C01 Calibration de Confiance & Mental Model

Conception Beginner

Comment penser la relation avec Claude pour en tirer le meilleur

Le bon mental model

La façon la plus productive de penser à Claude Code : un développeur junior très rapide, avec une connaissance encyclopédique des patterns, mais sans le jugement qui vient de l'expérience projet. Il peut produire 100 lignes de code solide en 10 secondes, et prendre une mauvaise décision d'architecture dans la même foulée.

Ce cadre a une implication immédiate : la supervision n'est pas optionnelle. Elle est calibrée selon le type de tâche.

Calibration par type de tâche

Tâche	Niveau de confiance	Action
Renommage, formatage	Élevé	Accepter sans relire
Génération de tests	Moyen-élevé	Vérifier la couverture
Refactoring d'un module	Moyen	Relire le diff
Décision d'architecture	Faible	Valider obligatoirement
Logique métier critique	Faible	Revue systématique

La règle générale : plus la décision est irréversible ou coûteuse à corriger, plus la validation est nécessaire.

Le piège de l'itération non supervisée

Le coût d'une erreur augmente de façon quasi exponentielle avec le nombre d'itérations non supervisées. Un mauvais choix de structure de données à l'étape 1 contaminera les 8 étapes suivantes.

La pratique recommandée : valider par petits incréments. Après chaque changement significatif, regarder le diff avant de continuer.

```
# Vérifier ce que Claude a réellement modifié
git diff --stat
git diff src/
```

Vous êtes le « main thread »

Claude Code exécute. Vous orchestrez. La distinction n'est pas stylistique, elle est structurelle : vous définissez les tâches, les priorités, et les critères de validation. Claude produit les artefacts. Cette répartition est ce qui permet de paralléliser plusieurs instances Claude sur des tâches indépendantes.

Ce que vous gardez toujours : les décisions d'architecture, les choix de bibliothèques, la définition des critères d'acceptation.

Ce que vous déléguez : l'implémentation dans un scope défini, la génération de boilerplate, les refactorings mécaniques.

Rewind : annuler une mauvaise direction

Si Claude part dans une direction problématique, `/rewind` permet de revenir à l'état précédent avant de ré-itérer. Mieux vaut rewindrer tôt que de construire 10 échanges supplémentaires sur une base bancale.

```
/rewind    # Annule les derniers changements
           # Revenir au dernier point sain
```

Pattern Amplification

Claude reproduit les patterns qu'il trouve. Dans une codebase bien structurée, il produit du code cohérent et idiomatique. Dans une codebase désordonnée, il amplifie le désordre. Si votre code manque de patterns clairs, fournissez-les explicitement dans `CLAUDE.md` plutôt que d'espérer que Claude les inférera.