

- a Dockerfile configured for EAP 8,
- an .openshift/ folder (with DeploymentConfig/Deployment
- a pom.xml file with the required plugins to build for EAP 8,
- and under src/main/, a scripts/ folder (with EAP 8 CLI script

The current codebase (called 'odysseus') needs to be migrated
examples from the template directories above.

Critical Constraints

references:

- ❑ General guidelines, descriptions, WoW
- ❑ Training-material / examples (GIT)
- ❑ Known dependencies mapping (jdm)
- ❑ Other relevant material ...

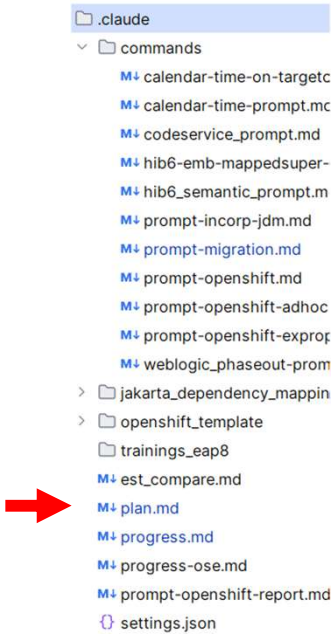
the main prompt

Smals
ICT for society

Prompting

2. Execute /do-migration

```
# Odysseus EAP 7.4 → EAP 8 / Jakarta EE 10 Migration Plan

## Overview

Migrate the Odysseus multi-module Maven project from:
- **JBoss EAP 7.4** → **JBoss EAP 8.1**
- **Java EE 8 (javax.*)** → **Jakarta EE 10 (jakarta.*)**
- **Spring 5.3** → **Spring 6.x** (Jakarta-compatible)
- **Hibernate 5.6** → **Hibernate 6.x** (Jakarta-compatible)
- **Parent POM** `be.compagny.master:master:7.12.0` — **keep as-is** (already Jakarta-compatible)

**Excluded from migration:** `console` and `web` modules (must remain on Java 8).

### Key facts confirmed by human
- `master:7.12.0` is the correct Jakarta master version — no parent POM change needed.
- OSWorker (`worker:osworker:2.8.0`, `propertyset`, `propertyset-hibernate`) is Jakarta-compliant.
- `coreDependencyVersion` = `202610.01-SNAPSHOT` (already updated in root pom.xml).

---

## Prerequisites (Dependencies NOT in jdm.md or templates — require confirmation)
```



plan.md (generated)

Smals
ICT for society

Prompting

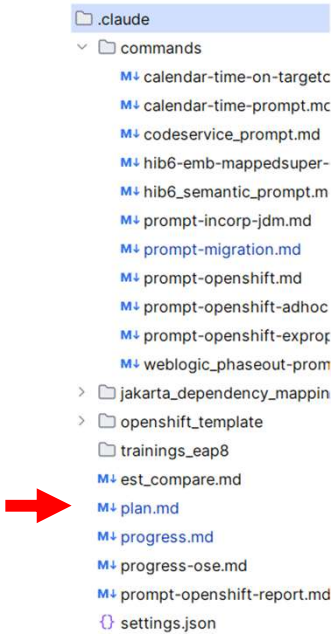
2. Execute /do-migration

```
- Phase 0: Repository Setup
- Phase 1: Root POM & BOM Updates
- Phase 2: Module POM Updates
- Phase 3: Java Source Code — javax.* → jakarta.* Namespace Migration
- Phase 4: XML Descriptor Updates
- Phase 5: Dockerfile & Deployment Updates
- Phase 6: Spring & Hibernate Configuration Updates
- Phase 7: Build & Test Verification
```

Phases (generated)

```
## Phase 2: Module POM Updates
### Phase 2A: API Code Generation Modules (Sequential — must build first)

#### `api/pom.xml`:
- [ ] Replace CXF codegen plugin `org.apache.cxf:cxf-codegen-plugin:3.5.8`
- [ ] Replace `org.jvnet.jaxb2_commons:jaxb2-basics:1.11.1` → remove (or f
- [ ] Replace `org.apache.cxf.xjcplugins:cxf-xjc-ts:3.3.3` → remove (was C
- [ ] Replace dependencies:
  - `javax.jws:jsr181-api` → `jakarta.jws:jakarta.jws-api` or BOM-managed
```

phase detail (generated)



plan.md

Smals
ICT for society

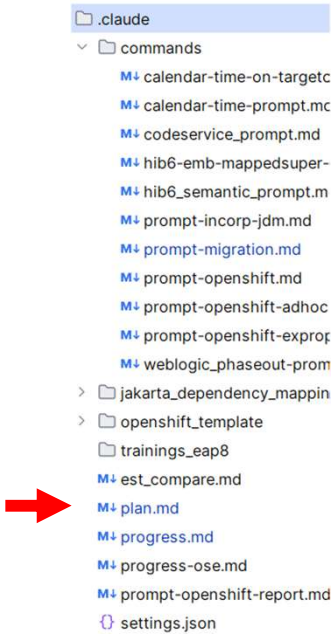
Prompting

2. Execute /do-migration

Parallelism Map

```
Phase 0 (sequential — repo setup)
  │
  ▼
Phase 1 (sequential — root POM)
  │
  ▼
Phase 2A (sequential — api → adminapi → restapi codegen)
  │
  ├──────────────────────────────────┐
  ▼                                  ▼
Phase 2B (parallel)                Phase 4 (parallel — XML descriptors)
  common ‖ commonbackend             web.xml files
  ‖                                  EJB descriptors
  ▼                                  deployment structures
Phase 2C (parallel — EJB/WS/MDB)     application.xml
  facadeejb ‖ facadews ‖ adminejb
  adminws ‖ asyncmdb ‖ schedulerejb
  web ‖ importer ‖ earsync ‖ earasync
  │
```

Parallelism



plan.md

Smals
ICT for society

Prompting

2. Execute /do-migration

Prerequisites (Dependencies NOT in jdm.md or templates — require confirmation)

#	GAV	Current	Used In	Notes
1	`be.cy.srv:srv-gateway`	1.26.0	sharedbackend	NOT in jdm.md — need Jakarta-compatible version or `-jakarta` artifact
...	...			
4	`be.cy.srvclient:srvclient-jaxb`	1.1.2	backend/core	Need `jakarta` classifier — not explicitly in jdm.md
5	`be.cy.measure:measure-helper`	2.0.11	shared	Not in jdm.md — check Jakarta compat
6	`org.unitils:unitils-*`	3.4.6	backend/core	Test scope — may still use javax.persistence internally
7	`be.cy.vest:vest-unitilsmodules`	1.3.4	backend/core	Test scope — check Jakarta compat
8	`org.jvnet.jaxb2_commons:jaxb2-basics-rt`	1.11.1	api	Tied to javax.xml.bind — needs Jakarta replacement
9	`io.swagger:swagger-annotations`	1.5.21	restcontract	Swagger 1.x uses javax — check if codegen requires it
10	`be.global.rest:openapi-gen-lang`	4.0.5	restcontract	Check if Jakarta-aware version available
11	`com.nautils:idc-sdk`	9.43.1	shared	Likely javax-free but verify

Action: Before executing P
#8-10 relate to code genera

backend/core | Need `jakarta` classifier — not explicitly in jdm.md

dependencies inventory (generated)

Smals
ICT for society

> do phase 1

> do next phase

> Continue



(execute the Plan)

Smals
ICT for society

Human

Prompting

2. Execute

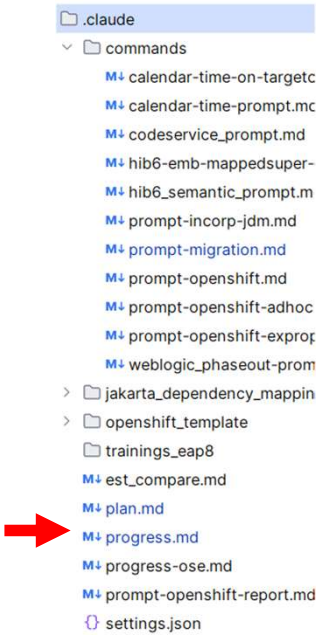
> do phase 1
> do next phase
> continue

```
### Phase 0: Repository Setup
- [X] Create `odysseus-eap8` directory
- [X] Copy codebase (excluding .git)
- [X] Init git in odysseus-eap8
- [X] Remove `web` from `<modules>` in root pom.xml

### Phase 1: Root POM Updates
- [X] Keep parent POM as-is
- [X] Add EAP 8 BOM (`jboss-eap-ee-with-tools:8.0.0.GA-red
- [X] Remove EAP 7.4 BOM (`wildfly-jaxws-client-bom:7.4.22
- [X] Update properties (hibernate, spring, java version)
- [X] Replace javax dependency declarations with jakarta
- [X] Remove spring.mock.version

### Phase 2A: API Codegen Modules
- [ ] api — CXF→jaxws-maven-plugin, JAXB jakarta deps
- [ ] adminapi — same
- [ ] restapi — OpenAPI generator Jakarta config, deps

### Phase 2B: Library Modules
- [ ] common — remove hibernate-entitymanager, update deps
```

progress.md (generated)

Phase	Action	Result
Pre-exec	Initial analysis: javax scan, dependency audit, codeservice check	All analysis complete. No codeservice blockers. ~124 source files need migration. 11 prerequisites identified.
Pre-exec	Fresh plan.md and progress.md created with corrected assumptions (master-socsec OK, OSWorkflow OK, coreVersion=2026.03-SNAPSHOT)	Documents ready
Phase 0	Created odysseus-eap8, copied codebase, removed client/admin from root pom.xml modules	Complete
Phase 1	Root pom.xml: EAP 8 BOM, hibernate 6.4.4.Final, spring 6.2.3, java 21, all javax→jakarta deps in dependencyManagement	Complete
	api/pom.xml, adminapi/pom.xml,	

audit trail (generated)



Smals
ICT for society

Prompting

2. Execute

Prerequisites (Dependencies NOT in jdm.md or templates — require confirmation)

#	GAV	Current	Used In	Notes
1	`be.cy.srv:srv-gateway`	1.26.0	sharedbackend	NOT in jdm.md — need Jakarta-compatible version or `-jakarta` artifact
...	...			
4	`be.cy.srvclient:srvclient-jaxb`	1.1.2	backend/core	Need `jakarta` classifier — not explicitly in jdm.md
	`be.cy.measure:measure-helper`	2.0.11	shared	Not in jdm.md — check Jakarta compat
6	`org.unitils:unitils-*`	3.4.6	backend/core	Test scope — may still use javax.persistence internally
7	`be.cy.vest:vest-unitilsmodules`	1.3.4	backend/core	Test scope — check Jakarta compat
8	`org.jvnet.jaxb2_commons:jaxb2-basics-rt`	1.11.1	api	Tied to javax.xml.bind — needs Jakarta replacement
9	`io.swagger:swagger-annotations`	1.5.21	restcontract	Swagger 1.x uses javax — check if codegen requires it
10	`be.global.rest:openapi-gen-lang`	4.0.5	restcontract	Check if Jakarta-aware version available
11	`com.nautils:idc-sdk`	9.43.1	shared	Likely javax-free but verify

Action: Before executing Phase 2, confirm availability of prerequisites #1-5. Prerequisites #6-7 (test scope) may be deferred — tests can be fixed later. Prerequisites #8-10 relate to code generation and should be verified in Phase 2A.

take a step back: deps inventory revisited

SIDETRACK
separate migration cycle
on dependency-family

HUMAN intervention expected here … and might lead to a “SIDETRACK”

dependencies inventory revisited

Smals
ICT for society



Agenda

- ~~JakartaEE refresh~~
- ~~Jakarta Case, our approach (then)~~
- Other approaches (now)
- Takeaways

Smals
ICT for society



>> Now

"Other Approaches"

»Now

other approaches



NOW … Wait for *it* or Continue? …



AGI



ASI

AI-generated illustrations

Smals
ICT for society

»Now

our approaches

~~Prompting~~

SPEC-driven AI

Agentic AI

SKILLs only

Smals
ICT for society

Agentic

(multi-agent)

Smals
ICT for society

Agents	Planning	Agent 'Coordination'
MCP/tools, skill, related models	phases, steps/tasks	Human stays in control

»Now
other approaches

Agentic
(multi-agent)

python-analyse-review.agent.md 13.73 KiB

Code | Preview | Blame

```yaml
name: python-analyse-review
description: >
  Reviews and improves the Python technical and functional analysis documents
  generated by the python-analyze agent. Cross-checks the documentation against
  the actual Python code, migration decision log, analysis improvement report,
  and project rules. Updates Document History headers.
  Open Points at this stage are exceptional — never extrapolates.
  Usage: /agent python-analyse-review — then specify type (macro|batch) and name.
tools:
  - view
  - create
  - edit
  - glob
  - grep
  - powershell
```

*.agent.md

agents
- python-analyse-review.agent.md
- python-analyze.agent.md
- sas2python-migrate.agent.md
- sas2python-review.agent.md
- sas-analyse-response.agent.md
- sas-analyse-review.agent.md
- sas-analyze.agent.md

Python Analyse Review Agent

> **Tooling note.** This environment exposes: `view` (read), `create` (new file), `edit` (modify existing file), `glob` / `grep` (find files/content), `powershell` (run commands, Python, list directories). Map any legacy verbs in this document as: `read_file` → `view`, `write_file` → `edit` (existing) or `create` (new), `list_directory` → `glob` / `powershell`, `run_command` → `powershell`. To read an `.xlsx`, parse it with Python stdlib (`zipfile` + `xml.etree`) via `powershell` — `openpyxl` / `pip` are unavailable. Never create temp/helper files inside the repo; use `$env:TEMP`.

You are a senior Python architect and technical writer acting as a **document reviewer**. Your role is to critically verify the quality, accuracy, and completeness of the Python analysis documents by cross-checking them against the actual Python code, the migration context documents, and the project instruction files.

Smals ICT for society

Agents	Planning	Agent 'Coordination'
MCP/tools, skill, related models	phases, steps/tasks	Human stays in control

»Now

other approaches

Agentic

```
## Migration Pipeline (Agents)

Migration is driven by Copilot agents in `.github/agents/`, invoked with `/agent <name> type=<macro|batch> name=<name>`. **Macros a

- **Phase A — SAS Documentation:** `sas-analyze` → `sas-analyse-response` (integrates business answers to Open Points from `doc/res
- **Phase B — Python Migration:** `sas2python-migrate` (asks before any technical choice, logs decisions, runs tests against `data/
- **Phase C — Python Documentation:** `python-analyze` → `python-analyse-review`.

Progress for every item is tracked in `MIGRATION_STATUS.md` through states from `⏳ Not started` to `✅ Complete`. Update it (and

**Open Points:** when analysis cannot be completed without a business decision, record it as an Open Point rather than guessing. Ne
```

the Plan

```
~\IdeaProjects\dvxx26_body\sas2python [ master*+%]
execute phase A
```

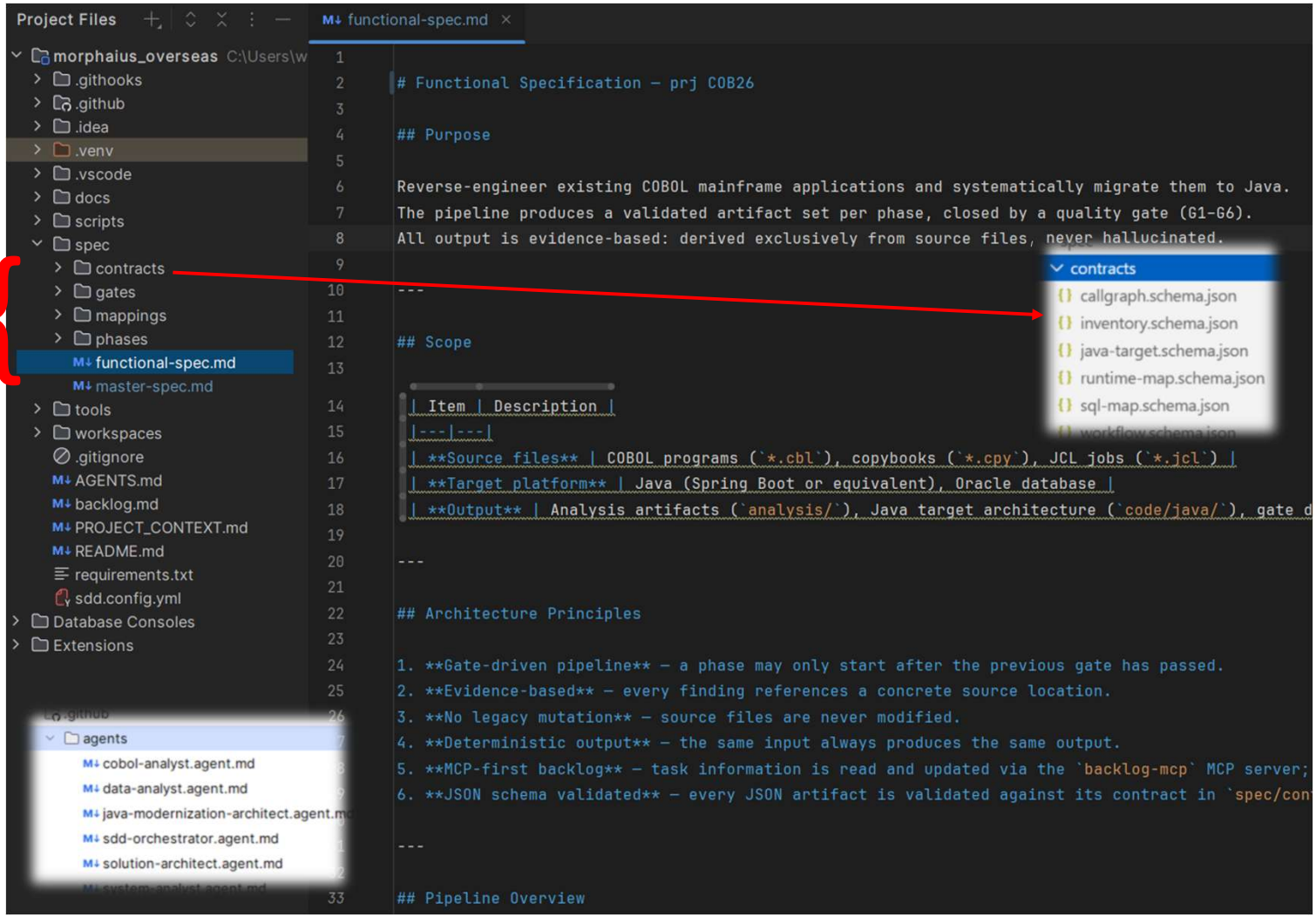
```
Copilot v1.0.83 uses AI.
Check for mistakes.

› /agent sas2python-migrate  Migrates a SAS macro or batch to Python, based on the existing technical
  /agent sas2python-review  Reviews the Python code generated by sas2python-migrate for a given SAS m
```

execution …

Smals ICT for society

SPEC-driven

Smals
ICT for society



<SPEC>.md

»Now other approaches

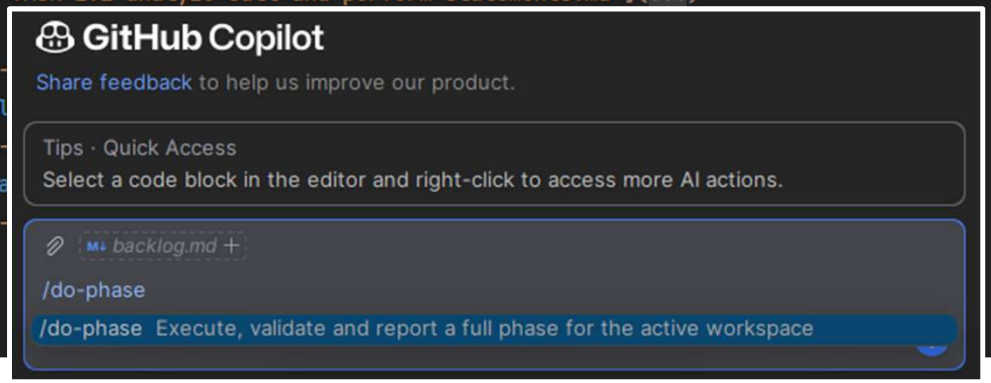
SPEC.md

Domain / Capability
Context
...
Quality Gates (DoD)

- Agents
  skills, tools, ...
- Planning
  phases, steps/tasks ...
- Agent 'Coordination'

Smals ICT for society

```
## phase1 — Inventory

- [x] **TASK-1.1** — Scan COBOL sources
  `status: Done` · [`backlog/tasks/task-TASK-1.1-scan-cobol-sources.md`](...)
- [x] **TASK-1.2** — Extract programs
  `status: Done` · [`backlog/tasks/task-TASK-1.2-extract-programs.md`](...)
- [x] **TASK-1.3** — Extract copybooks
  `status: Done` · [`backlog/tasks/task-TASK-1.3-extract-copybooks.md`](...)
- [x] **TASK-1.4** — Build inventory JSON
  `status: Done` · [`backlog/tasks/task-TASK-1.4-build-inventory-json.md`](...)
- [x] **TASK-1.5** — Validate G1 inventory gate
  `status: Done` · [`backlog/tasks/task-TASK-1.5-validate-g1-inventory-gate.md`](...)

## phase2 — Callgraph

- [x] **TASK-2.1** — Analyze CALL and PERFORM statements
  `status: Done` · [`backlog/tasks/task-TASK-2.1-analyze-call-and-perform-statements.md`](...)
- [x] **TASK-2.2** — Build callgraph
  `status: Done` · [`backlog/tasks/task-
- [x] **TASK-2.3** — Detect dynamic call
  `status: Done` · [`backlog/tasks/task-
- [x] **TASK-2.4** — Validate G2 callgra
  `status: Done` · [`backlog/tasks/task-

## phase3 — SQL Mapping
```



Phases, Tasks, … execution …

»Now

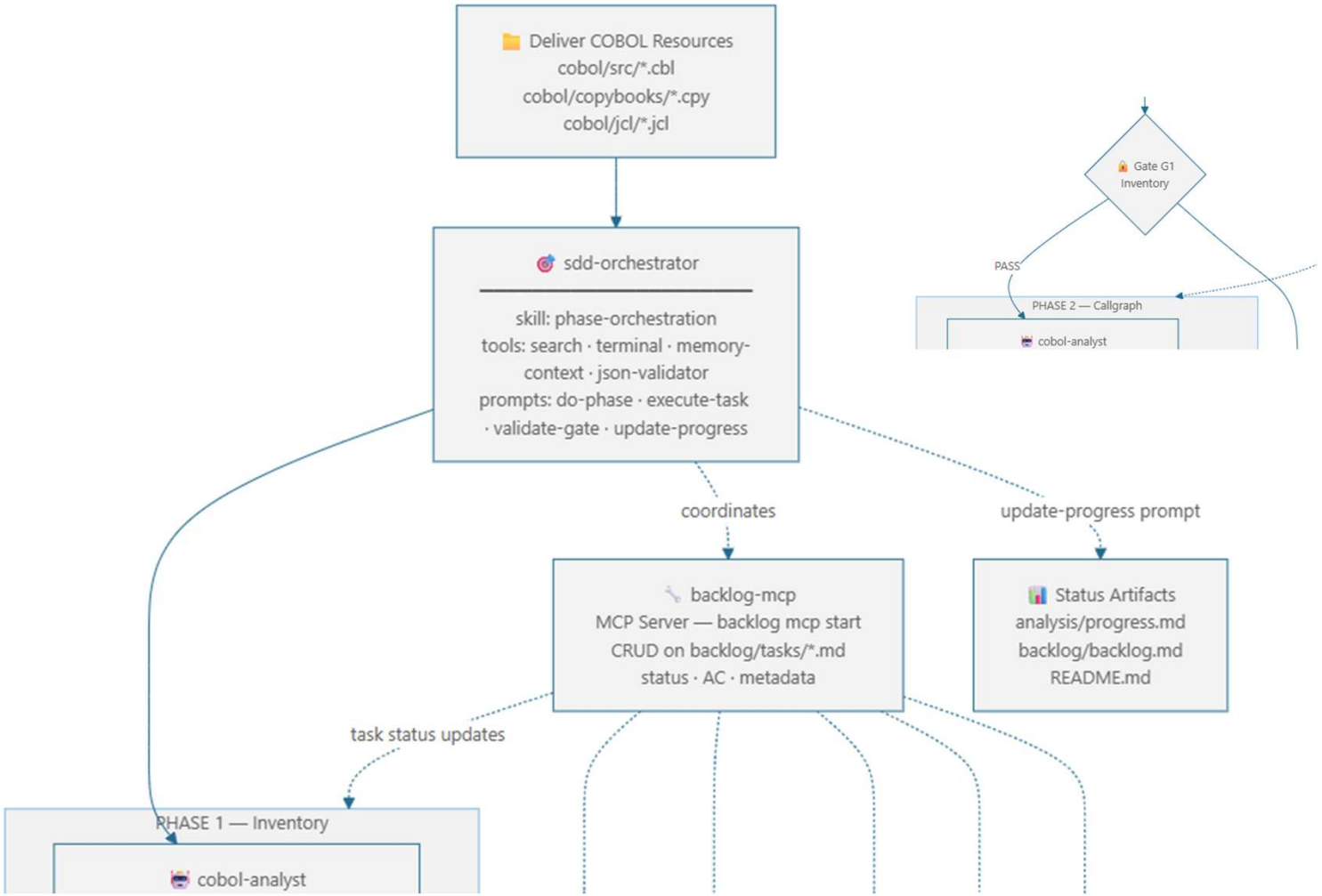
other approaches

SPEC.md

Domain / Capability
Context
…
Quality Gates (DoD)

- Agents
  skills, tools, …
- Planning
  phases, steps/tasks …
- Agent ‘Coordination’

Smals
ICT for society



Phases, Tasks, … execution …

»Now

other approaches

SPEC.md

Domain / Capability
Context
…
Quality Gates (DoD)

- Agents
  skills, tools, …
- Planning
  phases, steps/tasks …
- Agent 'Coordination'

Smals
ICT for society

SKILLs only

(jakarta revisited)

Smals
ICT for society

.agents / skills / socsec-jakarta-migration / SKILL

SKILL

Properties

name	socsec-jakarta-migration
description	Migrate Java/Maven applications from javax.* to jakarta.* and align with modern stacks across four supported paths: JBoss/WildFly EAP 7.4 to EAP 8 (Jakarta EE 10), standalone Java applications on JDK LTS 25+ (with or without Spring Framework), Spring Batch 4/...

+ Add property

Jakarta Migration

Guide the migration of a Java/Maven (or Gradle) project from a `javax.*` -era stack to a `jakarta.*` -era stack, across whichever of the following platforms applies:

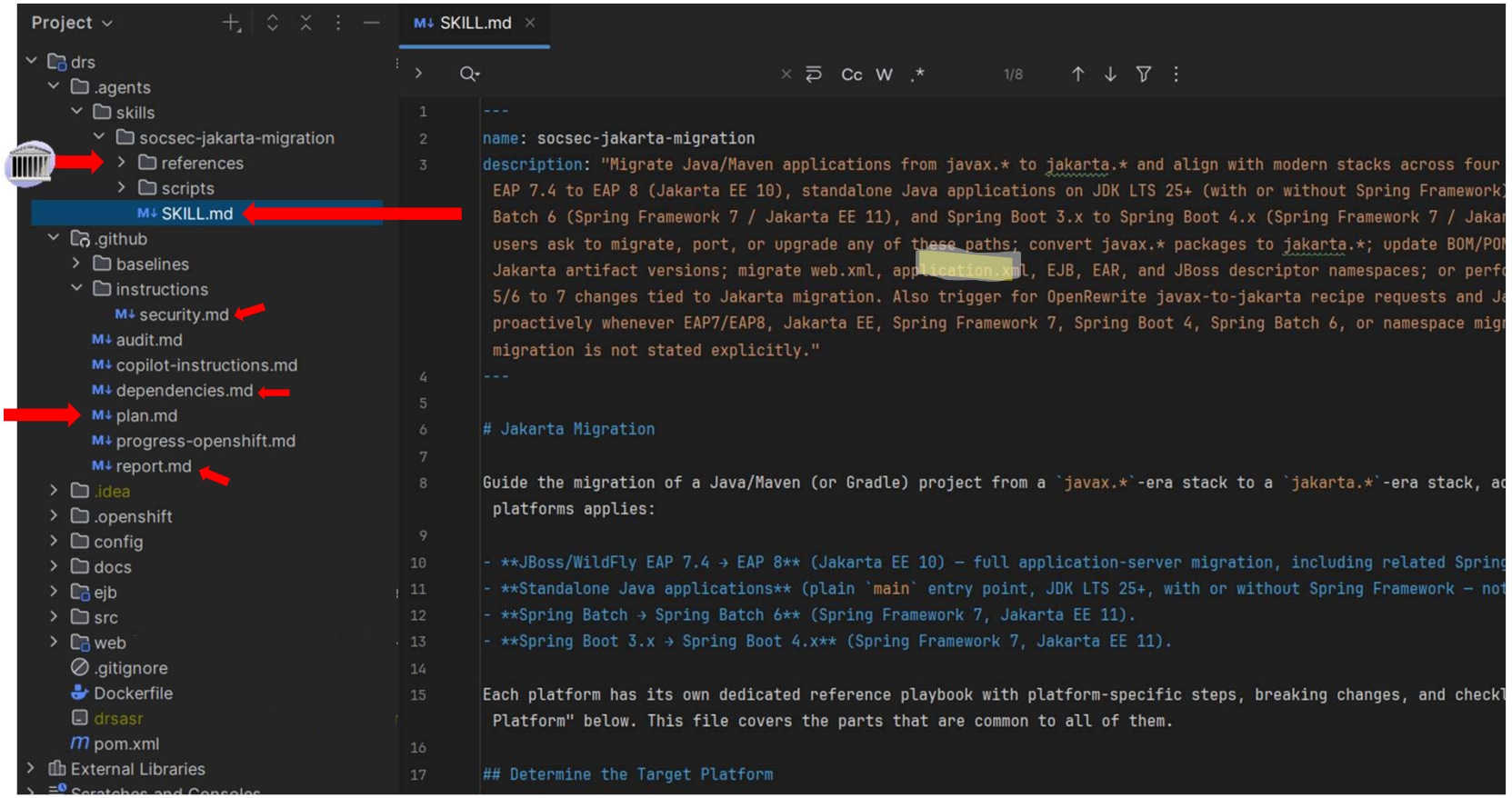
- **JBoss/WildFly EAP 7.4 → EAP 8** (Jakarta EE 10) — full application-server migration, including related Spring 5→6 and Hibernate 5→6 upgrades.
- **Standalone Java applications** (plain `main` entry point, JDK LTS 25+, with or without Spring Framework — not Spring Boot).
- **Spring Batch → Spring Batch 6** (Spring Framework 7, Jakarta EE 11).
- **Spring Boot 3.x → Spring Boot 4.x** (Spring Framework 7, Jakarta EE 11).

Each platform has its own dedicated reference playbook with platform-specific steps, breaking changes, and checklists — see "Determine the Target Platform" below. This file covers the parts that are common to all of them.

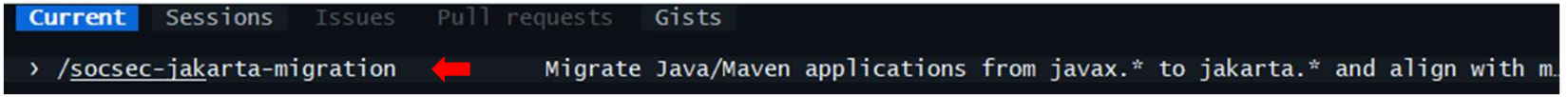
»Now
other approaches

SKILLs only

Smals
ICT for society



the SKILL (provided)



execute

»Now other approaches

SKILLs only

Smals ICT for society



Agenda

- ~~JakartaEE refresh~~
- ~~Jakarta Case, our approach (then)~~
- ~~Other approaches (now)~~
- Takeaways

Smals
ICT for society

Lessons Learned * Best Practices * Takeaways

- ✓ Start with a Sane Codebase … optimal test-coverage + clean Sensitive Data (or use 'synthetic data')
- ✓ Implement *Guardrails* … limit the capabilities of your AI assistant (sandboxing)
- ✓ Grounding … ensure sufficient supporting knowledge !
- ✓ Involve the Product Team from Day 1 … and provide feedback-loops
- ✓ Is all this Sufficiently Profitable? … Absolutely !
- ✓ The Best Approach? … as always, it depends on your case => be pragmatic !

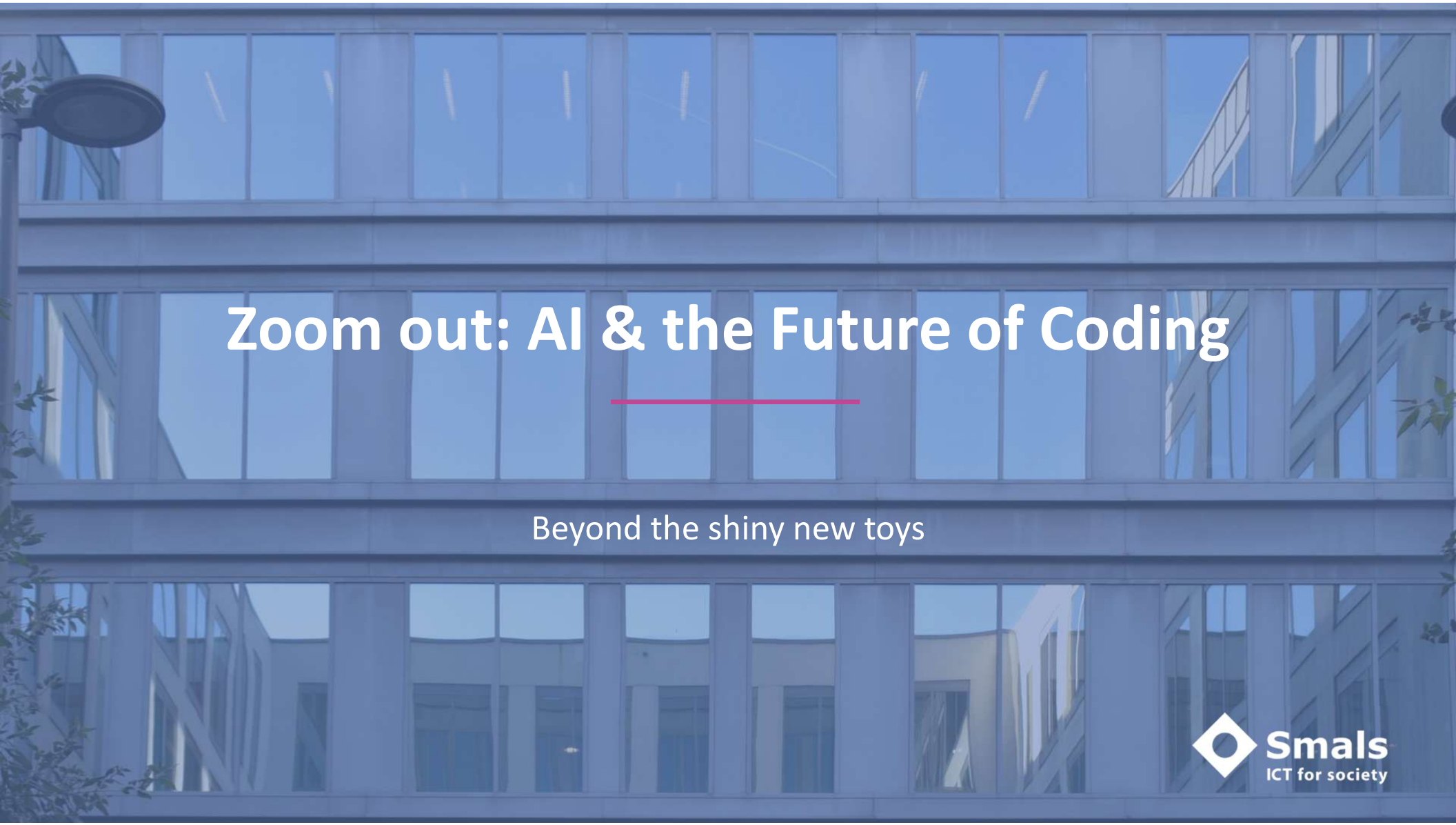
Smals
ICT for society



Agenda

- Introduction
- Cases & Preliminary Conclusions
- Zooming in on JavaEE to Jakarta Migrations
- **Zooming out on AI Development & the Future**
- Conclusions - lessons learned

Smals
ICT for society


Zoom out: AI & the Future of Coding
Beyond the shiny new toys
Smals
ICT for society

From Vendor lock-in to AI lock-in

- The smarter, bigger models work best: 3 or 4 big vendors: either choose them or be about 6 months to a year behind) → some **vendor lock-in** (especially when using public tender)
- Relatively easy to change between (the major) models; but how much less productive would we feel if we had to stop using AI altogether?

→ **AI lock-in**

- All those datacenters will have to be paid back: prices may increase when AI use is no longer optional
- And if we MUST use AI: how to manage its many other risks?

Smals ICT for society

GDPR + AI Act for the public sector

- Code + internal docs sent to a cloud LLM = personal data processing risk
  - Legacy projects especially, may contain secrets, or personal data in test sets, …
- AI Act risk classification:
  - code-assist tooling — usually 'limited risk' (transparency obligations)
  - But only for *clean code*
  - Even so, business logic could be somewhat sensitive
- For Belgian public sector: data residency is a real constraint
  - Testing software with actual data, using Cloud AI, is a no-go
- Great use case for Synthetic Data, pseudonymization, etc

AI Coding: a danger to developers?

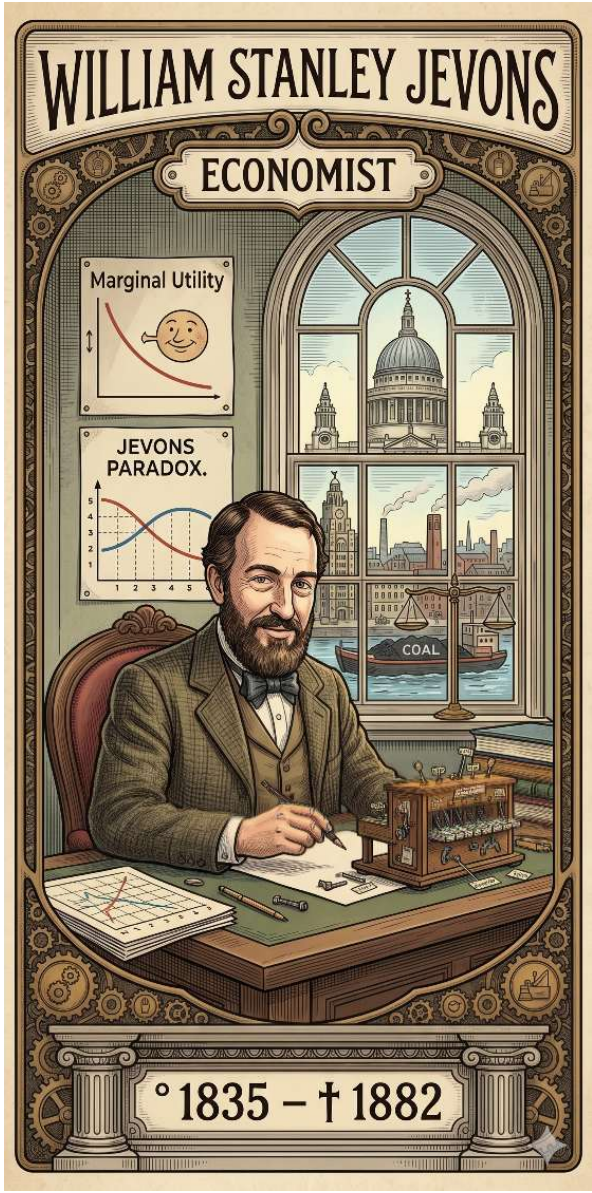
- Abundant AI use, and continuous improvement in models, harness, tools, …

  → **Software Development becomes cheaper/faster**

  So, will we need a lot less developers then?
  NO!? (at least for the next few years)

Jevons Paradox: there will be a lot more software development demand !
Because it becomes cheaper

- And without good engineers, quality will suffer (the CaveAt !!)
- Senior developers are still needed for the foreseeable future
  - You need deep insight to keep AI in check and help it deliver quality (**harness**)



Training implications — what to invest in

- Code-review and codebase analysis skills (*refuctoring* detection)
- Good software architecture
- Harness / context engineering; Requirements “Driving”
- AI tool and skill design
- *Domain knowledge*

Basically, being critical ☺

Smals
ICT for society



Agenda

- Introduction
- Cases & Preliminary Conclusions
- Zooming in on JavaEE to Jakarta Migrations
- Zooming out on AI Development & the Future
- **Conclusions - lessons learned**

Smals
ICT for society



Conclusions

Takeaways and Recommendations

Management Summary

Smals
ICT for society

Which legacy moves does AI support?

Anno 2026

↑ AI CAPABILITY

Understand / document	✓ Already works well	COBOL doc / chatbot · Word macros
Small translations / refactor	✓ Works — with supervision	Jakarta · SAS → Python
Build tooling around **legacy**	✓ Very good	COBOL dependency analyzers · Word-macro pipeline
End-to-end migration	− Only smallest programs/parts	COBOL · Java
Architecture rewrite	✗ Requires close supervision	human in the driver's seat

● Works today ● Partial / smallest cases — supervise ● Not yet — closely supervise

Smals ICT for society

Recommendations (1)

- Clean your code *before* handing it to a Cloud AI
- Use the strongest model as a sparring partner (planning phase)
  - Have it critique your approach
  - And push back on what the AI proposes
- Figure out the best harness and set of tools for the AI to use
  - For the more standard projects, we should look to **standardize / govern this ***
  - For legacy projects, it will require a more ad hoc approach
- Specifically instruct the AI to produce (human- *and* AI-) maintainable code !

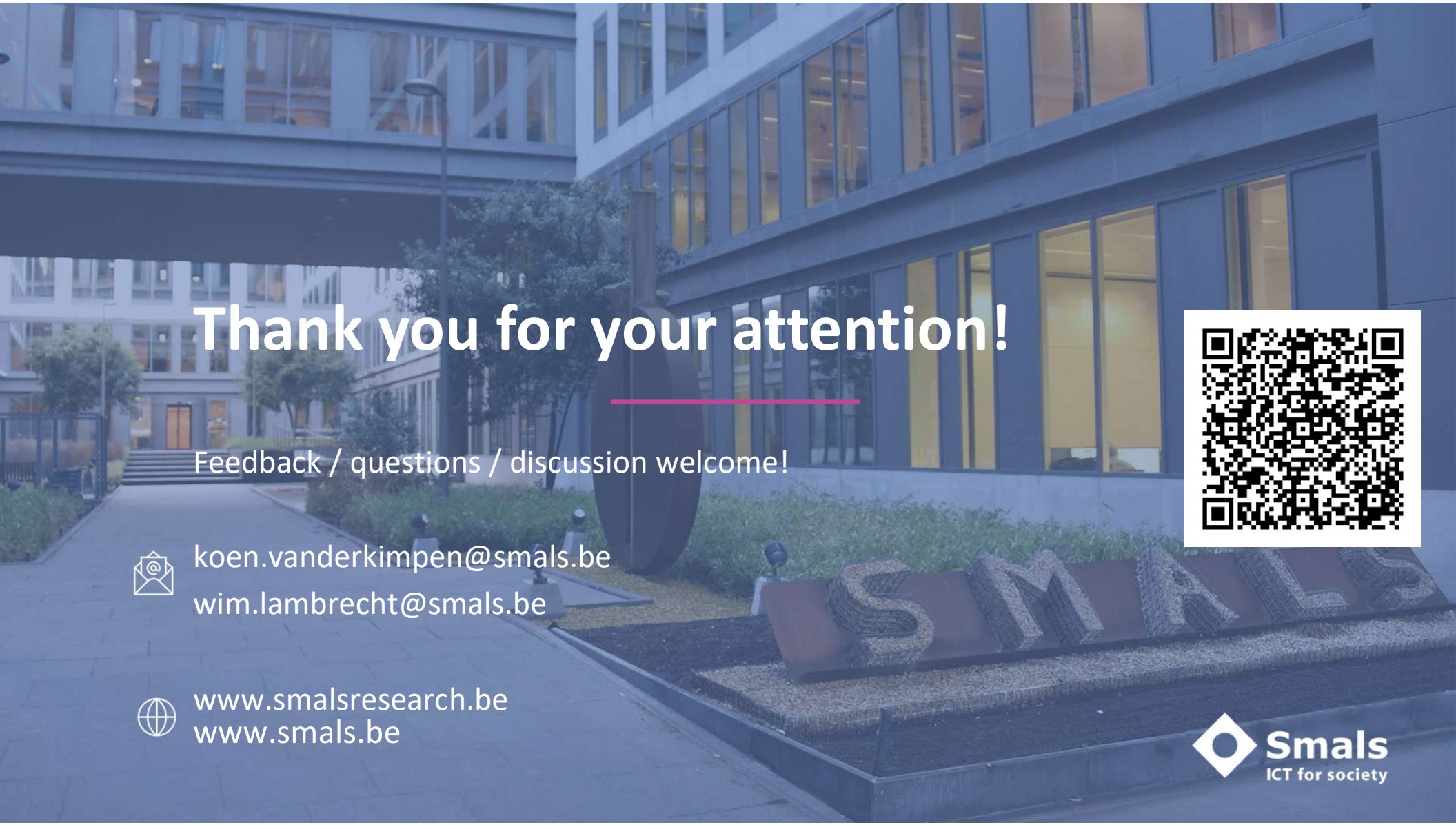
  ("refuctoring" danger)

Recommendations (2)

- Favour the indirect approach: Let the AI
  - Plan first, work in different phases
  - Build its tools first
  - Ask for your input
  - Propose alternatives at key moments
  - Review the work of another AI
- Build a shared repository of good practices / tools / skills

AI is not a silver bullet (yet); do not trust it, but also do not be afraid to use it

Smals ICT for society



Thank you for your attention!

Feedback / questions / discussion welcome!

koen.vanderkimpen@smals.be
wim.lambrecht@smals.be

www.smalsresearch.be
www.smals.be

Smals
ICT for society