AGILE TOUR — WORKSHOP 60 MIN

L’IA au service du *Craft.*

Spec-Driven Development avec SpecKit

Salah FOURAQ
Cloud Solution Architect · Hager Group

02 — LE SPEAKER

Salah Fouraq

Cloud Solution Architect — Hager

CE QUE JE FAIS

Architecte cloud — je conçois des plateformes et j'accompagne les équipes vers le craft.

MA CONVICTION

L'IA ne remplace pas le craft. Bien utilisée, elle l'amplifie.

CE QU'ON VA EXPLORER

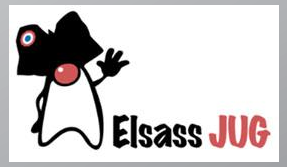
SpecKit — un outil qui remet la spec au centre, même (surtout) à l'ère de l'IA.

Agile Tour Strasbourg 2026 🥨

Hara

agile tour
STRASBOURG

Chargemap



{EPITECH}

Et si vos specs étaient *le meilleur code* que vous puissiez écrire ?

Pas le code que l'IA génère. La spec qui le décrit.

L'IA code vite.

Mais elle code quoi, exactement ?

Sans spec claire, on génère 10× plus de code… et 10× plus de dette.

HIER

On écrivait peu de specs.
On écrivait beaucoup de code.
La lenteur protégeait la qualité.

AUJOURD'HUI

L'IA écrit le code en secondes.
La spec devient le goulot.
Et aussi le levier.

3 signes que ça vous arrive déjà

01

L'AVALANCHE

L'IA produit du code plus vite qu'on ne peut le relire ou le tester.

02

L'AMNÉSIE

Le "pourquoi" disparaît. Reste un code que personne ne sait expliquer.

03

LA DÉRIVE

Chaque feature s'éloigne un peu plus de l'intention initiale. Sans bruit.

Spec-Driven Development

Et si on remettait *la spec au centre ?*

La spec devient l'artefact source. Le code en est la traduction.

4 qualités non négociables

01 **EXPLICITE**

Une intention claire, des contraintes nommées, rien laissé à l'interprétation.

02 **VÉRIFIABLE**

On peut écrire un test depuis la spec. Si on ne peut pas, elle est trop vague.

03 **VIVANTE**

Elle évolue avec le produit. Ce n'est pas un document figé posé en haut d'un wiki.

04 **EXÉCUTABLE**

Un agent IA doit pouvoir s'en servir pour décider quoi coder. C'est le test ultime.

SpecKit

Le SDD, opérationnalisé.

Un outil open-source (GitHub) qui pilote l'IA à partir d'une spec structurée. Au lieu de prompter du code, on prompte une spec — l'IA exécute.

OPEN-SOURCE

GitHub

AGENT-COMPATIBLE

Claude, Copilot, Cursor…

STACK-AGNOSTIC

Tout langage, tout projet

4 commandes, 1 boucle

PHASE 01

/specify

Décrire l'intention

Le QUOI et le POURQUOI.

PHASE 02

/plan

Choisir l'architecture

Stack, contraintes, découpage.

PHASE 03

/tasks

Découper en tâches

Liste exécutable, testable.

PHASE 04

/implement

Générer le code

L'IA exécute. Vous arbitrez.

Chaque étape produit un artefact validé avant de générer la suivante.

/specify

On décrit l'intention

Pas de code. Pas de framework. Juste : quel problème, pour qui, dans quel contexte.

À OBSERVER PENDANT LA DÉMO

- Le ton : on parle utilisateur, pas implémentation
- Les contraintes nommées explicitement
- Ce qui est volontairement laissé ouvert

→ *spec.md (l'artefact source)*

/plan

On choisit l'architecture

À partir de la spec, l'agent propose une stack, un découpage, des trade-offs.

À OBSERVER PENDANT LA DÉMO

- L'agent argumente ses choix
- On peut challenger, contraindre, raffiner
- La décision est tracée, pas devinée

→ *plan.md (les décisions techniques)*

/tasks

On découpe en tâches

Le plan devient une liste de tickets, chacun avec un critère de fin clair.

À OBSERVER PENDANT LA DÉMO

- Granularité : 1 tâche = 1 PR mentale
- Chaque tâche est testable indépendamment
- L'ordre suggère les dépendances

→ *tasks.md (la roadmap exécutable)*

/implement

L'IA exécute, vous arbitrez

Tâche par tâche, l'agent code, teste, commit. Vous restez l'arbitre du goût.

À OBSERVER PENDANT LA DÉMO

- Le code reste relisable, pas généré en bloc
- Les écarts sont signalés, pas dissimulés
- On peut revenir à la spec à tout moment

→ *commits (le code, traçable jusqu'à l'intention)*

Ce que vous venez de voir

L'INTENTION RESTE LISIBLE

Six mois plus tard, un dev qui arrive comprend pourquoi le code est comme ça. Pas juste comment.

L'IA EST DIRIGÉE

Elle ne devine plus. Elle exécute une spec qu'un humain a validée. Le craft revient à l'arbitrage.

LE CYCLE EST COURT

Un changement de besoin = un changement de spec = une régénération. Pas de réécriture aveugle.

Le code se génère. La pensée se documente. Et l'IA fait son meilleur travail quand elle sait pour quoi elle travaille.

SpecKit n'est pas pour tout

✓ OUI, ÇA VAUT LE COUP

- Nouvelle feature avec règles métier
- Refonte d'un module complexe
- Projet où plusieurs devs vont passer
- Domaine où le « pourquoi » compte

✕ PAS ENCORE

- Script jetable de 50 lignes
- Bug fix isolé, bien cadré
- POC qu'on jette dans une semaine
- Tâche purement cosmétique

Règle simple : si quelqu'un devra comprendre ce code dans 3 mois, la spec vaut l'investissement.

Ce qui change vraiment

Écrire du code → **Écrire l'intention**

Le code devient une sortie, pas l'objet.

Prompter l'IA → **Spécifier le besoin**

On sort du prompt jetable.

Reviewer du code → **Arbitrer des choix**

Le craft revient au jugement.

Ressources

ESSAYER

SpecKit

`github.com/github/spec-kit`

Le repo officiel. Quickstart, exemples, docs.

LIRE

Spec-Driven Development

`le manifeste GitHub Next`

Pourquoi la spec redevient l'artefact source.

PRATIQUER

Un kata SpecKit

`Votre prochaine feature`

Pas un side-project. Un cas réel, petit, cadré.

Discussion

Vos questions,
vos doutes, vos cas.

La meilleure spec, c'est celle qu'on écrit ensemble.

SALAH FOURAQ • CLOUD SOLUTION ARCHITECT • HAGER GROUP